Actes de la conférence CAID 2022

(Conference on Artificial Intelligence for Defense)

Organisée par

l'IA appliquée aux Systèmes multi agents

Essaim hétérogène pour le combat collaboratif reposant sur de l'apprentissage par renforcement multi-agent
Jacques Bois, Axel Puig, Loïc Rullière, Yonatan Teboul, Morgan Ossola, Alexandre Kotenkoff and Mathias Formoso

Algorithmes de placement optimisé de drones pour la conception de réseaux de communication
Zacharie Alès, Sourour Elloumi and Adèle Pass-Lanneau

L'IA au service de la cybersecéurité et Cybersécurité au service de l'IA

Empreinte de réseaux avec des entrées authentiques
Thibault Maho, Teddy Furon and Erwan Le Merrer

Comment tatouer un réseau de neurones à convolution de façon explicite et efficace
Mohammed Lansari, Katarzyna Kapusta and Vincent Thouvenot

Détection d'anomalies pour les réseaux smart-grids basée sur un autoencodeur LSTM
Joseph Azar, Youssef Laarouchi, Franck Bouzon and Raphaël Couturier

Modèles de Détection de Malware et de Ransomware basés sur l'IA
Benjamin Marais, Tony Quertier and Stéphane Morucci

TrustGAN : Apprentissage de réseaux de neurones profonds, sûres et confiants, grâce aux réseaux antagonistes génératifs
Hélion du Mas des Bourboux

Traitement de signaux radar par IA

Solution de Deep Learning robuste pour entrainer des modèles d'ATR SAR avec des données MOCEM entièrement synthétiques
Benjamin Camus, Corentin Le Barbu and Eric Monteux

Désentrelacement et classification des émetteurs RADARs basés sur l'utilisation des distances de Transport Optimal
Manon Mottier, Gilles Chardon and Frédéric Pascal

Traitement d'image et de signaux par IA

Détection de navires embarquable à bord de satellites optiques
Thomas Goudemant, Benjamin Francesconi, Houssem Farhat, Lionel Daniel, Erwann Kervennic, Seif Mzoughi, Adrien Girard and Olivier Thiery

Estimation de la trajectoire d'un projectile : une approche basée sur les réseaux LSTM
Alicia Roux, Sébastien Changey, Jonathan Weber and Jean-Philippe Lauffenburger

L'apprentissage auto-supervisé

Promesses et Limites de l'Apprentissage Auto-Supervisé pour le Traitement Automatique de la Parole
Lucas Maison, Marcely Zanon Boito and Yannick Estève

Préentraînement auto-supervisé sur imagerie satellite : un cas d'étude sur la détection de véhicules efficiente en annotation
Jules Bourcier, Thomas Floquet, Gohar Dashyan, Tugdual Ceillier, Karteek Alahari and Jocelyn Chanussot

Apprentissage auto-supervisé appliqué à la détection d'erreurs de labélisation
Tristan Bitard-Feildel, Amélie Chérubin and Corentin Lamboley

Heterogeneous Swarming for Collaborative Combat using Multi-Agent Deep Reinforcement Learning

Jacques Bois
MBDA
CentraleSupélec, Uni. Paris-Saclay
Paris, France
jacques.bois@mbda-systems.com

Axel Puig
MBDA
CentraleSupélec, Uni. Paris-Saclay
Paris, France
axel.puig@mbda-systems.com

Loïc Rullière
MBDA
Paris, France
loic.rulliere@mbda-systems.com

Yonatan Teboul
MBDA
Paris, France
yonatan.teboul@mbda-systems.com

Morgan Ossola
MBDA
Paris, France
morgan.ossola@mbda-systems.com

Alexandre Kotenkoff
MBDA
Paris, France
alexandre.kotenkoff@mbda-systems.com

Mathias Formoso
MBDA
Paris, France
mathias.formoso@mbda-systems.com

***Abstract* — Development of future weapon systems heavily relies on simulations where several agents interact in an environment. In such an environment, optimised decisions are crucial to leverage the capability of collaborative systems in a congested, cluttered, contested, connected, constrained [1] battlespace. In this work we present a simplified, yet complex, scenario derived from a notional air-to-surface mission featuring a set of heterogeneous effectors collaborating over a communication network in order to detect and engage a number of defended, relocatable ground targets. We propose a method based on a multi-agent, distributed version of SAC (Soft-Actor-Critic) and we highlight three benefits of our method: 1- Reinforcement Learning (RL) outperforms greedy and semi-random algorithms on a set of operational metrics. 2- Collaboration improves the global performance of the teamed agents. 3- Curriculum Learning is central to improve and speed-up the training of the agents.**

Keywords — Multi-Agent, Reinforcement Learning, Distributed Systems, Collaborative Decision Making. (*key words****)***

I. INTRODUCTION

Reinforcement Learning is a method for optimising the actions of an agent in an environment in order to maximise a reward function. The agent thus learns by trial and error by interacting with its environment. This method of machine learning gained momentum at the beginning of the 21st century, especially in automatics [2]. In the past few years, the use of deep neural networks made it possible to solve complex nonlinear problems such as Atari games [3] and more recently the game of Go [4].

Recent research in Reinforcement Learning focused on multi-agent problems where the reward no longer depends on the actions of a single agent but on its action combined with that of the other agents. This method made it possible to exhibit collaborative behaviours with a better performance than single agent RL [5] on a different set of problems, ranging from traffic regulation [6] to the coordination of drone fleets [7].

Armed forces are increasingly interested in collaborative combat, which requires the mastery of many new technological subjects at the crossroads of technology, tactics, doctrines and ethics. In particular, mastering fleet communications [8], formation flying of autonomous fleets [9], as well as tactical cooperation & coordination between air assets, whether manned or unmanned [10][11]. In addition to these topics, collaborative combat also increases the number of possible strategies for a given mission. Several works propose approaches to establish these strategies, with control laws derived from automatics [12], approaching the subject as an optimisation problem [13], or by getting inspired from animal strategies [14]. Finally, some propose more complex approaches combining different types of algorithms [15].

The present work proposes to approach the problem of defining collaborative combat tactics as a reinforcement learning problem. Several works highlighted the relevance of such algorithms for this type of problems [16]. RL is efficient to manage complex situations while bringing out innovative strategies, therefore its use in the context of collaborative combat seems to be increasingly relevant.

II. FRAMEWORK AND ENVIRONMENT

A. Markov Games framework

We consider the framework of Markov Games (Littman, 1994). It extends the framework of Markov Decision Process, to multi-agent problems.

We define:

- A set of states S,
- Action sets for each of N agents $A_1, \dots, A_N$,
- A state transition function which represents the probability distribution over possible next states, given the current state and actions $T: S \times A_1 \times \dots \times A_N \rightarrow P(S)$,

- A reward function for each agents that depends on the global state and actions of all agents $R_i: S \times A_1 \times \ldots \times A_N \rightarrow \mathbb{R}$.
- The observations of each agents, which contain partial information from the global state $s \in S$: $o_1, \ldots, o_N$
- The policy of each agent $\pi_i: O_i \rightarrow P(A_i)$ which returns a distribution over the set of actions given an observation,
- γ is a discount factor that determines how much immediate rewards are favoured over long-term gain.

In this framework, the objective of the ***i*th** agent is to find a policy that maximises their expected discounted rewards:

$$\pi_i = \underset{\pi}{\operatorname{argmax}} J_i(\pi)$$

$$J_i(\pi_i) = \mathbb{E}_{a_1 \sim \pi_1, \ldots, a_N \sim \pi_N, s \sim T}\left[\sum_{t=0}^{\infty} \gamma^t r_{it}(s_t, a_{1t}, \ldots, a_{Nt})\right]$$

B. *Environment and model*

Our environment contains two Surface Based Air-Defence (SBAD) systems that can move, or engage an asset and then fire. A Probability of Kill (Pk) is associated with the firing. The two systems have different ranges: short-range (SACP) and medium-range (SAMP). These SBAD systems follow a loop with the following steps:

- Deployment: preparation for engagement (10 min): no movement, no engagement;
- Engagement if possible
- Dismantling: preparation for motion (5 min): no movement, no engagement
- Relocation: random motion for 10 minutes at 5kph.

8 Multiple Rocket Launcher (MRL) systems can fire their surface-to-surface rounds and follow a ”shoot and scout” tactic:

- Fire (5 minutes) at a frequency of 4 rounds per minute;
- Movement (10 minutes): they move at 15kph in random direction and stay in pairs. They stay in the area protected by the air defence systems. The goal is to optimise the policy of the assets, while the behaviour of other elements are coded to be representative of their real operation, to minimise the number of rounds fired by the MRL systems (which therefore encourages their neutralisation).

The environment also contains a heterogeneous group of air assets, each one controlled by a reinforcement learning agent (blue team):

- 4 Observation assets (DRIL): they detect, recognise, identify and locate the elements in their field of view. This information is perfectly and instantaneously transmitted among all assets;
- 2 Jamming assets (EW): they can either activate the jamming mode, to reduce the Pk (probability of kill) of SBAD systems located in the jamming field of the asset, or the observation mode (ELS - Emitter Location System), to detect, identify and locate the elements in their field of view, with different performance parameters from DRIL assets;
- 8 Attacking assets (WPN): they can neutralise targets, either SBAD or MRL systems, by direct impact. They stay in a waiting area until they take the action “attack mode” (see below).

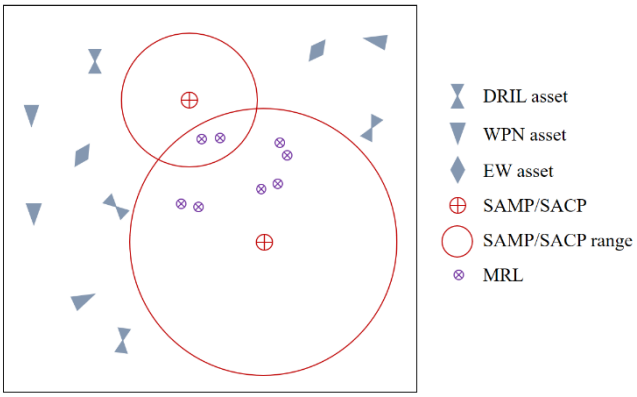


Fig. 1. Schematic representation of the simulated battlefield and its components

C. *State and action spaces*

The state space is the same for each asset. It contains the position and velocity of itself and of the other assets, the types and actions of the other assets, the position, heading and type of detected SBAD and MRL systems.

The action space differs depending on the type of assets. The action space of all the assets is composed of the orientation of the velocity vector. The action space of the WPN assets also contains the activation of an attack mode, and that of EW assets contains a choice of a mode which can be observation (ELS) or jamming.

D. *Multi- Agent Soft Actor-Critic (MASAC)*

Actor-Critic method [17] is a modified version of policy gradients methods [18][19] which reduces the high variance problem encountered in the original works.

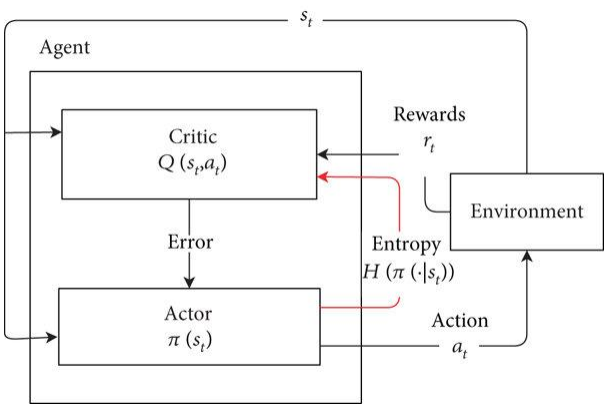


Fig. 2. Soft actor-critic algorithm

The algorithm has two parts. An actor samples actions in the environment, while a critic computes the gradient based on the temporal difference error. Decoupling actions sampling and policy updates allows to average experience and to reduce variance.

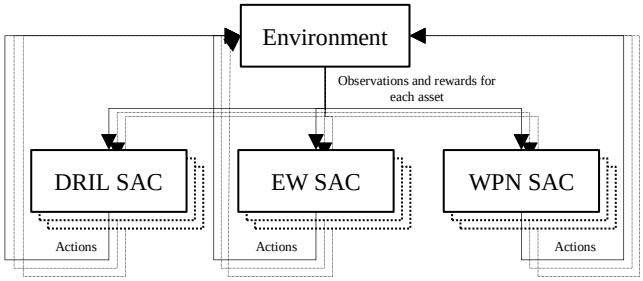


Fig. 3. MASAC: A SAC for each type of asset, replicated for each of the assets.

Three different actor-critic models are considered, one for each type of asset. They are then replicated according to the number of assets, since all the assets of a same type have the same role and can be exchanged. Every asset gets its own observations and rewards, so even if they follow the same policy, they take different actions.

This method, we call Multi-Agent Soft Actor Critic (MASAC), has several advantages. First, since all the assets of a same type learn the same policy, they can be swapped without consequences. Moreover, this solution is decentralised, therefore the loss of an asset should have a limited impact on the performance.

Every time a simulation is run, the observations, actions and rewards of all the assets of the same type are concatenated to train the actor and critic neural networks corresponding to this type of asset.

E. A distributed reward function

Table I below presents the different terms in the reward function.

In particular, some terms are distributed either individually to the involved asset, either partially to the asset group (DRIL, EW, WPN), or globally to the entire swarm. These distributed terms in the reward function may foster collaborative swarm actions.

Most of the terms are continuous, in order to tune the right balance between exploration time and providing arbitrary, subjective information to agents through their reward functions. However, some terms are discrete, in order to try and highlight a remarkable action led by the swarm of assets.

Reward terms are arbitrarily given an importance level according to their meaning. Table I presents these importance levels, which are scaled through fine-tuned coefficients.

Finally, mathematical functions are defined in order to quantify the terms presented in the table below. For instance, the anticollision reward function depends on the distance between the assets ; the detection reward function depends on the number of red assets detected by the blue team at each step ; the SBAD system neutralisation function depends on the distance between WPN assets and SBAD systems, it also has a discrete component depending on the effective neutralisation of SBAD systems ; the hazard reward function depends on the time spent by the blue assets in the interception domains of SBAD systems.

TABLE I. DECSRIPTION OF THE MASAC-DISTRIB REWARD FUCTION

Justification	**Nature**	**Distribution**	**Temporal nature**	**Importance**	**Order of appearance[a]**		
					DRIL	**EW**	**WPN**
Anticollision	Penalty	Individual	Continuous	Very high	1	1	1
Detection/Recognition/ Identification/Location	Reward	Partial	Continuous	Medium	2	2	-
Jamming	Reward	Partial	Continuous	Low	-	2	-
MRL system neutralisation	Reward	Global	Continuous, punctual	Very high	3	3	2
SBAD system neutralisation	Reward	Global	Continuous, punctual	High	3	3	2
Blue asset attrition	Penalty	Global	Punctual	High	4	4	4
Risk level (toward attrition)	Penalty	Individual	Continuous	Low	4	4	4

[a.] See subsection III-C.

III. EXPERIMENTAL METHOD FOR THE DEVELOPMENT OF COLLABORATIVE STRATEGIES

A. Evaluation metrics

Two **operational evaluation** metrics are defined. They reflect the operational objectives set by the mission commander of the blue team. These metrics are representative of a notional counter-battery scenario. They are, in order of priority:

1. **Minimise the number of rounds fired by the opposing weapons systems** (MRL systems). We measure the percentage reduction of the number of rounds fired compared to the maximum.
2. **Reduce the effectiveness of SBAD systems** to facilitate a follow-on attack. We measure the percentage reduction, averaged over 1 episode, of the number of operable surface-based air defence (SBAD) systems. An "operable SBAD" is neither jammed nor neutralised.
3. **Minimise attrition in the swarm of assets.** We measure the percentage of surviving assets at the end of the mission.

The following **technical metric** is averaged on a batch of episodes and allows to monitor the learning process evaluation.

- Variations in the reward function: average reward and standard deviations achieved on a mission, all smoothed over 10 missions.

B. Baseline methods

To benchmark our method, we used the following baselines:

- **ALEA.** Semi-random algorithm: decisions are made by generating random actions uniformly and independently, and applying a correction on the three components of the asset velocity to smooth the trajectory and to keep the mobiles inside the playground. For discrete actions, a threshold is applied. This algorithm ensure that the problem is not trivial to solve.
- **GREEDY.** Decisions are made by maximising the next step reward. This myopic strategy should give poor results if the reward function does not reduce the problem to an easy optimisation problem.
- **MASAC.** Algorithm defined in Section II, but without the collaborative incentives: all terms of the reward function are distributed individually. The goal is to measure the contribution of the partial or global distribution of the terms of the reward function.

In contrast to MASAC, **MASAC-Distrib,** our algorithm, uses a distributed reward function as described in Table I.

C. Curriculum learning

The definition of a training strategy reduces the complexity of the problem and improves the convergence and the results of the RL agent, while controlling the effectiveness of the learning [20]. It consists in a set of elementary functions, increasing by complexity, compounding a curriculum [21]. In our case, the task of each training phases are encompassing previous ones [22]. The 5 training phases are summarised below:

1. **ANTICOLLISION.** In the reward function, only the anticollision term is present (cf Table I). The goal of this phase is to teach the assets to fly and avoid collisions with each other and with the ground.
2. **SPECIALISATION.** The "specialisation" terms of the reward function (observation, jamming, attack) are introduced in Table I. In this phase only, WPN assets are omniscient about the positions of enemy assets. Assets learn how to neutralise targets without depending on the positions provided by other assets: the goal is to reduce the exploration needed to neutralise targets. The overall objective of this phase is to teach the assets their "expertise", assuming the others can do theirs.
3. **COLLABORATION.** In this phase, attacking assets depends on the information provided by the other groups of assets. In the case of the MASAC-Distrib algorithm, the reward function is modified to distribute rewards globally or partially according to Table I. The goal of this phase is to teach the assets to collaborate with each other. WPN assets are not omniscient anymore and rely on information provided by DRIL and EW assets.
4. **SURVIVE.** In this phase, risk and attrition penalties are added to the reward function. The goal is to encourage assets to consider the risk of surface-based air defence (SBAD) threats.
5. **ADAPTATION.** Finally the adaptation phase consists in training the agents to adapt to variations of ground mobiles initial positions. In order to progressively increase the complexity of the scenario, agents were trained on close to fixed initial positions. Prior to this phase, the initial positions of the mobiles on the ground varied by a maximum of 15 km between two missions. During this training phase, these initial positions varied by a maximum of 80 km between two missions, i.e. over the entire playground. This prevents the agent to memorise the zones in which enemy mobiles are to be found.

At each phase of curriculum learning, the weights of the neural networks are initialised using the weights trained during the previous phase. Hyperparameters are reinitialised and then monitored thanks to the evolution of the reward function and Q-values.

IV. RESULTS AND ANALYSIS

A. Experimental results

The plots of the rewards and Q-values obtained by MASAC and MASAC-Distrib algorithms showed the scientists the average rewards and cumulated Q-values obtained for each agent over the adapted number of missions. The training was systematically stopped when the reward function and the Q-values converged.

In the remaining parts of this section, we present the radar chart of the 3 operational metrics defined in subsection III A.

1) *Curriculum learning results*

a) ANTICOLLISION

Behaviour. Both algorithms MASAC and MASAC-Distrib are not differentiated since their reward function is the same at this stage. However, anti-collision measures were observed.

Metrics. The training seems to be satisfactory, considering the rewards obtained here, and the decrease in the number of collisions between RCs or with the ground, as shown in Table II.

TABLE II. RESULTS, ANTICOLLISION TRAINING PHASE

Batch	**Number of collisions *per hour* between RCs or with the ground**	
	AVG	**STD**
First 30 missions of the training phase	3.15	1.68
Last 30 missions of the training phase	0	0

b) SPECIALISATION

At this stage, the two algorithms MASAC and MASAC-Distrib are not differentiated, since their reward function is the same.

Behaviour. At the end of the training phase, the observed behaviours changed significantly: DRIL, EW and WPN assets fly on average much more in areas where SBAD defences and MRL systems are found. The assets learn to avoid triggering air-defence firings.

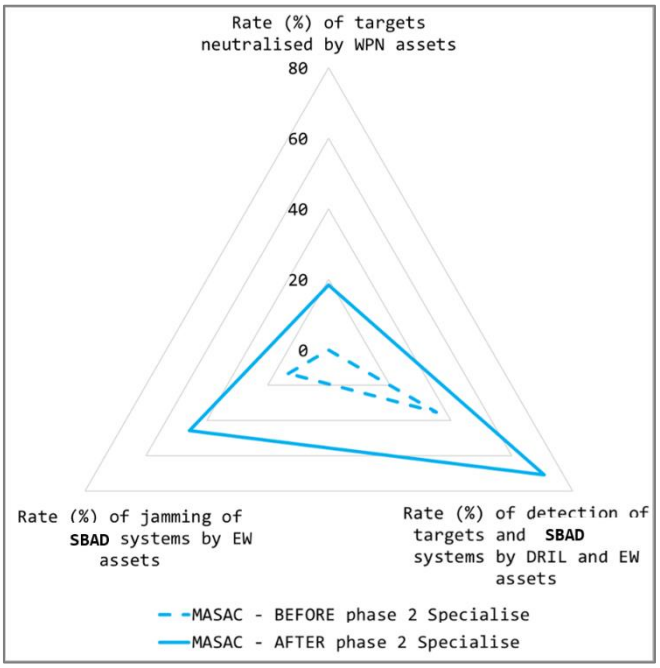


Fig. 4. Functional assessment, training phase N°2 Specialisation

Metrics. Fig. 4 allows to compare MASAC performance before and after the training phase: the training seems to be successful.

c) COLLABORATION

Behaviour.

MASAC-Distrib. At the end of the training phase, the observed behaviours changed significantly: DRIL, EW and WPN assets fly in formation in areas where surface-based air defence systems and targets are found. WPN assets wait until DRIL and EW assets have already spotted enemy mobiles before attempting to engage them. Tactics to avoid surface-based air defence systems are also in place.

MASAC. The training features much lower collaborative behaviour.

Metrics. According to Fig. 5-A, the training phase is successful for both algorithms, and for all metrics, in particular for MASAC-Distrib.

d) SURVIVE

Behaviour. At the end of the training, and for both algorithms, there is no significant change in the behaviour of the assets, except that they spend less time in the interception domains of ground-air systems.

Metrics. According to Fig. 5-B, the training phase is successful for both algorithms, regarding the surviving rate metric, in particular for MASAC-Distrib.

e) ADAPTATION

Behaviour. The observed behaviours are very similar for the MASAC-Distrib and MASAC algorithms. At the beginning of the training phase, the behaviour of the assets overfits the previous configuration of the mission area. The assets evolve in the area without taking into account the new location of the ground mobiles. Targets are no longer hit and assets run straight into the surface-based air defence (SBAD) systems' interception zones. At the end of the training, assets are able to adapt to the different positions adopted by the ground mobiles.

Metrics. According to Fig. 5-C, the significant variations in initial positions slightly reduces the global performance of both algorithms. In return, both algorithms improve their generalisation capability.

f) *Training from scratch*

In this experimentation, the agent is trained on the entire reward function *from scratch*: there is no curriculum learning strategy.

Behaviour. At the end of the training phase, the observed behaviours changed significantly, compared with the non-trained algorithm: the assets fly in the areas where the surface-based air defence systems and targets are found. No particularly collaborative behaviour are observed.

Metrics. According to Fig. 5-D, the curriculum approach significantly improves the global algorithm performance.

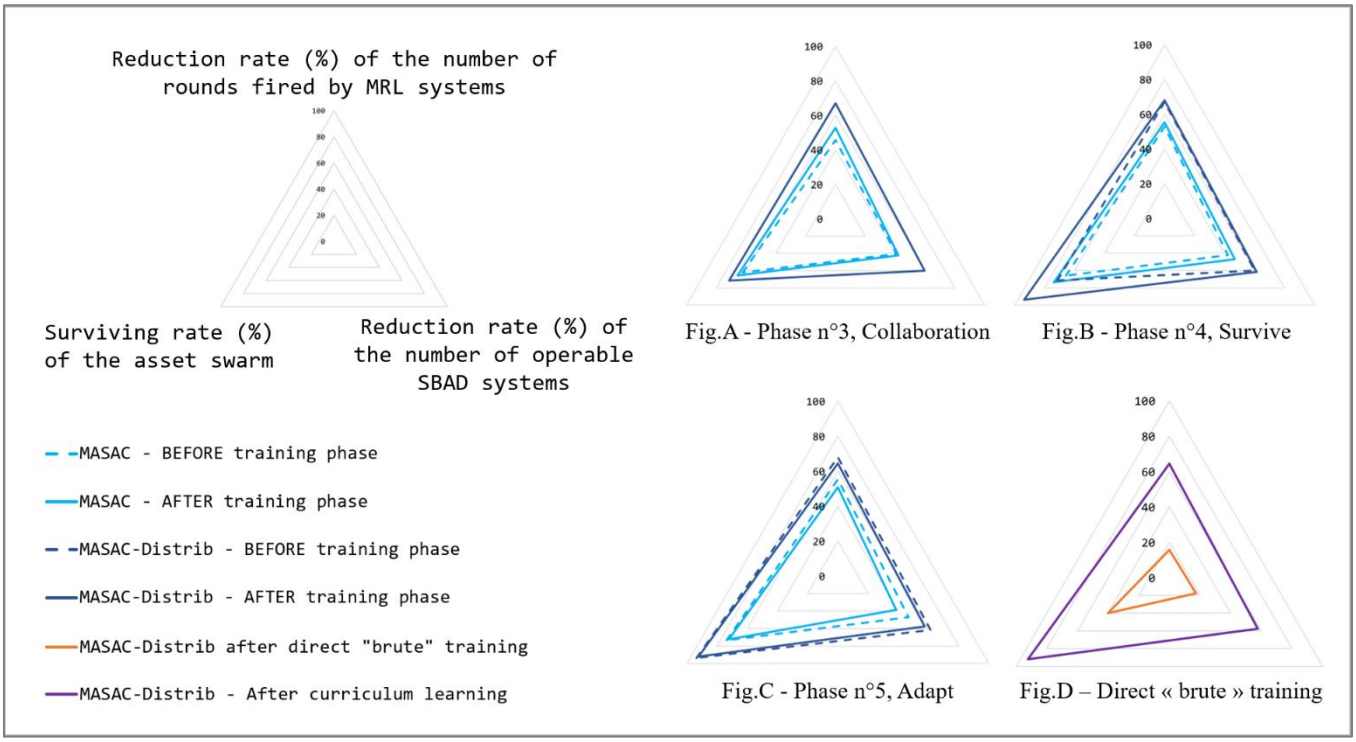


Fig. 5. Operational assessment, curriculum learning, phases n°3, 4, 5 and training *from scratch*.

2) Operational evaluation and benchmark

Behaviour.

ALEA. The behaviour of the agents controlled by the ALEA algorithm practically corresponds to a random walk, except that we impose that the agents remain in the mission zone. The trajectories being artificially smoothed, this last constraint explains the observation of sometimes less erratic behaviour.

GREEDY. The GREEDY algorithm allows the agents to fulfil some of their operational objectives. No collaborative behaviour is observed.

MASAC. The behaviour of the agents controlled by MASAC seems more intelligent and adaptative than the behaviour of the ALEA and GREEDY algorithms. In particular, we sometimes observe collaborative behaviours, and especially an excellent adaptation to the variations of the scenario.

MASAC-Distrib. The MASAC-Distrib algorithm takes up the clear improvements observed between MASAC and GREEDY, to which are added **numerous collaborative behaviours between agents**, in particular regarding target designation and coordination of jamming and attacks.

Finally, it should be remembered that the simulation, and therefore the observed behaviours, are **simplified** compared to a real environment**.** While the adaptability of the agents seems satisfactory within training phase N°5 ADAPT, it is not certain that the agents adapt correctly to greater variations in the scenario, nor real wartime conditions.

Metrics. In the following section, MASAC-Distrib is benchmarked against the algorithms presented in section III-C. The results are plotted as a bar chart featuring a confidence interval shown in black for each algorithm. Results were averaged over 100 missions.

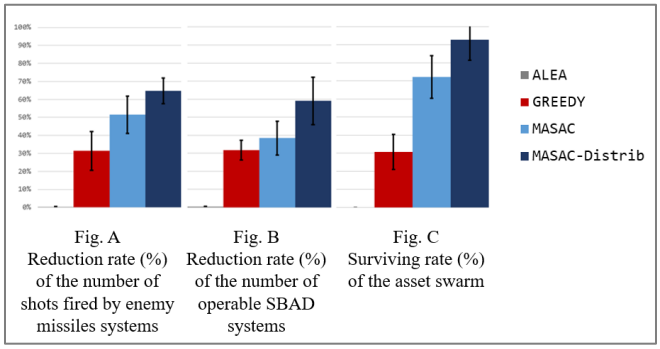


Fig. 6. Results, operational metrics in exploitation (test) mode

TABLE III. RESULTS SUMMARY

Algorithm	**Reduction rate (%) of the number of rounds fired by MRL systems**		**Reduction rate (%) of the number of operable SBAD systems**		**Surviving rate (%) of the asset swarm**	
	AVG	*STD*	**AVG**	*STD*	**AVG**	*STD*
ALEA	**0**	*0*	**0**	*0*	**0**	*0*
GREEDY	**31**	*11*	**32**	*6*	**31**	*10*
MASAC	**51**	*10*	**38**	*9*	**72**	*12*
MASAC-Distrib	**65**	*7*	**59**	*13*	**93**	*11*

Table III (above) summarises the results of Figure 6. These results were obtained by running each algorithm over 100 missions, in its evaluation mode.

B. Results analysis

1) Benchmark: a strong interest for DRL algorithms

It can be seen from Fig. 6 that the performance of the algorithm on the main operational metric is systematically ranked in the following order, from worst to best performance: ALEA, GREEDY, MASAC, MASAC-Distrib.

The semi-random algorithm ALEA completely fails to meet the operational objectives. The very low, but non-zero, observation and interference rates are due to the random exploration of the mission area by the DRIL and EW assets. The performance of the other algorithms is above that of ALEA, which confirms that problem is not trivial.

The functional performance of the MASAC algorithm is consistently above the performance of the GREEDY algorithm in all considered metrics (see Fig. 6). **DRL achieves results at least similar to a classical deterministic greedy approach in this scenario**.

Standard deviations are represented by black bars in Fig. 6. These **standard deviations are satisfactory** from a statistical point of view, as they allow a perfect distinction between all MASAC and MASAC-Distrib results. Nevertheless, they represent a significant proportion of the mean values with which they are associated. Reducing these standard deviations may improve the stability of the decisions, and thus the confidence that can be placed in these algorithms. It should be noticed, however, that **the GREEDY algorithm obtains standard deviations with the same order of magnitude as both DRL algorithms.**

Finally, according to section IV-A-2, where the final behaviours of MASAC and MASAC-Distrib are compared to other algorithms, **the interest of DRL seems to lie in a substantial improvement of the adaptability of the agents**.

2) Collaboration: interest of distributed reward functions

The secondary metrics may be divided into two classes:

- Performance measures of coercive actions, such as attrition rates of assets, surface-based air defence systems and targets,
- Performance measures of distant actions, such as MRL availability rates, jamming rates, and observation rates.

The MASAC and GREEDY algorithms obtain disparate results on the former, and relatively close on the latter. Paradoxically, remote actions such as jamming or observation are more "direct" than coercive actions: they do not imply a chain of actions and rely little on information provided by other assets, as they themselves are the source of information on positions for example. Coercive actions, on the other hand, require knowledge of the position of mobiles on the ground, or the implementation of collaborative strategies to reduce attrition while seeking to meet operational objectives, for example. The significant difference between the performance of the GREEDY and MASAC algorithms on coercive action measures could thus be explained by a **better collaboration in the MASAC case than within the GREEDY algorithm**. This collaboration encompasses the attacks of WPN assets relying on sensing and jamming performed by DRIL and EW assets.

Moreover, the MASAC-Distrib algorithm consistently performs better than the MASAC algorithm. This corroborates the results and behaviours identified in the training and **confirms the interest of the partial or global distribution of terms of the reward function**.

3) Contribution of curriculum learning

The training *from scratch* phase allows for a substantial improvement in results, as shown in Fig.5-D, but these results are very far from what is obtained with the sequential curriculum training. **The interest of progressiveness is therefore validated.**

Moreover, the MASAC-Distrib behaviours observed at the end of the training *from scratch* phase present relevant features, such as assets systematically flying over the targets area, but no collaborative behaviour is observed, such as collaborative target designation or collaborative spatial deployment. On the contrary, the MASAC-Distrib algorithm, trained with a curriculum approach, present numerous collaborative behaviours between agents, such as: collaborative target designation, coordination of jamming and attacks, formation flight, and collaborative planning of actions. **Collaborative swarm behaviour may complexify the training, in our case, curriculum is a key enabler to keep agent training efficient.**

V. CONCLUSION

Our MASAC-Distrib algorithm improves the operational performance of a swarm of assets in an air-to-surface complex mission. While it confirms the interest of Multi-Agent RL to optimise complex and collaborative decisions, many improvements need to be implemented in order to integrate these algorithms into operational systems. As part of the challenge we still need to face, we can mention the representativeness of the simulated environment and the robustness of the agents to improve generalisation on real-world wartime environments; anticipation of the decisions in order to improve trust with the teamed operator. Moreover, we need to integrate DRL in a weapon system architecture in a way that is compatible with safety, reliability and supervision requirements.

ACKNOWLEDGMENT

The authors would like to thank the engineers who were involved in this project for the quality of their contributions.

REFERENCES

[1] UK Ministry of Defence (2015), Future Character of Conflict, *https://www.gov.uk/government/publications/future-character-of-conflict*, visited on 1st June 2022

[2] Kim, H., Jordan, M., Sastry, S., & Ng, A. (2003). Autonomous helicopter flight via reinforcement learning. *Advances in neural information processing systems*, *16*.

[3] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[4] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Hassabis, D. (2017). Mastering the game of go without human knowledge. *nature*, *550*(7676), 354-359.

[5] Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning* (pp. 330-337)

[6] Bazzan, A. L. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Autonomous Agents and Multi-Agent Systems, 18(3), 342-375.

[7] Hüttenrauch, M., Adrian, S., & Neumann, G. (2019). Deep reinforcement learning for swarm systems. Journal of Machine Learning Research, 20(54), 1-31.

[8] Fan, J. R., Li, D. G., Li, R. P., & Wang, Y. (2020). Analysis on MAV/UAV cooperative combat based on complex network. *Defence Technology*, *16*(1), 150-157.

[9] Wang, X., Yadav, V., & Balakrishnan, S. N. (2007). Cooperative UAV formation flying with obstacle/collision avoidance. *IEEE Transactions on control systems technology*, *15*(4), 672-679.

[10] Chandler, P. R., Pachter, M., & Rasmussen, S. (2001, June). UAV cooperative control. In *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)* (Vol. 1, pp. 50-55). IEEE.

[11] Liang, X. U., Xuanhong, P. A. N., & Ming, W. U. (2018). Analysis on manned/unmanned aerial vehicle cooperative operation in antisubmarine warfare. *中国舰船研究*, *13*(6), 154-159.

[12] Hou, Y., Liang, X., He, L., & Zhang, J. (2019). Time-coordinated control for unmanned aerial vehicle swarm cooperative attack on ground-moving target. *IEEE Access*, *7*, 106931-106940.

[13] Zhen, Z., Xing, D., & Gao, C. (2018). Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm. *Aerospace Science and Technology*, *76*, 402-411.

[14] Duan, H., Zhao, J., Deng, Y., Shi, Y., & Ding, X. (2020). Dynamic discrete pigeon-inspired optimization for multi-UAV cooperative search-attack mission planning. *IEEE Transactions on Aerospace and Electronic Systems*, *57*(1), 706-720.

[15] Ziyang, Z. H. E. N., Ping, Z. H. U., Yixuan, X. U. E., & Yuxuan, J. I. (2019). Distributed intelligent self-organized mission planning of multi-UAV for dynamic targets cooperative search-attack. *Chinese Journal of Aeronautics*, *32*(12), 2706-2716.

[16] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., ... & Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, *12*(4), e0172395

[17] Konda, V. R., & Tsitsiklis, J. N. (2003). Onactor-critic algorithms. *SIAM journal on Control and Optimization*, *42*(4), 1143-1166.

[18] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3), 229-256.

[19] Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, *12*.

[20] Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*

[21] Wang, X., Chen, Y., & Zhu, W. (2021). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*

[22] Shao, K., Zhu, Y., & Zhao, D. (2018). Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *3*(1), 73-84.

Algorithmes de placement optimisé de drones pour la conception de réseaux de communication

Zacharie Alès
UMA
ENSTA Paris
Palaiseau, France
zacharie.ales@ensta-paris.fr

Sourour Elloumi
UMA
ENSTA Paris
Palaiseau, France
sourour.elloumi@ensta-paris.fr

Adèle Pass-Lanneau
CATOD
DGA

***Abstract*—L'établissement d'un réseau de télécommunications performant et robuste est un besoin opérationnel clé pour les forces armées en opérations. Dans les prochaines années, des drones haute altitude pourront être utilisés comme noeuds du réseau sur le théâtre. Une question est alors de décider de leur placement spatial. Dans ce travail, une approche de recherche opérationnelle est proposée afin d'obtenir des algorithmes efficaces de placement de drones.**

***Index Terms*—drones haute altitude, conception de réseau, recherche opérationnelle, programmation linéaire en nombres entiers, heuristiques.**

I. Introduction

Sur un théâtre d'opérations, les unités déployées au sol se déplacent et communiquent entre elles par des moyens de communication basés sur différentes technologies : UHF, VHF, 5G, etc. Dans un contexte de numérisation croissante du champ de bataille et afin d'augmenter les capacités de mise en réseau des unités sur le terrain, l'utilisation future de drones haute altitude (HAPS) est envisageable. Ces systèmes, positionnés dans la stratosphère, peuvent emporter des relais de communication de différents types. Ils sont ainsi un intermédiaire entre les moyens au sol et les moyens satellitaires.

Pour évaluer l'intérêt opérationnel de tels systèmes, il est nécessaire de savoir comment ils seraient déployés sur un théâtre, au profit des forces au sol. La question du déploiement porte à la fois sur la position géographique de ces vecteurs, et sur les relais de communication qu'ils devraient emporter.

L'optimisation du réseau formé par un ensemble de drones peut être représenté par un problème d'optimisation combinatoire, proche de problèmes classiques en recherche opérationnelle. Dans le cadre du projet AID "LocHAPS" conjoint entre l'ENSTA Paris et la DGA, nous investiguons de nouvelles techniques de recherche opérationnelle, et notamment de programmation mathématique, afin de résoudre efficacement ce problème d'optimisation de réseau.

Les problématiques de placement de drones ont récemment attiré l'attention de la communauté académique, pour des applications duales et diversifiées (gestion de crise, déploiement de senseurs, ou déploiement d'un réseau de communication, voir [1] pour une revue de littérature). Les outils développés relèvent souvent de l'algorithmique pour les systèmes embarqués, ou visent à obtenir un réseau auto-organisé de drones [2]. Dans les travaux présentés, nous nous posons à l'inverse la question du point de vue d'un optimisateur centralisé, et nous investiguons des outils algorithmiques de résolution exacte. Le but est de répondre à une question de dimensionnement à un niveau stratégique : de quels systèmes aura-t-on besoin, et en quelle quantité, pour répondre au besoin opérationnel ?

Travail effectué dans le cadre du projet AID : Méthodes de Recherche Opérationnelle pour la conception de réseaux optimaux et robustes, 2021-2024

II. Formalisation du problème d'optimisation

Présentons plus en détail le problème considéré.

Eléments sur le théâtre. Nous considérons trois types d'éléments sur le théâtre : les unités, éventuellement une base et des drones déployés. Les unités sont des unités terrestres déployées sur le théâtre. On considère que l'on connaît leur position en fonction du temps. La base est une base avancée positionnée en un point du théâtre. Les drones sont déployés à une altitude fixée, en des points dont les coordonnées sont à optimiser. Ils ont une capacité d'emport déterminée par une charge utile et une puissance électrique maximale.

Relais de communication. On considère un ensemble de types de communications. Les unités et les drones sont munis de moyens de communications associés à ces différents types. Pour chaque unité, on dispose d'un rayon associé à chaque type de communication. Pour la base, on dispose similairement d'un rayon par type de communication. Pour ce qui est des drones, on dispose d'un ensemble de relais de communications, caractérisés par un rayon, un poids, et une puissance électrique. Un ensemble de relais forme un emport possible pour un drone si le poids total et la puissance totale n'excèdent pas la charge utile et la puissance maximale du drone, respectivement. Un drone muni de relais est ainsi doté de capacités de communication, représentées par les rayons de ses relais pour les différents types de communications.

Réseau recherché. Nous représentons la mise en réseau des différents éléments de la façon suivante. On considère des communications entre paires d'éléments drone-drone, drone-unité, drone-base. On dit qu'une paire d'éléments communique s'il existe au moins un type de communications pour lequel la distance entre les deux éléments est inférieure au rayon de communication des deux éléments. L'ensemble des éléments

qui communiquent entre eux forme ainsi un réseau connexe entre des drones, des unités et la base. Pour un réseau connectant uniquement un sous-ensemble d'unités, on dira que ces unités sont les unités *couvertes* par le réseau de drones.

Problème d'optimisation. Le problème d'optimisation considéré consiste à optimiser de façon jointe:

- les positions spatiales d'un nombre fixé p de drones ;
- les relais de communication qu'ils emportent ;

afin de former un réseau connexe entre la base, les p drones, et permettant de couvrir un sous-ensemble d'unités de taille maximale.

Ce problème comporte deux sous-problèmes distincts : le choix des relais emportés par les drones s'apparente à un problème de bin packing [3], tandis que le choix des positions des drones est un problème de couverture-connexité. Dans cet article, nous nous intéressons à ce second aspect.

III. Problème de couverture-connexité

A. Définition

Nous étudions un cas particulier du problème précédent dans lequel on ne considère qu'un seul type de communication. Dans ce cas, chaque drone est doté d'un relais de communication et l'emport des drones n'est plus à optimiser. Seules les positions des drones sont à déterminer.

Plus précisément, le problème (Loc) que nous considérons est le suivant : *connaissant les coordonnées des unités et un ensemble de positions possibles pour une flotte de p drones identiques, connaissant aussi les rayons de couverture et de communication des drones, trouver les positions des drones qui permettent aux p drones de communiquer entre eux tout en couvrant un nombre maximum d'unités.*

Un exemple d'instance et une solution au problème (Loc) sont représentés dans la Figure 1. L'instance comporte 12 unités sur un terrain plat, représentées par les carrés. Les positions possibles pour les drones sont obtenues par discrétisation du plan de placement des drones : on considère comme positions possibles les points d'une grille de pas unitaire. Dans la solution, les $p = 4$ drones sont indiqués par les disques bleus. Le rayon de couverture des drones est égal à 1, le rayon de communication entre drones est égal à 2.5. Les liens de communications entre drones sont indiqués. La solution représentée permet de couvrir 8 unités sur 12.

Le problème (Loc) permet d'isoler la difficulté combinatoire principale du problème global, celle de la couverture-connexité. Des algorithmes développés pour (Loc) pourront ensuite être utilisés pour la résolution du problème global.

B. Littérature associée en Recherche Opérationnelle

Les problèmes de localisation et de couverture sont largement étudiés dans la littérature et ont de nombreuses variantes. Plusieurs problèmes classiques de Recherche Opérationnelle peuvent être vus comme des cas particuliers du problème (Loc).

Le problème de localisation simple [4], [5] consiste à ouvrir des dépôts d'une centrale d'achat parmi un ensemble d'emplacements possibles, puis de servir un ensemble de clients par le dépôt ouvert le plus proche de façon à minimiser la somme des coûts d'ouverture des dépôts et de service des clients. Une variante également très étudiée fixe le nombre de dépôts à ouvrir à un nombre k et s'appelle le problème des k-médians.

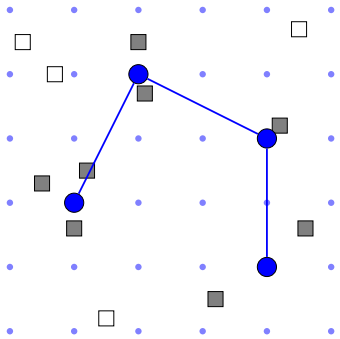


Fig. 1. Exemple d'instance et de solution au problème (Loc), représentées en vue de dessus. Les 4 drones sont représentés par les disques, les unités couvertes (resp. non couvertes) par les carrés gris (resp. blancs).

Concernant la connexité, un problème classique est celui de l'arbre de Steiner [6]. Il se pose dans un graphe non orienté avec des poids positifs sur les arêtes. On se donne aussi un sous-ensemble de sommets, généralement appelés terminaux. Le problème de l'arbre de Steiner consiste à sélectionner des arêtes qui relient tous les terminaux (en passant éventuellement par des sommets supplémentaires) et dont la somme des poids est minimale. Les arêtes d'une solution optimale forment donc un graphe sans cycle, soit un arbre.

Le problème de localisation connexe [7] associe ces deux derniers problèmes et a des similitudes avec le problème (Loc) puisqu'en plus d'ouvrir des dépôts, il faut qu'ils forment entre eux un graphe connexe, par exemple par des lignes de chemin de fer, de façon à ce qu'il soit toujours possible d'acheminer la marchandise d'un dépôt à un autre par le train.

Chacun de ces problèmes fait partie de la classe des problèmes NP-durs [3]. Ils trouvent des applications dans les télécommunications [7], dans la conception de réseaux de transport [8], ou dans le placement de capteurs [9]. Ils sont fréquemment traités dans la littérature par des techniques de Programmation Linéaire en Nombres Entiers, qui peuvent donner des résultats rapides même sur des instances très grandes, voir par exemple [5].

La problématique de placement de drones se distingue de la littérature par la nature a priori continue du placement des drones. En effet ceux-ci peuvent occuper n'importe quelle position dans un plan d'altitude fixée. On peut se ramener à un ensemble fini de positions par une discrétisation. Une telle discrétisation, si elle est fine, donne alors une instance du problème (Loc) avec un grand nombre de positions possibles.

IV. Formulations PLNE

A. Généralités

Afin de résoudre le problème (Loc), nous proposons deux formulations différentes représentées par des Programmes

Linéaires en Nombres Entiers (PLNE). Disposer de différentes formulations permet d'évaluer leurs performances numériques, et ainsi d'identifier la ou les formulations qui se comportent le mieux selon les instances à résoudre.

Une formulation PLNE peut être utilisée pour résoudre le problème par Branch & Bound, via des solveurs spécialisés (e.g., CPLEX). Cela nécessite cependant que la formulation soit de taille raisonnable, i.e., polynomiale en nombre de variables et contraintes. A l'inverse, les formulations de grande taille, par exemple avec un très grand nombre de contraintes, ne peuvent pas être directement traitées par les solveurs. Leur résolution se fait par un schéma algorithmique itératif, qui ajoute les contraintes parcimonieusement au fur et à mesure de la résolution grâce à une *séparation.* Ce schéma sera détaillé dans la suite, en Section IV-C.

Introduisons quelques notations pour le problème (Loc). On note $\mathcal{U}$ l'ensemble d'unités qui sont indexées par u allant de 1 à $|\mathcal{U}|$ et $\mathcal{P}$ l'ensemble de positions possibles des drones, indexées indifféremment par i ou j allant de 1 à $|\mathcal{P}|$. On connaît les coordonnées de chaque unité et de chaque position. On peut donc calculer la distance d entre n'importe quelle paire de points. Le nombre maximal de drones disponibles est p. Les drones sont tous identiques et ont un rayon de couverture $Rcouv$ (distance au-dessus de laquelle un drone ne peut pas communiquer avec une unité) et un rayon de communication $Rcom$ (distance au-dessus de laquelle un drone ne peut pas communiquer directement avec un autre drone).

Pour assurer la connexité entre les drones, on peut représenter la communication entre les drones par un graphe non orienté. Les sommets sont les positions où des drones peuvent être placés et une arête représente la possibilité de communication d'un drone vers un autre. On sait que le sous-graphe des positions occupées par des drones est connexe si et seulement s'il contient un arbre couvrant, soit une sélection connexe d'arêtes, avec autant d'arêtes que le nombre de sommets moins un.

Plusieurs définitions existent pour la notion d'arbre couvrant. L'une d'elles stipule que l'arbre couvrant, par une orientation de ses arêtes, forme une arborescence couvrante. Une telle orientation est équivalente à l'existence d'une numérotation des sommets où, pour tout arc (i, j) de l'arborescence, le numéro de i est strictement plus petit que le numéro de j. Cette numérotation est appelée un tri topologique. C'est cette définition de la notion d'arbre couvrant qui est utilisée dans la formulation MTZ, proposée Section IV-B. Une définition équivalente repose sur le fait qu'un arbre couvrant est exactement un sous-ensemble d'arêtes ne contenant pas de cycle et de taille égale au nombre de sommets couverts moins un. Cette définition est utilisée par la formulations sous-tours généralisés présentée Section IV-C.

B. Formulation MTZ

Ce type de formulation a été introduit dans [10] pour le problème du voyageur du commerce puis a été étendu à divers problèmes de connectivité. Les variables de décision sont les suivantes :

- y_j variable binaire qui vaut 1 si et seulement si un drone est placé à la position $j \in \mathcal{P}$;
- $x_{ij},\ i \neq j$ variable binaire qui vaut 1 si l'arc (i, j) est sélectionné pour faire partie de l'arborescence couvrante. Cela sous-entend que des drones sont placés en i et j et que la distance entre ces positions est inférieure au rayon de communication ;
- c_u variable binaire qui vaut 1 si et seulement si l'unité $u \in \mathcal{U}$ est couverte ;
- t_j numéro attribué à $j \in \mathcal{P}$ pour assurer l'existence d'un tri topologique.

La formulation (MTZ) pour le problème (Loc) s'écrit comme suit:

$$
(MTZ)\begin{cases}
\max \quad \displaystyle\sum_{u\in\mathcal{U}} c_u & \\
\text{s.c.} & \\
\quad c_u \leq \displaystyle\sum_{j:d_{ju}\leq Rcouv} y_j & u \in \mathcal{U} \quad (1)\\
\quad \displaystyle\sum_{j\in\mathcal{P}} y_j \leq p & (2)\\
\quad x_{ij} = 0 & i,j : d_{ij} > Rcom \quad (3)\\
\quad x_{ji} \leq y_j & j,i \in \mathcal{P} \quad (4)\\
\quad \displaystyle\sum_{i\in\mathcal{P}} x_{ij} \leq y_j & j \in \mathcal{P} \quad (5)\\
\quad \displaystyle\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} x_{ij} = \sum_{j\in\mathcal{P}} y_j - 1 & (6)\\
\quad t_j \geq t_i + 1 - p(1 - x_{ij}) & j,i \in \mathcal{P} \quad (7)\\
\quad y_j \in \{0,1\}\ c_u \in \{0,1\} & \\
\quad x_{ij} \in \{0,1\}\ t_j \geq 0 &
\end{cases}
$$

L'objectif est de maximiser le nombre d'unités couvertes. Les contraintes (1) imposent que l'unité u soit identifiée comme non-couverte si aucun drone n'est placé aux positions qui peuvent la couvrir. La contrainte (2) assure qu'au plus p drones peuvent être déployés. Les contraintes (3) fixent à 0 les variables de communication x_{ij} entre des positions trop éloignées. Les contraintes (4) et (5) assurent qu'aucune communication n'est possible ni en sortie ni en entrée d'une position j non équipée de drone. Les contraintes (5) assurent de plus qu'au plus un arc entrant en j est sélectionné pour la position j, si elle est équipée d'un drone. La contrainte (6) impose que le nombre d'arcs dans l'arborescence couvrante soit égal au nombre de drones positionnés moins un. Enfin, les contraintes (7) assurent la cohérence du tri topologique. Si l'arc (i, j) est sélectionné alors $x_{ij} = 1$ et le numéro de j est au moins celui de i plus un. Sinon, on peut montrer que l'inégalité est toujours vérifiée puisqu'on n'a pas besoin d'attribuer plus de p numéros différents. La Figure 2 illustre les valeurs possibles de la variable t pour une configuration avec quatre drones positionnés.

C. Formulation sous-tours généralisés

Cette formulation est inspirée de la formulation de [11] pour le problème du voyageur de commerce. Dans cette formulation, on définit un arbre couvrant comme un choix d'arêtes ne

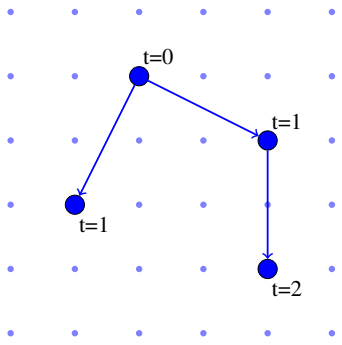


Fig. 2. Illustration de valeurs possibles des variables t, correspondant à des numéros selon un tri topologique.

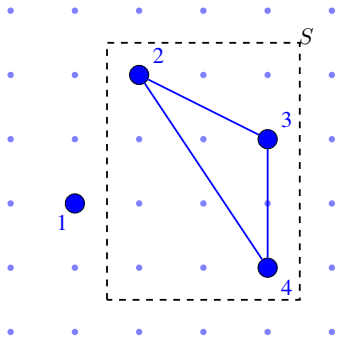


Fig. 3. Illustration de l'élimination des sous-tours. Les Contraintes (12) pour l'ensemble S ci-dessus et la position 2, imposent qu'au plus 2 arêtes soient sélectionnées à l'intérieur de S. Elles permettent ainsi d'éliminer le cycle constaté dans la solution ci-dessus.

comportant pas de cycle. Nous reprenons les mêmes variables y et c que ci-dessus. Pour les variables représentant les liens, on n'a pas besoin ici de les orienter. On considère donc une nouvelle variable binaire $z_{ij},\ i < j$ valant 1 si et seulement si l'arête (i, j) est sélectionnée dans l'arbre couvrant.

La formulation (ST) pour le problème (Loc) est la suivante:

$$
(ST)\begin{cases}
\max \quad \sum_{u\in\mathcal{U}} c_u & \\
\text{s.c.} & \\
\quad (1)-(2) & \\
\quad z_{ij} = 0 & i<j\in\mathcal{P} : d_{ij} > Rcom \quad (8)\\
\quad z_{ij} \le y_i & i<j\in\mathcal{P} \quad (9)\\
\quad z_{ij} \le y_j & i<j\in\mathcal{P} \quad (10)\\
\quad \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}:i<j} z_{ij} = \sum_{j\in\mathcal{P}} y_j - 1 & \quad (11)\\
\quad \sum_{i\in S}\sum_{j\in S:i<j} z_{ij} \le \sum_{j\in S} y_j - y_{j_0} & \forall S\subset\mathcal{P}:\ j_0\in S \quad (12)\\
\quad y_j\in\{0,1\}\ c_u\in\{0,1\} & \\
\quad z_{ij}\in\{0,1\} &
\end{cases}
$$

Les contraintes (8)-(11) ont le même effet que les contraintes (3)-(6) de la formulation (MTZ). Les contraintes (12) interdisent à un ensemble de positions S de contenir un cycle, comme illustré dans la Figure 3.

Séparation des contraintes de sous-tours. La formulation (ST) utilise moins de variables que la formulation (MTZ) mais a un nombre exponentiel de contraintes (12). Pour contourner cette difficulté et résoudre néanmoins cette formulation, on les ajoute au fur et à mesure que l'on constate leur violation par une solution $(\tilde{y}, \tilde{z}, \tilde{c})$. Une telle procédure est appelée *séparation*. Le schéma algorithmique pour résoudre (ST) est alors le suivant :

- résoudre la formulation (ST) avec seulement un sous-ensemble $\mathcal{C}$ de contraintes (12), en déduire une solution $(\tilde{y}, \tilde{z}, \tilde{c})$;
- appliquer la séparation :
  - si aucune contrainte (12) n'est violée, alors $(\tilde{y}, \tilde{z}, \tilde{c})$ est une solution optimale de (ST) ;
  - sinon, ajouter la contrainte (12) violée à $\mathcal{C}$;
- itérer.

Pour la séparation, il faut trouver un ensemble S et une position $j_0 \in S$ qui maximisent la violation de la contrainte associée (12). Plus formellement, ce problème peut être modélisé par le problème (SEP) suivant :

$$
(SEP)\begin{cases}
\max\ \Delta = \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}:i<j} \tilde{z}_{ij}a_i a_j - \sum_{j\in\mathcal{P}} \tilde{y}_j a_j + \sum_{j\in\mathcal{P}} \tilde{y}_j h_j & \\
\text{s.c.} & \\
\quad h_j \le a_j \qquad j\in\mathcal{P} & \quad (13)\\
\quad \sum_{j\in\mathcal{P}} h_j = 1 & \quad (14)\\
\quad a_j, h_j \in \{0,1\} &
\end{cases}
$$

où la variable binaire a_i vaut 1 si et seulement si la position i appartient à l'ensemble recherché S et la variable binaire h_j vaut 1 si et seulement si $j \in S$ et joue le rôle du j_0 recherché. Les contraintes (13) forcent h_j à valloir 0 si j n'est pas dans S. Les contraintes (14) assurent qu'une seule position j_0 sera sélectionnée. La fonction objectif mesure la violation maximale du point $(\tilde{y}, \tilde{z}, \tilde{c})$ par les contraintes (12). Ainsi, si on résout (SEP) et que l'on trouve, dans une solution optimale, $\Delta \le 0$, cela signifie qu'aucune contrainte (12) n'est violée par ce point et le problème (ST) est résolu. Sinon, la solution (a, h) nous permet d'identifier un couple (S, j_0) correspondant à une contrainte (12) violée par le point. On ajoute cette contrainte et on continue le processus de résolution. Cet ajout dynamique d'inégalités violées peut être effectué à chaque nœud de la méthode and Branch & Bound et peut être implémenté avec un solveur comme CPLEX à l'aide des Callbacks.

V. Résolution heuristique

Lorsque la taille du problème augmente, la résolution exacte des modèles PLNE présentés en section précédente peut nécessiter un temps de calcul prohibitif. Afin d'obtenir des solutions en temps limité pour ces instances de grande taille, nous introduisons dans cette section deux algorithmes gloutons. Nous en présentons, tout d'abord, une version déterministe puis, dans un second temps, une version randomisée. Nous verrons en Section VI que ces dernières permettent d'obtenir des solutions fournissant de bonnes performances en des temps de calcul très raisonnables.

A. Algorithmes déterministes

Le fonctionnement des algorithmes déterministes est illustré par l'Algorithme 1. Soit $\mathcal{P}$ l'ensemble des positions possibles. On considère un sous-ensemble S de positions où sont placés des drones, initialement vide. A chaque itération, l'ensemble $\mathcal{P}_S \subseteq \mathcal{P} \backslash S$ des positions à portée d'au moins un des drones de S est identifié. Un drone est ensuite déployé dans une des positions de $\mathcal{P}_S$ maximisant une fonction d'évaluation f. L'algorithme s'arrête lorsque S contient p positions, autrement dit p drones sont placés à ces positions, ou lorsqu'il n'existe plus aucune position à portée des drones de S.

Entrées :
- I : une instance du problème
- $f : \mathcal{P} \to \mathbb{R}^+$: fonction d'évaluation des unités

Sorties : Un ensemble S de positions formant un réseau connexe

1 $S \leftarrow \emptyset$
2 $\mathcal{P}_S \leftarrow \mathcal{P}$
3 **tant que** $|S| < p$ *et qu'il est possible d'ouvrir un site* $(\mathcal{P}_S \neq \varnothing)$ **faire**
4 $\quad s \leftarrow argmax_{s \in \mathcal{P}_S} f(s)$
5 $\quad S \leftarrow S \cup \{s\}$
6 $\quad \mathcal{P}_S \leftarrow$ Sites de $\mathcal{P} \backslash S$ à portée d'un site de S
7 **retourner** S

Algorithme 1 : Pseudo-code des heuristiques déterministes A_U et A_{PL}.

Nos algorithmes, nommés A_{PL} et A_U, se distinguent par leurs fonctions d'évaluation nommées f_{PL} et f_U, respectivement. L'algorithme A_{PL} se base sur la relaxation linéaire du programme (MTZ) défini dans la section précédente. La relaxation linéaire d'un PLNE correspond au problème obtenu en relâchant les contraintes d'intégrité (ex : $x \in \{0, 1\}$ deviendrait $x \in [0, 1]$). L'avantage de ce problème relaché est que sa résolution est beaucoup plus rapide, puisqu'il s'agit d'un Programme Linéaire (PL). Son inconvénient est qu'il fournit une solution généralement non réalisable car fractionnaire. L'information fournie par une solution fractionnaire peut néanmoins être exploitée et nous en tirons partie dans l'algorithme A_{PL}. A chaque itération d'A_{PL}, on calcule la relaxation linéaire (c^*, t^*, x^*, y^*) de notre problème initial auquel ont été ajoutées des contraintes imposant l'ouverture des sites de S. La fonction d'évaluation est définie de la manière suivante :

$$f_{PL}(j) = y_j^*. \tag{15}$$

Ainsi, à chaque itération, A_{PL} ouvre un des sites $j \in \mathcal{P}_S$ pour lequel la valeur fractionnaire de la variable y_j^* est maximale.

Le second algorithme A_U ouvre à chaque itération un site permettant de maximiser le nombre de nouvelles unités couvertes :

$$f_U(j) = \text{nombre d'unités couvertes par } S \cup \{j\} - \text{nombre d'unités couvertes par } S. \tag{16}$$

Ces deux algorithmes sont dits déterministes puisque pour une instance donnée, ils fournissent toujours la même solution (sous réserve que la relaxation linéaire soit résolue de manière déterministe pour A_{PL}). Nous verrons que leurs temps de calcul sont très rapides en pratique mais que la qualité de leurs solutions n'est pas toujours satisfaisante.

B. Algorithmes randomisés

Afin de permettre d'améliorer les solutions au cours du temps, nous considérons également des versions randomisées A_{PL}^r et A_U^r de ces algorithmes représentées par l'Algorithme 2. Cet algorithme applique itérativement une version modifiée de l'Algorithme 1 (Instructions 4 à 11). La modification intervient à l'Instruction 7 qui au lieu de choisir le site maximisant f, effectue un tirage aléatoire pondéré par f. Cette étape permet d'apporter de la diversité dans les solutions en ne choisissant pas nécessairement à chaque étape le site le plus prometteur. Si la solution S obtenue est la meilleure trouvée jusqu'ici, S_{best} est mis à jour (Instructions 9 et 10). Ce processus est répété tant que le nombre maximal d'itérations n'est pas atteint.

Entrées :
- I : une instance du problème
- $f : \mathcal{P} \to \mathbb{R}^+$: fonction d'évaluation des sites
- i_{max} : nombre maximal d'itérations

Sorties : Un ensemble S_{best} de sites formant un réseau connexe

1 $S_{best} \leftarrow$ Solution de l'Algorithme 1
2 $i \leftarrow 1$
3 **tant que** $i \leq i_{max}$ **faire**
4 $\quad S \leftarrow \emptyset$
5 $\quad \mathcal{P}_S \leftarrow \mathcal{P}$
6 $\quad$ **tant que** $|S| < p$ *et qu'il est possible d'ouvrir un site* **faire**
7 $\quad\quad s \leftarrow$ tirage aléatoire dans $\mathcal{P}_S$ pondéré par f
8 $\quad\quad S \leftarrow S \cup \{s\}$
9 $\quad\quad \mathcal{P}_S \leftarrow$ Sites de $\mathcal{P} \backslash S$ à portée d'un site de S
10 $\quad$ **si** S *couvre plus de clients que* S_{best} **alors**
11 $\quad\quad S_{best} \leftarrow S$
12 $\quad i \leftarrow i + 1$
13 **retourner** S_{best}

Algorithme 2 : Pseudo-code des heuristiques randomisées A_U^r et A_{PL}^r.

VI. Résultats numériques

A. Résolution exacte

Nous étudions les performances numériques des algorithmes proposés, et l'impact sur celles-ci du pas η de la grille utilisée pour discrétiser les positions des drones. Pour ces expériences nous utilisons un Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz équipé de 15GByte de mémoire RAM.

Jeux de données. On considère un terrain carré de côté $C = 6000$. Les abscisses et ordonnées des positions et

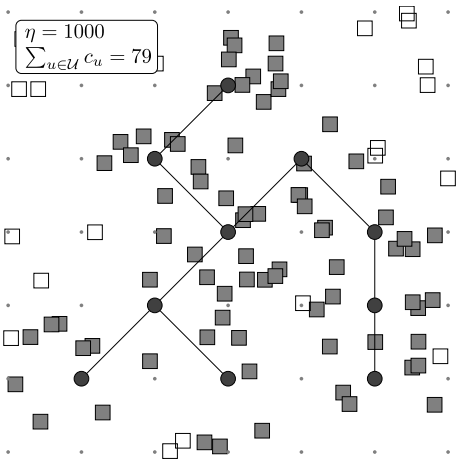


Fig. 4. Exemple de solution de l'instance $n = 100$, $p = 10$ et $\eta = 1000$, représentée en vue de dessus. Les drones sont représentés par les disques, les unités couvertes (resp. non couvertes) par les carrés gris (resp. blancs).

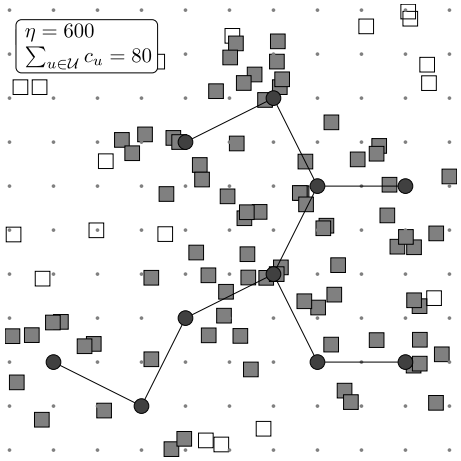


Fig. 5. Exemple de solution de l'instance $n = 100$, $p = 10$ et $\eta = 600$, représentée en vue de dessus. Les drones sont représentés par les disques, les unités couvertes (resp. non couvertes) par les carrés gris (resp. blancs).

des unités sont aléatoires dans l'intervalle $[0, C]$. Le nombre d'unités est $n = 100$ ou $n = 1000$. On dispose d'un modèle d'altimétrie du terrain, qui fournit l'altitude de chaque unité. On considère $p = 10$ drones, $Rcom = Rcouv = 1500$, placés à une altitude $Alt = 1200$. Le pas η de la grille de positions des drones est choisi comme un diviseur de C, au plus égal à $Rcouv$. Nous avons choisi $\eta = 1500, 1200, 1000, 750, 600, 500, 400, 375, 300, 200$ ce qui donne un nombre de positions m respectivement égal à $25, 36, 49, 81, 121, 169, 256, 289, 441$ et 961.

Solutions obtenues. La résolution de la formulation (MTZ) permet d'obtenir des solutions optimisées au problème (Loc), dont certaines sont illustrées dans les Figures 4,5,6. On observe dans ces trois figures qu'une réduction du pas η permet une augmentation du nombre d'unités couvertes $\sum_{u\in\mathcal{U}} c_u$.

En pratique, il peut être souhaitable d'imposer la couverture d'un sous-ensemble d'unités prioritaires $\mathcal{U}_p \subset \mathcal{U}$ (e.g., les unités à l'avant du dispositif). Ceci peut aisément être intégré dans notre modélisation par l'ajout des contraintes :

$$c_u = 1 \quad \forall u \in \mathcal{U}_p \tag{17}$$

Dans les Figures 4,5,6 on peut par exemple observer que les cinq unités situées en haut à droite ne sont jamais couvertes. En imposant leur couverture, nous obtenons la solution représentée en Figure 7. La couverture des unités rouges se fait cependant au détriment du nombre total d'unités couvertes.

Performances de la formulation MTZ. Les résultats de la formulation MTZ sont donnés en Table I pour un nombre d'unité n égal à 100 et 1000. Afin d'accélérer le temps de résolution, la solution obtenue par A_U est donnée en entrée de la formulation.

Le temps de calcul est donné en secondes dans la colonne (Temps (s)). Il ne peut pas excéder un temps limite TL que nous avons fixé à 1800 secondes, soit 30 minutes. La colonne intitulée "gap" contient l'écart relatif entre la meilleure solution réalisable trouvée et la borne supérieure obtenue à l'issue

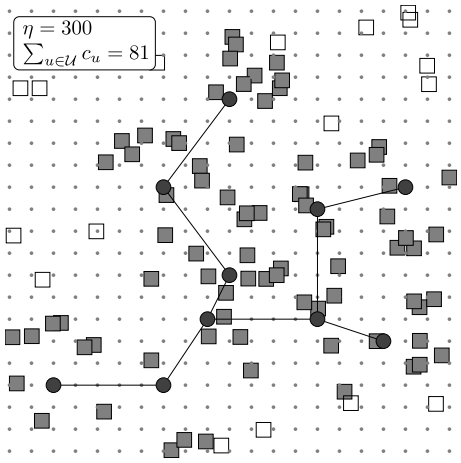


Fig. 6. Exemple de solution de l'instance $n = 100$, $p = 10$ et $\eta = 300$, représentée en vue de dessus. Les drones sont représentés par les disques, les unités couvertes (resp. non couvertes) par les carrés gris (resp. blancs).

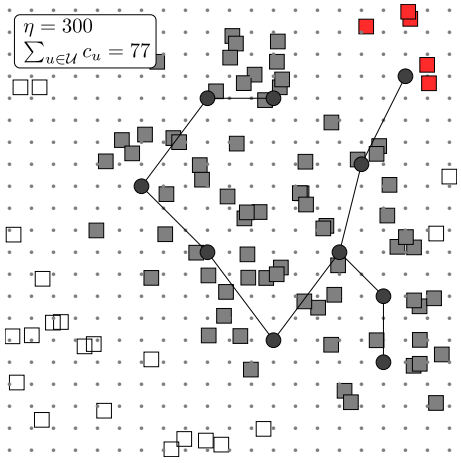


Fig. 7. Exemple de solution de l'instance $n = 100$, $p = 10$ et $\eta = 300$ en vue de dessus dans laquelle on impose que les unités représentées par des carrés rouges soient couvertes.

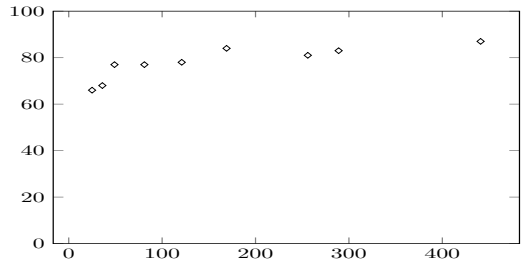


Fig. 8. Nombre optimal d'unités couvertes (%) selon le nombre de positions m.

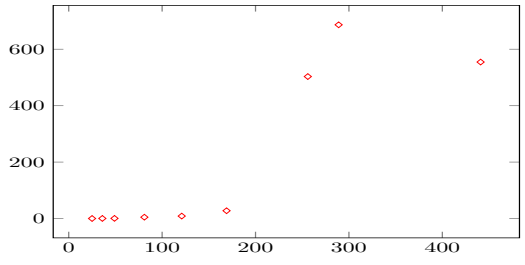


Fig. 9. Temps de calcul (secondes) selon le nombre de positions m.

de la résolution. Il est égal à 0 lorsque le problème est résolu optimalement en dessous du temps limite TL. Lorsque TL est atteint, ce gap permet de juger de la qualité de la solution trouvée à la fin de l'algorithme Branch & Bound. On observe que les instances ne sont pas résolues optimalement dès que l'on atteint 400 positions pour $n = 100$ et 600 positions pour $n = 1000$.

A priori, le positionnement réel des drones n'est pas contraint à des positions discrètes dans le plan d'altitude Alt. On recherche donc un pas η le plus petit possible, pour obtenir une discrétisation la plus fine possible. Afin d'estimer une taille de grille raisonnable, on peut s'intéresser à la valeur optimale du problème, c'est-à-dire le nombre maximal d'unités couvertes. Les Figures 8 et 9 montrent que le nombre maximal d'unités couvertes croît très lentement avec l'augmentation du nombre de positions au delà de 200 positions, tandis que le temps de calcul explose. Cela indique qu'une discrétisation comportant $m = 256$ positions correspondant à une grille 16x16 – qui pouvait a priori paraître grossière – peut en pratique offrir un bon compromis entre la valeur optimale et le temps de calcul.

Performances de la formulation (ST). Nous ne rapportons pas ici le détail des performances de la formulation (ST) car elles sont bien en-deça de celles de la formulation (MTZ), comme on peut le voir sur l'exemple en Figure 10. Cette conclusion numérique est cohérente avec les résultats de la littérature [7]. Un travail algorithmique approfondi sur la séparation est nécessaire pour obtenir des temps de calcul probants pour (ST).

B. Résolution heuristique

La Table I permet également de comparer les résultats obtenus par la formulation (MTZ) et les heuristiques présentées en Section V. Le nombre d'itérations des algo-

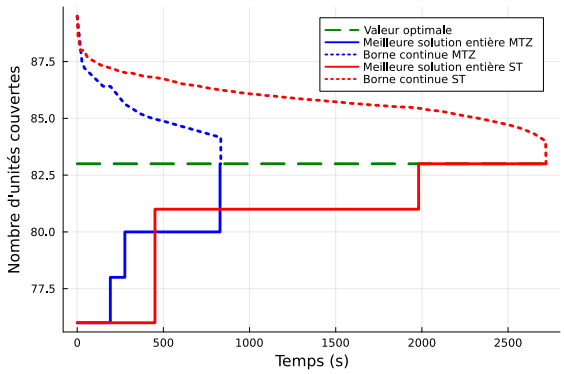


Fig. 10. Evolution des bornes inférieures et supérieures des formulations ST et MTZ au cours de la résolution pour l'instance $n = 100$, $p = 10$ et $\eta = 500$.

rithmes randomisés est fixé à 20. On constate tout d'abord que les heuristiques basées sur la relaxation continues A_{PL} et A^r_{PL} fournissent dans la plupart des cas des solutions de moins bonne qualité et des temps de calcul bien plus élevés que les heuristiques basées sur le nombre de clients couverts A_U et A^r_U. Ces temps de calcul peuvent s'expliquer par le fait que la taille de la formulation augmente rapidement lorsqu'η diminue rendant ainsi la résolution de la relaxation linéaire prohibitive. On peut également observer que les heuristiques randomisées permettent dans de nombreux cas d'augmenter le nombre d'unités couvertes. Ceci est d'autant plus intéressant pour A^r_U dont les temps de calcul restent très faibles.

Les heuristiques fournissent des solutions de bonne qualité mais rarement des solutions optimales. Il semble donc plus pertinent d'utiliser (MTZ) lorsque la taille du problème permet une résolution exacte. En revanche, pour les instances les plus grandes, A^r_U fournit les meilleurs résultats.

C. Compromis entre le nombre de drones et la capacité de couverture offerte

La résolution du problème (Loc) doit également permettre d'éclairer des choix sur le compromis à rechercher entre le coût du réseau de drones employé, et la capacité de couverture offerte par ce réseau.

On peut obtenir, pour différentes valeurs du nombre de drones p, le nombre maximal d'unités couvertes, et ainsi représenter le front de Pareto comme en Figure 11. On voit ainsi que la couverture de 80% des unités peut être assurée avec 9 drones, contre 15 drones pour couvrir 100% des unités.

D. Visualisation interactive de solutions

La résolution efficace du problème (Loc), soit par une méthode exacte soit par les heuristiques, fournit un placement de drones optimisé. Il est utile en pratique de visualiser les solutions obtenues pour les critiquer et éventuellement apporter des modifications à la modélisation du problème. Un exemple de visualisation de solution est donné Figure 12.

n	η	m	Nombre d'unités couvertes (MTZ)	A_{PL}	A^r_{PL}	A_U	A^r_U	Gap final (%) (MTZ)	Temps (s) (MTZ)	A_{PL}	A^r_{PL}	A_U	A^r_U
100	1500	25	**72**	**72**	**72**	**72**	**72**	**0.0**	0	0	0	0	0
	1200	36	**70**	66	69	**70**	**70**	**0.0**	1	0	3	0	0
	1000	49	**79**	24	71	71	73	**0.0**	1	0	4	0	0
	750	81	**79**	18	70	72	74	**0.0**	6	1	18	0	0
	600	121	**80**	56	76	77	77	**0.0**	100	1	27	0	0
	500	169	**83**	14	62	76	81	**0.0**	772	3	96	0	1
	400	256	**85**	14	66	83	83	**2.7**	TL	25	525	0	0
	375	289	**83**	14	62	79	81	**5.6**	TL	19	774	0	1
	300	441	**82**	13	60	81	81	**10.2**	TL	123	TL	0	1
	200	961	79	45	45	79	**80**	**17.4**	TL	1072	TL	1	3
1000	1500	25	**607**	**607**	**607**	604	**607**	**0.0**	0	0	2	0	2
	1200	36	**604**	568	577	602	**604**	**0.0**	1	0	4	0	2
	1000	49	**671**	361	635	665	665	**0.0**	2	0	11	0	1
	750	81	**692**	292	537	690	**692**	**0.0**	17	1	28	0	2
	600	121	634	347	428	634	**647**	**10.2**	TL	3	93	0	2
	500	169	689	463	589	689	**699**	**5.1**	TL	5	269	0	3
	400	256	665	267	489	665	**685**	**11.5**	TL	21	1042	0	3
	375	289	683	424	434	683	**690**	**9.0**	TL	50	1327	0	2
	300	441	676	386	437	676	**721**	**12.9**	TL	159	TL	0	4
	200	961	686	248	249	686	**703**	**13.8**	TL	TL	TL	0	7

TABLE I
COMPARAISON DES RÉSULTATS DES DIFFÉRENTES MÉTHODES DE RÉSOLUTION. $TL = 1800$ SECONDES.

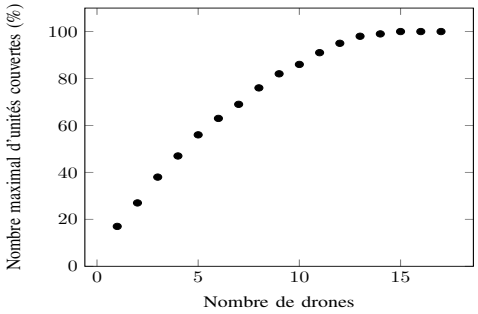


Fig. 11. Front de pareto entre le nombre de drones déployés p et le nombre d'unités couvertes (%), pour $m = 256$ positions.

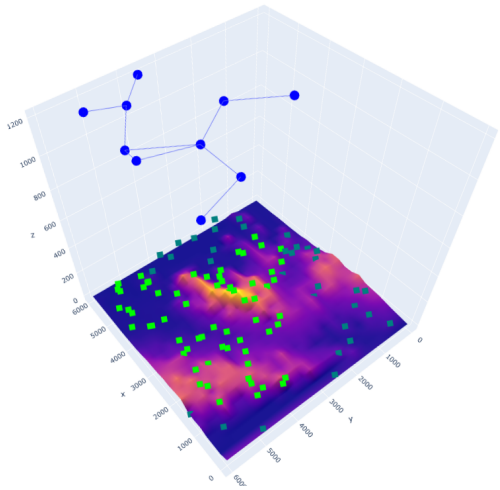


Fig. 12. Vue en 3D d'une solution obtenue, comprenant le modèle terrain, les unités couvertes en vert, et le réseau de drones en bleu.

VII. EXTENSIONS

De nombreuses extensions du problème (Loc) peuvent être considérées afin de mieux représenter la réalité du terrain. L'intérêt des modélisations PLNE est d'être très souples à l'ajout de nouvelles contraintes ou critères. Parmi ces extensions, peuvent être prises en compte :

– les communications multi-technologies, les contraintes de charge utile et de puissance max des drones comme présentées Section II. Cette généralisation a été étudiée précédemment [12], et les algorithmes proposés pour (Loc) s'étendent naturellement.

– l'efficacité énergétique du réseau, par l'introduction d'un critère d'optimisation secondaire pénalisant les distances de communication drone-drone importantes.

– la géométrie du réseau, afin d'améliorer sa résilience. On peut par exemple imposer que les unités soient à portée de deux drones pour être couvertes, ou bien contraindre les variables x définissant les communications entre drones.

VIII. CONCLUSION

Ces travaux ont permis d'identifier, au coeur de la problématique de placement de drones, un problème de couverture-connexité pour lequel les outils de Recherche Opérationnelle sont adaptés. Des approches algorithmiques exactes et heuristiques ont été proposées. Un point clé des performances des algorithmes est la discrétisation des positions possibles des drones. Les heuristiques proposées permettent d'obtenir des solutions très proches de l'optimum sur les instances considérées.

Une perspective est d'utiliser des techniques plus avancées de programmation mathématique pour résoudre, à l'exact, des instances de plus grande taille. L'extension des algorithmes présentés ici pour les cas plus généraux – multi-technologies, efficace en énergie, résilient – sera également poursuivie.

REFERENCES

[1] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[2] K. Danilchenko, M. Segal, and Z. Nutov, "Covering users by a connected swarm efficiently," in *Algorithms for Sensor Systems*, C. M. Pinotti, A. Navarra, and A. Bagchi, Eds. Cham: Springer International Publishing, 2020, pp. 32–44.

[3] B. H. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. New York, NY: Springer-Verlag, 2012.

[4] G. Laporte, S. Nickel, and F. S. da Gama, *Introduction to Location Science*. Cham: Springer International Publishing, 2015, pp. 1–18. [Online]. Available: https://doi.org/10.1007/978-3-319-13111-5_1

[5] J.-F. Cordeau, F. Furini, and I. Ljubić, "Benders decomposition for very large scale partial set covering and maximal covering location problems," *European Journal of Operational Research*, vol. 275, no. 3, pp. 882–896, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221718310737

[6] I. Ljubic, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti, "An algorithmic framework for the exact solution of the prize-collecting steiner tree problem," *Math. Program.*, vol. 105, pp. 427–449, 02 2006.

[7] S. Gollowitzer and I. Ljubić, "MIP models for connected facility location: A theoretical and computational study," *Computers & Operations Research*, vol. 38, no. 2, pp. 435–449, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054810001334

[8] V. Bucarey, B. Fortz, N. González-Blanco, M. Labbé, and J. A. Mesa, "Benders decomposition for network design covering problems," *Computers & Operations Research*, vol. 137, p. 105417, 2022.

[9] A. Murray, K. Kim, J. Davis, R. Machiraju, and R. Parent, "Coverage optimization to support security monitoring," *Computers, Environment and Urban Systems*, vol. 31, pp. 133–147, 03 2007.

[10] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

[11] G. Laporte, "Generalized subtour elimination constraints and connectivity constraints," *Journal of the Operational Research Society*, vol. 37, no. 5, pp. 509–514, 1986.

[12] Z. Alès, S. Elloumi, M. Y. Naghmouchi, A. Pass-Lanneau, and O. Thuillier, "Planification optimisée du déploiement d'un réseau de télécommunication multitechnologie par dispositifs aéroportés sur un théâtre d'opérations extérieures," in *23ème congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*. Villeurbanne - Lyon, France: INSA Lyon, Feb. 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03595379

Empreinte de réseaux avec des entrées authentiques

Thibault MAHO
Univ. Rennes, Inria, CNRS
IRISA, Rennes, France
thibault.maho@inria.fr

Teddy FURON
Univ. Rennes, Inria, CNRS
IRISA, Rennes, France
teddy.furon@inria.fr

Erwan LE MERRER
Univ. Rennes, Inria, CNRS
IRISA, Rennes, France
erwan.lemerrer@inria.fr

***Abstract*—Les avancées récentes dans le domaine des empreintes de réseaux profonds détectent des instances de modèles placées dans une boîte noire. Les entrées utilisées en tant qu'empreintes sont spécifiquement conçues pour chaque modèle à vérifier. Bien qu'efficace dans un tel scénario, il en résulte néanmoins un manque de garantie après une simple modification (*e.g.* réentraînement, quantification) d'un modèle.**

Cet article s'attaque aux défis de proposer i) des empreintes qui résistent aux modifications significatives des modèles, en généralisant la notion de familles de modèles et leurs variantes, ii) une extension de la tâche d'empreinte à des scénarios où l'on souhaite un modèle précis (précédemment appelé tâche de *detection*), mais aussi d'identifier la famille de modèles qui se trouve dans la boîte noire (tâche d'*identification*).

Nous atteignons ces deux objectifs en démontrant que des entrées authentiques (non modifiées) sont un matériau suffisant pour les deux tâches. Nous utilisons la théorie de l'information pour la tâche d'identification et un algorithme glouton pour la tâche de détection. Les deux approches sont validées expérimentalement sur un ensemble inédit de plus de 1 000 réseaux.

***Index Terms*—Empreinte, Réseaux profonds, Théorie de l'information**

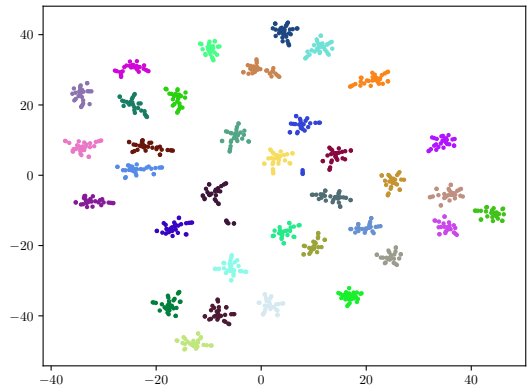


Fig. 1: Une représentation t-SNE des distances par paire de 1081 modèles différents: 10 types de variation appliqués sur 35 modèles vanilla prêts à l'emploi pour ImageNet avec différents paramètres. Ce travail exploite la séparabilité claire (groupes de couleurs) observée dans les décisions de ces modèles.

I. Introduction

L'empreinte est une signature identifiant de manière unique un modèle, comme les minuties de l'empreinte digitale en biométrie. Il s'agit essentiellement d'un problème en boîte noire: le classifieur à identifier se trouve dans une boîte noire dans le sens où l'on peut uniquement faire quelques requêtes et observer ses résultats. Par exemple, le modèle est intégré dans une puce ou nous pouvons l'interroger via une API (MLaaS).

La principale application visée par les travaux actuels [1]–[5] est la preuve de la propriété. Un réseau de neurones profonds précis est un actif industriel précieux en raison du savoir-faire nécessaire à son entrainement, de la difficulté de rassembler un ensemble de données bien annoté, et des ressources nécessaires à l'apprentissage de ses paramètres. Le coût de GPT-3, l'un des modèles NLP les plus grands et les plus précis est estimé à 4,6 millions de dollars[1]. Dans ce contexte, l'entité qui identifie une boîte noire veut *detecter* s'il ne s'agit pas d'un de ses modèles volés.

Une autre application critique est le gain d'informations. Par exemple, un attaquant désireux de tromper le classifieur gagne d'abord des connaissances sur le modèle distant. Une entreprise peut également déterminer quel modèle est utilisé dans le système de production d'un concurrent. Cette application a été laissée de côté jusqu'à présent, et nous l'abordons sous la notion d'*identification*.

Pour plus de clarté, nous nommons Alice l'entité voulant identifier le modèle que Bob a intégré dans la boîte noire.

a) Challenges: Il existe de nombreuses façons de modifier un modèle tout en conservant sa précision. Ces procédures permettent de simplifier un réseau (quantification des poids, pruning, voir *e.g.* [6]), ou le robustifier (prétraitement de l'entrée, réentraînement [7]). Nous nommons un modèle modifié une *variante*. Ces mécanismes n'ont pas été conçus a priori pour rendre l'empreinte plus difficile, mais ils laissent la possibilité à Bob d'altérer l'empreinte d'un modèle. Nous supposons qu'Alice connaît également certaines de ces procédures. Or, elles sont souvent définies par de nombreux paramètres, et parmi eux des scalaires, qui peuvent donner une infinité de variantes. Comme en biométrie, l'empreinte doit être suffisamment discriminante pour être unique par modèle mais aussi suffisamment robuste pour identifier une variante.

Les approches existantes reposent sur deux piliers. Elles utilisent les frontières de décision du classifieur comme empreinte [2], [4], [5]. Deux réseaux avec la même architecture et les mêmes ensemble et procédure d'apprentissage sont différents car l'apprentissage est stochastique. Ainsi, leurs

[1]https://lambdalabs.com/blog/demystifying-gpt-3/

frontières dans l'espace d'entrée ne se chevauchent pas. La plupart des articles de la littérature recherchent des déviations discriminantes de ces frontières. Deuxièmement, la tâche clé est la détection: Alice fait une supposition sur le modèle dans la boîte noire et l'interroge ensuite avec des requêtes spécifiques pour valider son hypothèse [2], [4], [5], [8].

b) Justification: Nous constatons que l'utilisation d'entrées non modifiées n'a pas été étudiée en profondeur. Elle constitue un avantage certain, car elle supprime la nécessité de concevoir des moyens complexes pour les créer. Elle est moins sujette à l'utilisation de défenses du côté de Bob (*e.g.* rejet basé sur la distance à la frontière [9]). Les travaux précédents se restreignent à la tâche de détection. La possibilité plus générale d'identifier un modèle à l'intérieur de la boîte noire n'a pas été étudiée. Notre travail diffère donc des travaux précédents sur ces deux aspects: *i)* nous ne forgeons pas d'entrées spécifiques mais utilisons des entrées non modifiées (l'espace d'entrée n'est pas sondé pour découvrir les frontières de décision), et *ii)* nous identifions directement les modèles en utilisant leurs difficultés de classification sur une poignée d'entrées. Cette méthode est plus efficace que la détection séquentielle de modèles.

En résumé, quand Bob a choisi un modèle parmi un ensemble de réseaux connus par Alice, notre solution est essentiellement déterministe: Alice doit trouver la plus petite séquence d'entrées pour identifier la boîte noire. Nous appliquons un algorithme gourmand qui sélectionne soigneusement l'entrée pour réduire itérativement l'ensemble des suspects. La théorie de l'approximation indique que cette méthode est sous-optimale, mais nous constatons qu'en pratique, de nombreux réseaux sont identifiés en moins de trois requêtes.

Lorsqu'une attaque de Bob a créé une variante d'un modèle, la sortie de celui-ci peut ne pas correspondre à la sortie d'un modèle connu. Nous utilisons alors la théorie de l'information de Shannon pour mesurer la dépendance statistique entre les sorties des deux modèles. La Figure 1 est la représentation t-SNE à partir des distances par paire dans un ensemble de 35 modèles vanilla et de leurs variantes. Les familles de modèles sont bien regroupées dans le sens où les variantes sont plus proches de leur réseau d'origine que des autres modèles. Alice peut ne pas identifier précisément la variante du modèle mais elle peut identifier sa famille, déduire quel était le modèle vanilla d'origine et même quel type de variation a été appliqué.

c) Contributions: Notre contribution est quadruple. 1) Nous démontrons que la simple utilisation d'images non modifiées est suffisante pour atteindre des taux de réussite élevés pour l'empreinte des modèles. 2) La tâche de détection, introduite par l'état de l'art, est complétée par l'introduction de la tâche d'identification. Nous considérons cette dernière comme un problème de théorie de l'information. 3) Nous présentons une distance basée sur l'information mutuelle empirique, évaluant la proximité de deux modèles. Cette distance permet de généraliser la notion d'attaques sur les modèles à travers la notion de familles de modèles et de variantes. 4) Nous effectuons une expérimentation approfondie en considérant plus de 1 000 modèles de classification sur ImageNet. Elle apporte des améliorations significatives en termes de précision dans la tâche de détection.

La section II est une analyse de menace listant toutes les hypothèses de travail. La section III s'intéresse à la détection (Alice vérifie son hypothèse sur la boîte noire) puis l'identification (Alice découvre quel modèle se trouve dans la boîte noire). Elle contient des résultats expérimentaux. La section IV est consacrée aux travaux apparentés et le benchmark avec les détections de l'état de l'art.

II. Modèle de menace

Cette section détaille les objectifs d'Alice et de Bob.

A. Bob: garder son modèle anonyme

1) Objectifs: Bob joue en premier en choisissant secrètement un modèle et en le mettant dans la boîte noire. Ce modèle peut être un modèle classique ou une variante d'un modèle connu. Une variante est créée en appliquant sur un modèle vanilla m donné, la procédure V paramétrée par $\theta \in \Theta$ qui décrit le type de modification et les paramètres associés. Cela peut être considérer comme une attaque de Bob sur le modèle vanilla pour renforcer l'identification. Nous désignons une telle variante par $\mathsf{v} = \mathsf{V}(\mathsf{m}, \theta)$.

L'objectif de Bob est d'offrir un classifieur en boîte noire précis en conservant l'anonymat du modèle utilisé. Seule une petite perte de performance est tolérée. Pour la classification, les performances d'un modèle m sont évaluées par la précision du top-1, notée $\mathsf{acc}(\mathsf{m})$. Nous formalisons cette exigence par

$$\frac{\mathsf{acc}(\mathsf{m}) - \mathsf{acc}(\mathsf{V}(\mathsf{m}, \theta))}{\mathsf{acc}(\mathsf{m})} < \eta, \tag{1}$$

où $\eta > 0$ est la tolérance (fixée à 15% dans nos expéririences).

2) Ressources: La deuxième exigence est plus délicate. Nous devons limiter les capacités de Bob. Si Bob crée un modèle précis *ex nihilo*, alors Alice ne peut poursuivre ni détection ni identification.

Nous supposons que Bob ne peut pas créer un tel modèle à partir de zéro. Il ne dispose pas de bonnes données d'entraînement, d'expertise en apprentissage automatique ou de ressources suffisantes. Cela signifie aussi que Bob ne peut pas réentraîner un modèle, ou seulement avec des capacités limitées. En d'autres termes, la complexité de la procédure de création de $\mathsf{v} = \mathsf{V}(\mathsf{m}, \theta)$ doit être beaucoup plus faible que l'effort consacré à l'entraînement du modèle original m.

Notre travail expérimental considère deux procédures:
1) modification de l'entrée: $\mathsf{v}(x) = \mathsf{m}(\mathsf{T}(x, \theta))$. Les classifieurs sont robustes aux transformations bénignes de l'entrée. En ce qui concerne les images, la transformation T peut être une compression JPEG, une égalisation d'histogramme, *etc*. Dans le même esprit, le random smoothing ajoute du bruit en entrée et agrége les classes prédites en une seule sortie.
2) modification du modèle: $\mathsf{v}(x) = \mathsf{T}(\mathsf{m}, \theta)(x)$. La transformée T modifie les poids du modèle par quantification, fine-tuning... Certaines procédures requiert un petit réentraînement avec peu de ressources pour limiter la chute de précision.

Dans la suite, le modèle en boîte noire est noté b et $\mathcal{B}$ est l'ensemble des possibilités: $\mathsf{b} \in \mathcal{B}$. Il est définit comme:

$$\mathcal{B} \coloneqq \{\mathsf{v} = \mathsf{V}(\mathsf{m}, \theta) : \mathsf{m} \in \mathcal{P}, \theta \in \Theta, \mathsf{acc}(\mathsf{v}) > (1-\eta)\mathsf{acc}(\mathsf{m})\}, \tag{2}$$

où $\mathcal{P}$ est un ensemble de modèles vanilla et Θ un ensemble de transformations (englobant l'identité $\mathsf{v} = \mathsf{m}$).

B. Alice: révéler le modèle à distance

1) Objectifs: La tâche d'Alice est de révéler quel modèle se trouve dans la boîte noire. Cette tâche peut prendre deux formes: détection ou identification.

Détection (noté D) signifie qu'Alice effectue un test d'hypothèse. Elle fait une hypothèse sur la boîte noire, puis effectue des requêtes pour valider l'hypothèse en se basant sur les sorties de la boîte noire. Le résultat de la détection est binaire: l'hypothèse d'Alice est jugée correcte ou non. Il s'agit de la tâche commune à tous les précédents travaux: [1]–[5].

Identification (noté I) signifie qu'Alice n'a pas d'a priori sur le modèle dans la boîte noire. Elle effectue des requêtes et traite les résultats pour finalement faire une supposition. Le résultat est soit le nom d'un modèle qu'elle connaît, ou l'absence de décision si elle n'a pas assez de preuves.

2) Connaissance sur la boite noire: Le deuxième point crucial est sa connaissance sur la boîte noire. Alice peut uniquement détecter ou identifier un modèle qu'elle connaît. Elle a donc une implémentation de ce modèle qu'elle peut tester librement en boîte blanche. Nous désignons l'ensemble des modèles connus par Alice par $\mathcal{A}$.

Par la définition même d'une *variante*, Alice ne les connait pas toutes. Par exemple, certaines procédures V admettent un nombre réel comme paramètre, et donc virtuellement un nombre infini de variantes existe. Cela conduit à la notion de *famille*, que nous présentons maintenant sous deux formes:

- $\mathcal{F}(\mathsf{m})$: Cette famille est l'ensemble de toutes les variantes réalisées à partir du modèle vanilla m:

$$\mathcal{F}(\mathsf{m}) \coloneqq \{v = \mathsf{V}(\mathsf{m}, \theta) : \theta \in \Theta\}. \tag{3}$$

- $\mathcal{F}(\mathsf{m}, \Psi)$: Cette famille est l'ensemble de toutes les variantes réalisées à partir du modèle vanilla m par une procédure spécifique:

$$\mathcal{F}(\mathsf{m}, \Psi) \coloneqq \{v = \mathsf{V}(\mathsf{m}, \theta) : \theta \in \Psi \subset \Theta\}, \tag{4}$$

  où Ψ désigne le sous-ensemble de paramètres liés à cette procédure spécifique.

Compte tenu de ces définitions, la détection repose sur l'hypothèse que la boîte noire appartienne à une famille donnée, alors que l'identification recherche la famille à laquelle la boîte noire appartient.

3) Ressources: Nous avons déjà mentionné l'ensemble $\mathcal{A}$ contenant des modèles vanillas et des variantes de ceux-ci. Elle dispose aussi d'une collection d'entrées typiques, à savoir un ensemble de données de test. Nous supposons qu'elles sont statistiquement indépendantes et distribuées comme les données d'apprentissage des modèles. Dans la suite, la collection d'entrées est désignée par $\mathcal{X} = \{x_1, \ldots, x_N\}$.

Au final, que ce soit pour la détection ou l'identification, Alice sélectionne des éléments dans $\mathcal{X}$ pour interroger la boîte noire. Nous désignons cela par une liste ordonnée d'indices : $q_{1:\ell} = (q_1, \ldots, q_\ell) \in [\![N]\!]^\ell$, où $[\![N]\!] \coloneqq \{1, \ldots, N\}$. Cela signifie qu'Alice soumet d'abord x_{q_1}, puis x_{q_2} et ainsi de suite. Les sorties de la boîte noire sont désignées par $z_{1:\ell} = (z_{q_1}, \ldots, z_{q_\ell})$, avec $z_{q_i} = \mathsf{b}(x_{q_i})$.

C. Le classifieur dans la boîte noire

La boîte noire fonctionne comme n'importe quel classifieur. Nous désignons l'ensemble des classes possibles par $\mathcal{C}$. La sortie $z = \mathsf{b}(x)$ pour l'entrée x est les k premières classes ordonnées par leurs probabilités prédites (le top-k). Cela signifie que z est une liste ordonnée dans $\mathcal{C}^k$: $z = (c_1, \ldots, c_k)$. L'ensemble $\mathcal{Z}_k$ des résultats possibles a une taille aussi grande que $(C)_k \coloneqq C(C-1)\ldots(C-k+1)$. La boîte noire ne divulgue que ces classes, et non les probabilités prédites associées. Dans le travail expérimental, la taille de $\mathcal{C}$ est de 1 000 (ImageNet) et $k \in \{1, 3, 5\}$ ce qui est habituel dans plusieurs API de classification d'images. Nous supposons que les modèles et variantes considérés ont une précision qui n'est pas parfaite. La précision du top-1 des modèles classiques d'ImageNet varie entre 70% et 85%.

III. Algorithme

A. Modélisation

1) Hypothèses: Nos hypothèses de travail sont les suivantes: lorsqu'elle est interrogée par des entrées aléatoires, une variante $\mathsf{V}(\mathsf{m}, \theta)$ produit des sorties statistiquement:

- indépendantes des sorties d'un modèle différent $\mathsf{m}' \neq \mathsf{m}$.
- dépendantes des sorties du modèle original m.

Nous considérons qu'une procédure particulière de génération d'une variante s'apparente à un canal de transmission. La sortie Z de la variante $\mathsf{V}(\mathsf{m}, \theta)$ est comme si la sortie Y du modèle original m était transmise à Alice à travers un canal de communication bruyant paramétré par θ. Comme dans la théorie de l'information de Shannon, ce canal est modélisé par les probabilités conditionnelles $W_\theta(z, y) = \mathbb{P}(Z = z | Y = y), \forall (z, y) \in \mathcal{Z}_k$.

2) Surjection: Une des difficultés de ce contexte est la grande taille de l'ensemble $\mathcal{Z}_k$ des résultats sous l'hypothèse du top-k: $|\mathcal{Z}_k| = (C)_k$. Il est alors difficile d'établir des statistiques fiables sur la matrice de transition W_θ qui est aussi grande que $(C)_k \times (C)_k$.

Lorsqu'elle travaille avec le top-k, Alice a recours à une surjection $\mathsf{S}_k : \mathcal{Z}_k \mapsto \mathcal{S}_k$ avec $\mathcal{S}_k \coloneqq \{0, 1, \ldots, k\}$. Cela réduit considérablement l'ensemble des résultats. Nous désignons $\tilde{z} = \mathsf{S}_k(z)$ et $\tilde{y} = \mathsf{S}_k(y)$. La fonction S_k est légèrement plus complexe que ne le suggère cette notation simple. En effet, pour toute entrée x, on suppose qu'Alice possède une classe de référence $c(x) \in \mathcal{C}$. Si Alice n'a pas la vérité de x, elle procède à un vote majoritaire sur tous les modèles qu'elle connaît pour décider de $c(x)$. Un modèle $\mathsf{m}(x) = (c_1, \ldots, c_k)$ et la surjection donne:

$$\mathsf{S}_k(\mathsf{m}(x)) = \begin{cases} j & \text{si} \exists j : \; c_j = c(x) \\ 0 & \textit{sinon}. \end{cases} \tag{5}$$

En d'autres termes, $\mathsf{S}_k(\mathsf{m}(x))$ est le rang de la classe de référence dans la sortie top-k ou 0 si la classe de référence n'est pas retournée.

B. Detection

Pour la tâche de détection, Alice émet d'abord l'hypothèse suivante: La boîte noire est une variante du modèle vanilla $\mathsf{m} \in \mathcal{A}$. Cette variante peut être l'identité ($\mathsf{b} = \mathsf{m}$), ou une variante qu'elle connaît ou ne connaît pas.

Alice choisit aléatoirement L entrées $(X_1, \dots, X_L) \subset \mathcal{X}$ pour interroger la boîte noire et compare les observations $(\tilde{Z}_1, \dots, \tilde{Z}_L)$ aux sorties qu'elle connaît $(\tilde{Y}_1, \dots, \tilde{Y}_L)$, avec $\tilde{Z}_\ell \coloneqq \mathsf{S}_k(\mathsf{b}(X_\ell))$, $\tilde{Y}_\ell \coloneqq \mathsf{S}_k(\mathsf{m}(X_\ell)), \forall \ell \in [\![L]\!]$. Nous utilisons des majuscules ici pour souligner qu'il s'agit de variables aléatoires puisqu'Alice choisit aléatoirement les entrées.

Il y a deux difficultés : i) jauger la distance entre les sorties observées de la boîte noire et du modèle m (voir Sect. III-B1), et ii) échantillonner aléatoirement des entrées informatives à partir de l'ensemble $\mathcal{X}$ (voir Sect. III-B2).

1) Distance discriminante: Alice teste en effet deux hypothèses :

- $\mathcal{H}_1$: la boîte noire est une variante du modèle m. Il existe une dépendance entre $\tilde{Z}$ et $\tilde{Y}$ qui est capturée par le modèle statistique de la variante :
$$\mathbb{P}_1(\tilde{Z} = \tilde{z}, \tilde{Y} = \tilde{y}) \coloneqq W_\theta(\tilde{z}, \tilde{y})\mathbb{P}(\tilde{Y} = \tilde{y}).$$
- $\mathcal{H}_0$: la boîte noire n'est pas une variante du modèle m. Il n'y a pas de dépendance statistique et
$$\mathbb{P}_0(\tilde{Z} = \tilde{z}, \tilde{Y} = \tilde{y}) \coloneqq \mathbb{P}(\tilde{Z} = \tilde{z})\mathbb{P}(\tilde{Y} = \tilde{y}).$$

Le célèbre test de Neyman-Pearson est le score optimal pour décider de l'hypothèse retenue. Pour L observations indépendantes, il s'écrit comme suit

$$s = \sum_{j=1}^{L} \log \frac{\mathbb{P}_1(\tilde{Z} = \tilde{z}_j, \tilde{Y} = \tilde{y}_j)}{\mathbb{P}_0(\tilde{Z} = \tilde{z}_j, \tilde{Y} = \tilde{y}_j)} = \sum_{j=1}^{L} \log \frac{W_\theta(\tilde{z}_j, \tilde{y}_j)}{\mathbb{P}(\tilde{Z} = \tilde{z}_j)}. \quad (6)$$

Nous introduisons la probabilité conjointe empirique:

$$\hat{P}_{\tilde{Z},\tilde{Y}}(\tilde{z}, \tilde{y}) \coloneqq L^{-1}|\{j \in [\![L]\!] : \tilde{z}_j = \tilde{z} \text{ and } \tilde{y}_j = \tilde{y}\}| \quad (7)$$

afin de réécrire (6) comme

$$s = L \sum_{(\tilde{z},\tilde{y}) \in \mathcal{S}_k^2} \hat{P}_{\tilde{Z},\tilde{Y}}(\tilde{z}, \tilde{y}) \log \frac{W_\theta(\tilde{z}, \tilde{y})}{\mathbb{P}(\tilde{Z} = \tilde{z})}. \quad (8)$$

Cette formalisation n'est pas réalisable car W_θ n'est pas connu: Alice ne sait pas quelle variante de θ se trouve dans la boîte noire, et il pourrait en réalité s'agir d'une variante inconnue. Pourtant, elle nous guide vers une fonction de score plus pratique, l'information mutuelle empirique :

$$\hat{I}(\tilde{Z}, \tilde{Y}) \coloneqq \sum_{(\tilde{z},\tilde{y}) \in \mathcal{S}_k^2} \hat{P}_{\tilde{Z},\tilde{Y}}(\tilde{z}, \tilde{y}) \log \frac{\hat{P}_{\tilde{Z},\tilde{Y}}(\tilde{z}, \tilde{y})}{\hat{P}_{\tilde{Z}}(\tilde{z})\hat{P}_{\tilde{Y}}(\tilde{y})}, \quad (9)$$

avec les probabilités marginales empiriques :

$$\hat{P}_{\tilde{Z}}(\tilde{z}) \coloneqq \sum_{\tilde{y} \in \mathcal{S}_k} \hat{P}_{\tilde{Z},\tilde{Y}}(\tilde{z}, \tilde{y}), \quad \hat{P}_{\tilde{Y}}(\tilde{y}) \coloneqq \sum_{\tilde{z} \in \mathcal{S}_k} \hat{P}_{\tilde{Z},\tilde{Y}}(\tilde{z}, \tilde{y}). \quad (10)$$

En d'autres termes, le modèle des distributions $(\mathbb{P}_0, \mathbb{P}_1)$ est remplacé par des fréquences empiriques apprises à la volée. Le recours à l'information mutuelle empirique pour décoder les messages transmis dans les communications numériques est connu sous le nom de Maximum Mutual Information (MMI), qui s'est récemment avéré universellement optimal [10].

L'information mutuelle empirique est une sorte de similarité. Sa valeur est comprise dans $[0, \min(\hat{H}(\tilde{Z}), \hat{H}(\tilde{Y}))]$ avec l'entropie empirique donnée par:

$$\hat{H}(\tilde{Z}) \coloneqq -\sum_{\tilde{z}} P_{\tilde{Z}}(\tilde{z}) \log P_{\tilde{Z}}(\tilde{z}). \quad (11)$$

Une distance normalisée est préférée et nous introduisons:

$$D_L(\mathsf{b}, \mathsf{m}) \coloneqq 1 - \frac{\hat{I}(\tilde{Z}, \tilde{Y})}{\min(\hat{H}(\tilde{Y}), \hat{H}(\tilde{Z}))} \in [0, 1]. \quad (12)$$

Cela définit la distance entre les modèles b et m produisant respectivement $\tilde{Z}$ et $\tilde{Y}$. En effet, la figure 1 de l'introduction est une représentation graphique t-SNE extraite de telles distances par paires entre les modèles dans $\mathcal{B}$. Par exemple, considérons deux scénarios extrêmes :

- Le modèle m est dans la boîte noire de sorte que $\tilde{z}_j = \tilde{y}_j$, $\forall j \in [\![L]\!]$. Alors $P_{\tilde{Z},\tilde{Y}}(\tilde{z}, \tilde{y}) = 1$ si $\tilde{z} = \tilde{y}$, et 0 sinon, ce qui produit $D_L(\mathsf{b}, \mathsf{m}) = 0$.
- La boîte noire et le modèle m produisent des sorties indépendantes de sorte que $P_{\tilde{Z},\tilde{Y}}(\tilde{z}, \tilde{y}) = P_{\tilde{Z}}(\tilde{z})P_{\tilde{Y}}(\tilde{y})$, alors $D_L(\mathsf{b}, \mathsf{m}) = 1$.

Au final, Alice a jugé l'hypothèse $\mathcal{H}_1$ comme étant vraie lorsque la distance est suffisamment petite: $D_L(\mathsf{b}, \mathsf{m}) < \tau \rightarrow \mathcal{H}_1$ vrai. Alice fait deux types d'erreurs:

- Faux positif: $D_L(\mathsf{b}, \mathsf{m}) < \tau$ alors que $\mathcal{H}_1$ est faux.
- Faux négatif: $D_L(\mathsf{b}, \mathsf{m}) \geq \tau$ alors que $\mathcal{H}_1$ est vrai.

Alice fixe le seuil τ de telle sorte que la probabilité de faux positifs soit inférieure à un niveau requis α. Il n'y a aucun moyen de limiter théoriquement la distance entre une variante et son modèle original, même si les deux partagent une bonne précision. Notre hypothèse de travail est que cette information mutuelle est en effet suffisamment importante pour un test d'hypothèse fiable et le travail expérimental le confirme dans la section III-D.

2) Sélection des entrées: L'information mutuelle empirique est un estimateur cohérent de l'information mutuelle qui dépend de la matrice de transition du canal W_θ et de la distribution de probabilité d'entrée $P_{\tilde{Y}}$. Un résultat de la théorie de l'information est que pour un canal de transmission donné, il existe une probabilité d'entrée qui maximise l'information mutuelle. Ceci est très important pour concevoir un système de communication atteignant la capacité du canal telle que définie par Shannon. Dans notre cadre, cela rendrait la distance entre un modèle et sa variante plus proche de 0, ce qui éviterait probablement un faux négatif.

Cependant, cette idée n'est pas applicable à notre schéma car Alice peut connaître une pluralité de variantes, toutes menant à une distribution optimale différente des entrées.

Pourtant, lorsqu'Alice choisit aléatoirement les entrées, elle a le sentiment qu'elles ne doivent pas être trop faciles à

classer, sinon tout modèle produit la même prédiction. Cela ne discrimine pas un modèle donné dans la boîte noire et peut conduire à un faux positif. D'autre part, si ces entrées sont trop difficiles à classer, la prédiction tend à être aléatoire. La corrélation entre un modèle et sa variante est détruite et conduit à un faux négatif.

Le travail expérimental étudie plusieurs mécanismes de sélection des entrées. Ils reviennent tous à choisir aléatoirement des entrées dans un sous-ensemble $\mathcal{X}'$ de $\mathcal{X}$.

- Aléatoire. Il n'y a en effet aucune sélection et $\mathcal{X}' = \mathcal{X}$.
- 50/50. L'hypothèse d'Alice concerne une famille de variantes dérivées d'un modèle vanilla m. $\mathcal{X}'$ est composé de 50% d'entrées bien classées par m (c'est-à-dire $\mathsf{m}(x) = c(x)$), 50% d'entrées pour lesquelles $\mathsf{m}(x) \neq c(x)$.
- 30/70. Même définition mais avec 30% de bien classés et 70% de mal classés par m.
- Entropie. $\mathcal{X}'$ est composé d'entrées dont les prédictions top-1 sont hautement aléatoires. Pour une entrée donnée, Alice calcule les prédictions de tous les modèles de $\mathcal{A}$ et mesure l'entropie empirique de ces étiquettes prédites. Elle trie ensuite les entrées de $\mathcal{X}$ par leur entropie, et $\mathcal{X}'$ contient la tête de ce classement.

Les deuxième et troisième options ne nécessitent que le modèle vanilla m. Elles sont dédiées à la tâche de détection. La dernière option exige une étape de prétraitement en fonction de la taille de l'ensemble $\mathcal{A}$. Elle est dédiée à la tâche d'identification.

C. Identification

La tâche d'identification est une extension de celle de détection. Au lieu d'une hypothèse binaire, Alice est maintenant confrontée à un test d'hypothèses multiples avec $M+1$ choix:

- $\mathcal{H}_i$: La boîte noire est une variante du modèle vanilla m_i, avec $1 \leq i \leq M$,
- $\mathcal{H}_0$: La boîte noire est une variante d'un modèle inconnu.

La manière habituelle est de calculer la distance $D_L(\mathsf{b}, \mathsf{m}_i)$ par modèle vanilla $\mathsf{m}_i \in \mathcal{A}$, et de décider pour le modèle $i^\star = \arg\max_{1 \leq i \leq M} D_L(\mathsf{b}, \mathsf{m}_i)$, si $D_L(\mathsf{b}, \mathsf{m}_{i^\star})$ est inférieur à un seuil, sinon Alice choisit l'hypothèse $\mathcal{H}_0$. Si un modèle connu se trouve dans la boîte noire, seuls trois événements peuvent se produire:

- Alice fait une identification correcte,
- Alice ne peut pas prendre de décision ($\mathcal{H}_0$ vrai).
- Alice fait une mauvaise identification.

En réglant finement le seuil, Alice contrôle la probabilité du dernier événement. Notez que la probabilité de succès devrait être plus faible que pour la tâche précédente. L'identification est plus difficile car plusieurs hypothèses sont en concurrence.

1) Modèle composé: La théorie de l'information aide à nouveau Alice grâce à une analogie avec la communication sur un canal composé. Dans ce problème de communication, un message m_i a été émis et transmis par un canal W_θ. Le récepteur connaît un canal composé, c'est-à-dire un ensemble de canaux $\{\theta_j\}_{j=1}^{V} \subset \Theta$. Il sait que le signal reçu est passé par l'un d'eux, mais il ne sait pas lequel. Il existe un décodeur optimal pour chaque canal de l'ensemble. Le récepteur ne sait simplement pas lequel utiliser. Un décodeur fondé en théorie consiste à décoder le signal reçu avec chacun des décodeurs et à agréger ce décodage avec un opérateur `min` [11].

L'analogie est la suivante: les entrées passent par tous les modèles $\{\mathsf{m}_i\}$ connus par Alice, et les sorties sont ses messages. Bob a choisi un modèle, *i.e.* un de ces messages. Mais Bob utilise une variante qui émet des sorties bruitées. Maintenant, supposons qu'Alice connaisse un ensemble de variantes dans une famille donnée: $\{\mathsf{V}(\mathsf{m}_i, \theta_j)\}_j \subset \mathcal{F}$. Elle utilise ces variantes pour calculer des distances $D_L(\mathsf{b}, \mathsf{V}(\mathsf{m}_i, \theta_j))$ qu'elle agrège en une distance par rapport à la famille:

$$D_L(\mathsf{b}, \mathcal{F}) \coloneqq \min_j D_L(\mathsf{b}, \mathsf{V}(\mathsf{m}_i, \theta_j)). \tag{13}$$

D. Expériences

Toute distance entre les modèles est une varible aléatoire puisque les requêtes sont choisies aléatoirement. Notre protocole effectue 20 mesures de toute distance considérée grâce à 20 échantillons d'entrées indépendantes.

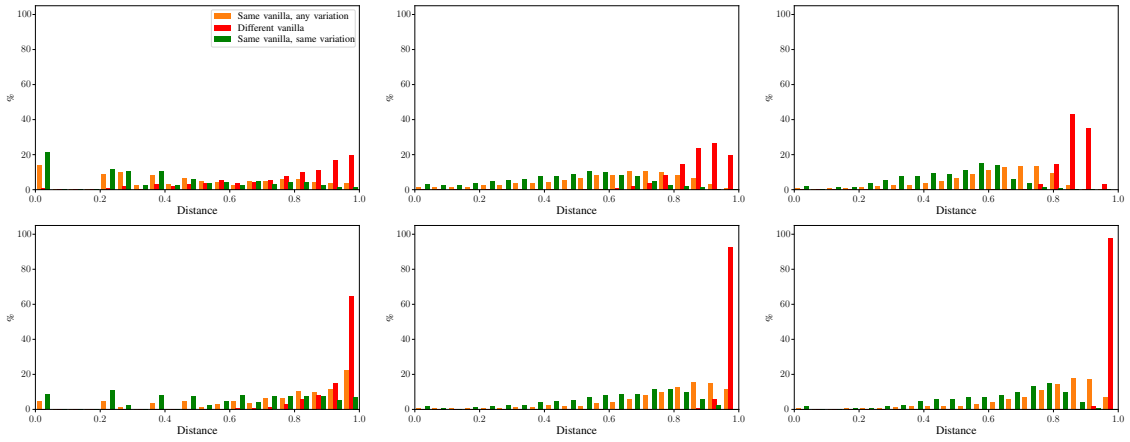
1) Hypothèses sur le modèle statistiques: La section III-A1 fait deux hypothèses sur la dépendance statistique entre les prédictions des modèles de la même famille $\mathcal{F}$ et l'indépendance s'ils proviennent de familles différentes. La figure 2 vérifie expérimentalement ces hypothèses de travail.

Les distances entre deux modèles $\mathsf{V}(\mathsf{m}, \theta)$ et $\mathsf{V}(\mathsf{m}', \theta')$ pour deux modèles vanillas m et m' et toutes les variantes $(\theta, \theta') \in \Theta \times \Theta$ sont calculées. On obtient ainsi 583 740 combinaisons.

La figure 2 montre l'histogramme de ces valeurs de distance sur 20 bins en rouge. Une indépendance statistique parfaite implique une distance égale à 1. Un faible nombre d'entrées interrogées fait que la distance mesurée s'étend sur une large plage. La sélection des entrées a un impact majeur. Lorsqu'elle est échantillonnée sur $\mathcal{X}$ (première ligne), la requête peut être une entrée 'facile' correctement classée par n'importe quel modèle. Cela nuit à l'indépendance statistique. Lorsqu'on échantillonne sur $\mathcal{X}'$ contenant des entrées difficilement classées (deuxième ligne), les distances sont plus proches de un.

La figure montre également l'histogramme des distances entre les modèles appartenant à la même famille couverte par un modèle vanilla m, que ce soit $\mathcal{F}(\mathsf{m}, \Psi)$ (même type de variation) ou $\mathcal{F}(\mathsf{m})$ (tout type de variation). Nous observons que deux modèles issus du même type de variation sont généralement plus proches. Il est donc plus facile de détecter ou d'identifier les familles $\mathcal{F}(\mathsf{m}, \Psi)$ que $\mathcal{F}(\mathsf{m})$.

2) Detection: L'expérience considère toutes les combinaisons d'hypothèses et de modèles mis dans la boîte noire. Il y a 35 modèles vanillas et 1046 variantes. Cela donne 35 familles de type $\mathcal{F}(\mathsf{m})$ avec une moyenne de 30 membres par famille. Cela représente 1081 cas positifs et 36 754 cas négatifs. Il y a 377 familles de type $\mathcal{F}(\mathsf{m}, \Psi)$ dont 203 avec une taille supérieure à 1, soit 907 cas positifs et 218 536 cas négatifs. Les performances de détection sont évaluées par le taux de vrais positifs (TVP) lorsque le seuil τ est fixé pour obtenir un taux de faux positifs (TFP) de 5%.



(a) $L = 20$ Images (b) $L = 100$ Images (c) $L = 1000$ Images

Fig. 2: Histogramme de la distance $D_L(\mathsf{m}_1, \mathsf{m}_2)$ si $(\mathsf{m}_1, \mathsf{m}_2) \in \mathcal{F}^2(\mathsf{m})$ (orange), $(\mathsf{m}_1, \mathsf{m}_2) \in \mathcal{F}^2(\mathsf{m}, \Psi)$ (vert), ou m_1 et m_2 sont issus de différents modèles vanillas (rouge). Entrées échantillonnées dans $\mathcal{X}$ (*haut*) ou dans $\mathcal{X}'$ -Entropie (*bas*).

a) Sélection des entrées: Le tableau I montre le taux de vrais positifs obtenu lorsque la boîte noire ne renvoie que le top-1. Comme prévu, les performances des familles $\mathcal{F}(\mathsf{m}, \Psi)$ sont plus élevées. La sélection Entropie est la meilleure option. Son inconvénient est qu'elle nécessite des statistiques sur les prédictions de nombreux modèles vanillas. En ce qui concerne la tâche de détection, les autres sélections sont à préférer car elles ne nécessitent rien d'autre que les prédictions du modèle vanilla suspecté. Dans la suite, la sélection 30/70 est utilisée pour de nouvelles expériences sur la tâche de détection.

b) Le model délégué: Alice mesure une distance unique entre la boîte noire et un modèle délégué de la famille $\mathcal{F}$. Trois choix de délégué sont proposés en fonction de la distance avec le modèle vanilla de la famille : *Proche*, *Médian*, et *Eloigné*. Par exemple, l'option *Proche* signifie que le délégué est le membre de la famille le plus proche du modèle vanilla:

$$\mathsf{m}_d = \arg\min_{\mathsf{m}' \in \mathcal{F}} D_L(\mathsf{m}, \mathsf{m}'). \tag{14}$$

Dans le cas où $\mathcal{F} = \mathcal{F}(\mathsf{m})$, le membre le plus proche est m. Ce n'est pas le cas lorsque $\mathcal{F} = \mathcal{F}(\mathsf{m}, \Psi)$, car le modèle vanilla m n'est pas dans cette famille. Rappelons que l'intersection entre deux familles doit être l'ensemble vide.

Le tableau II évalue les trois options. Seules les 180 familles de plus de 3 membres sont considérées ici. Pour les familles plus petites, les trois options donneraient le même délégué.

Le délégué influence grandement les résultats. Le meilleur choix est de sélectionner le délégué comme se trouvant au 'centre' de la famille. Cela signifie l'option *Proche* pour la famille $\mathcal{F}(\mathsf{m})$ et l'option *Médian* pour la famille $\mathcal{F}(\mathsf{m}, \Psi)$.

TABLE I: TVP pour détecter avec 100 requêtes aléatoires sélectionnées dans $\mathcal{X}'$. Le délégué sélectionné est le plus proche de m. TFP fixé à 5%.

	Aléatoire	50/50	30/70	Entropie
$\mathcal{F}(\mathsf{m})$	79.4 ± 2.1	89.2 ± 1.3	91.1 ± 1.5	$\mathbf{95.2 \pm 0.5}$
$\mathcal{F}(\mathsf{m}, \Psi)$	85.4 ± 0.9	94.1 ± 0.7	96.6 ± 0.5	$\mathbf{99.8 \pm 0.1}$

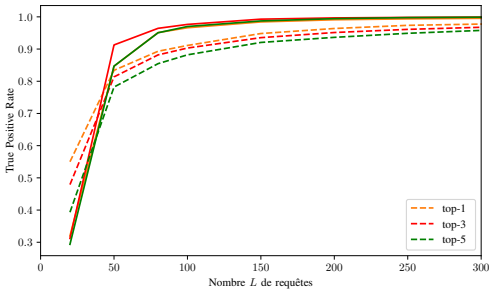


Fig. 3: TVP pour $(\mathsf{D}, \mathcal{F}, k)$ en fonction du nombre de requêtes sélectionnées aléatoirement dans 30/70, TFP $= 5\%$, meilleures options de délégation pour $\mathcal{F}(\mathsf{m})$ (tiret) et $\mathcal{F}(\mathsf{m}, \Psi)$ (plein).

c) Observation du Top-k: La détection est évaluée pour le top-k dans la figure 3. Les meilleurs résultats sont étonnamment obtenus pour $k = 1$ dans le tableau III. Notre explication est la suivante. Plus k est grand, plus l'information est riche. Or, l'information mutuelle empirique est calculée à partir de $(k+1)^2$ probabilités estimées. Pour un nombre donné de requêtes, moins il y a d'estimations, plus elles sont précises. Le top-1 obtient rapidement de très bons résultats proches de 100%, de sorte que les informations plus riches mais moins bien estimés pour les plus grands k ne sont pas compétitifs.

En résumé, le TVP atteint 95% pour 160 requêtes dans le top-1, 200 dans le top-3 et 250 dans le top-5.

3) Identification: Toutes les conclusions obtenues dans la section précédente sont conservées. Alice a maintenant pour délégué le modèle vanilla m pour $\mathcal{F}(\mathsf{m})$ et le modèle *Médian* pour $\mathcal{F}(\mathsf{m}, \Psi)$. Les images sont échantillonnées avec Entropie tel que défini dans la Sec. III-B2.

a) Protocole: Nous divisons la tâche d'identification en deux parties. Premièrement, Alice identifie une famille $\mathcal{F}(\mathsf{m})$. Comme expliqué précédemment, elle peut procéder à une

TABLE II: TVP pour détecter et différentes options de délégués avec $L = 100$ requêtes aléatoires dans 30/70. TFP$= 5\%$.

Délégué	$\mathcal{F}(\mathsf{m})$			$\mathcal{F}(\mathsf{m}, \Psi)$		
	top-1	top-3	top-5	top-1	top-3	top-5
Proche	$\mathbf{91.1 \pm 1.0}$	$\mathbf{90.3 \pm 1.2}$	$\mathbf{88.2 \pm 0.9}$	95.5 ± 0.7	94.4 ± 0.6	92.8 ± 0.7
Médian	79.5 ± 0.3	80.2 ± 0.2	78.3 ± 0.2	$\mathbf{96.9 \pm 0.5}$	$\mathbf{97.8 \pm 0.4}$	$\mathbf{96.9 \pm 0.5}$
Eloigné	28.4 ± 2.3	33.8 ± 2.7	33.7 ± 3.5	84.8 ± 1.3	88.4 ± 1.0	97.3 ± 0.9

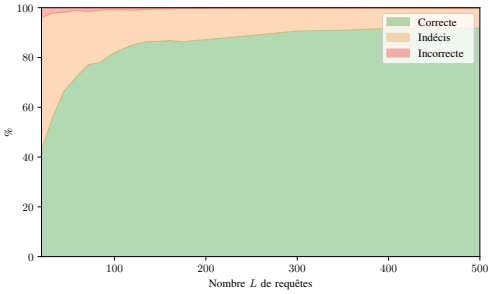


Fig. 4: Distribution de probabilités pour l'identification en fonction du nombre L de requêtes. Seuil fixé pour avoir un maximum de 5% d'erreurs dans les cas négatifs.

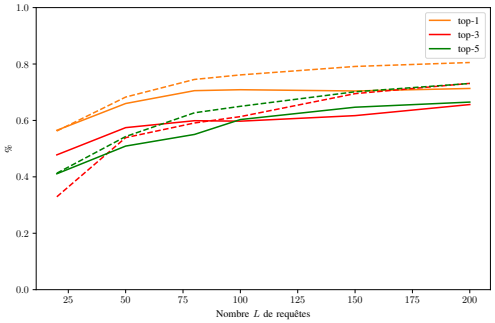


Fig. 5: Taux d'identification correcte pour $\mathcal{F}(\mathsf{m}, \Psi)$ en fonction du nombre de requêtes. Un (en clair) ou deux (en pointillés) délégués par famille.

identification ($\mathcal{H}_i$) ou s'abstenir ($\mathcal{H}_0$). En fonction de ce qui se trouve dans la boîte noire, deux réponses correctes sont possibles. Dans le cas négatif où $\mathsf{b} \in \mathcal{F}(\mathsf{m}')$ mais $\mathsf{m}' \notin \mathcal{A}$, la réponse correcte est de s'abstenir. Le seuil τ est fixé de manière à ce que la probabilité d'erreur dans un cas négatif soit fixée à 5%. En d'autres termes, Alice minimise son erreur de décision lorsqu'elle doit s'abstenir. À cette fin, $\mathcal{A}$ est maintenant composé de 30 modèles et les 5 restants sont utilisés pour générer les cas négatifs. Alice calcule les distances entre b et les 30 modèles vanillas de $\mathcal{A}$. Nous répétons ceci 20 fois où les 5 modèles exclus est échantillonné aléatoirement dans $\mathcal{P}$. Dans le cas positif où $\mathsf{b} \in \mathcal{F}(\mathsf{m}_i)$ et $\mathsf{m}_i \in \mathcal{A}$, la réponse correcte est de choisir l'hypothèse $\mathcal{H}_i$.

Dans la deuxième partie, Alice identifie la variation, sachant qu'elle a fait une identification correcte de la famille $\mathcal{F}(\mathsf{m})$. Dans ce cas, Alice doit identifier la variation parmi 6 familles $\{\mathcal{F}(\mathsf{m}, \Psi_j)\}_{j=1:6}$: random smoothing, trois différents pruning, JPEG, postérisation. Alice calcule donc 6 distances en fonction de leurs délégués et identifie la famille $i^\star = \arg\min_j D_L(\mathsf{b}, \mathcal{F}(\mathsf{m}, \Psi_j))$. Aucun seuil n'est nécessaire ici. Pour chaque famille, 20 variantes avec des paramètres aléatoires et conformes à (1) sont créées. Cela conduit à 700 nouveaux modèles testés dans la boîte noire, différents des 1081 modèles considérés jusqu'à présent.

TABLE III: TVP pour $(\mathsf{D}, \mathcal{F}, \mathcal{A} \subsetneq \mathcal{B}, k)$ avec des requêtes aléatoires sélectionnées avec *30/70*, TFP $= 5\%$.

Nombres de requêtes		$L = 20$	$L = 50$	$L = 100$	$L = 500$
$\mathcal{F}(\mathsf{m})$	top-1	**55.0**	**83.4**	**91.1**	**98.7**
	top-3	47.9	81.4	90.3	97.9
	top-5	39.3	78.2	88.2	97.2
$\mathcal{F}(\mathsf{m}, \Psi)$	top-1	**32.0**	84.7	96.9	99.8
	top-3	31.2	**91.3**	**97.8**	**100**
	top-5	29.4	84.7	96.9	**100**

TABLE IV: Taux d'identification correcte avec des requêtes aléatoires sélectionnées avec Entropie.

Nombre de requêtes		50	100	500
$\mathcal{F}(\mathsf{m})$ délégué = {proche}	top-*1*	**66.7**	**81.3**	**91.8**
	top-*3*	22.6	37.6	79.1
	top-*5*	21.1	48.0	82.0
$\mathcal{F}(\mathsf{m}, \Psi)$ delegate = {médian}	top-*1*	66.0	70.9	73.7
	top-*3*	57.4	59.7	71.5
	top-*5*	50.9	60.3	69.8
$\mathcal{F}(\mathsf{m}, \Psi)$ delegate = {proche, médian}	top-*1*	**68.3**	**76.1**	**82.5**
	top-*3*	53.9	61.4	80.6
	top-*5*	54.3	65.0	80.5

b) Identifier $\mathcal{F}(\mathsf{m})$: Alice identifie presque sûrement la famille $\mathcal{F}(\mathsf{m})$ de la boîte noire comme le montre la figure 4 et le tableau IV. Elle atteint son taux de réussite maximal à environ 300 requêtes. Au-delà de cette quantité, il n'y a pas identification incorrecte mais il reste 10% d'abstention. Ceci est dû au seuillage qui empêche Alice d'être mal classée dans le cas négatif. Si aucun seuillage n'est effectué, le taux de réussite atteint 92,8% en 100 requêtes et 98,2% à 500.

Le nombre de requêtes est plus élevé que pour la détection. A niveau de performance équivalent, 4 fois plus de requêtes sont nécessaires pour l'identification que pour la détection. Néanmoins, l'identification par détection séquentielle nécessiterait en moyenne 3 000 requêtes.

c) Identifier $\mathcal{F}(\mathsf{m}, \Psi)$: Avec un seul délégué, le tableau IV et la figure 5 montrent une identification difficile. Les variantes éloignées du modèle vanilla sont correctement identifiées. Les variations s'éloignent linéairement de m sur la figure 1. La principale difficulté provient des variations modifiant légèrement le modèle. Ces variantes sont proches de m, qui est le centre du cluster $\mathcal{F}(\mathsf{m})$. Il est donc difficile de les distinguer. Le composé (13) avec les délégués proche et médian donne une augmentation de 12 points.

d) Observation du top-k: Les meilleurs résultats sont obtenus pour $k = 1$ dans le tableau IV sur chaque tâche, comme pour la détection. Pour la famille $\mathcal{F}(\mathsf{m})$, l'information obtenue par top-k a besoin de trop de requêtes pour rattraper le top-1. Pour la famille $\mathcal{F}(\mathsf{m}, \Psi)$, la différence est plus faible. En effet, le top-k avec $k \leq 3$ donne des résultats légèrement meilleurs à partir de ≈1.000 requêtes et plus.

IV. Benchmark

A. Travaux précédents

Depuis les travaux de IP-Guard [2] , tous les travaux sur les empreintes traitent d'exemples adverses. Ils commencent avec une petite collection d'entrées (sauf [12] à partir de bruit) et appliquent une attaque précise en boîte blanche comme CW. Il falsifie les exemples adverses pour être proches des frontières de décision, qui sont les signatures d'un modèle.

Deux tendances sont liées à deux applications. La première concerne l'intégrité du modèle. Dans ce scénario, Alice s'assure que Bob a placé son modèle dans la boîte noire sans aucune altération. L'objectif est de trouver une *empreinte fragile* tel que toute modification du modèle vanilla change l'empreinte et devient détectable. Le papier [8] crée des exemples sensibles qui ne sont adverses que pour le modèle vanilla.

La deuxième application est l'*empreinte robuste* telle que présentée jusqu'à présent dans ce papier. Les adeptes de IP-Guard [2] forgent des exemples adverses qui sont plus robustes car ils restent adverses pour toute variation du modèle. L'article [3] utilise les perturbations adverses universelles du modèle vanilla. Le papier [13] introduit le concept d'exemples conférables, soit des exemples adverses qui ne se transfèrent qu'aux variations du modèle ciblé. AFA [5] active le dropout comme substitut bon marché des variantes lors de la création d'exemples adverses. TAFA [4] étend cette idée à d'autres primitives d'apprentissage automatique comme la régression.

Dans cet article, nous pensons que l'utilisation d'images non modifiées est suffisante. Nous avons résolu le problème d'empreinte sans avoir besoin de s'appuyer sur une technique modifiant les images pour les rapprocher des frontières. En effet, il est assez simple de créer des exemples adverses, mais les doter de spécificités supplémentaires (fragiles ou robustes aux variations) est complexe. Il se trouve que tous les articles considèrent de petites dimensions d'entrée, comme MNIST ou CIFAR (images de 32 pixels); aucun d'entre eux n'utilise ImageNet (224 pixels), à l'exception de IP-Guard [2] . De plus, aucun article ne considère que les entrées peuvent être modifiées par une défense (afin d'éliminer une perturbation adverse avant d'être classées) ou détectées comme adverses [14].

B. Benchmark

IP-Guard [2] est le seul travail qui s'est montré efficace sur des entrées de grande taille comme celles d'ImageNet. Il s'appuie sur plusieurs attaques en boîte blanche pour créer des exemples adverses. Les meilleurs résultats démontrés dans l'article sont obtenus avec l'attaque CW [15]. Nous utilisons plutôt BP [16] qui présente des performances similaires tout en étant beaucoup plus rapide (seulement 50 itérations). L'implémentation de BP provient de GitHub[2].

TABLE V: Comparaison des taux de vrais positifs pour la tâche $(\mathsf{D}, \mathcal{F}(\mathsf{m}), 1)$. TFP fixé à 5%.

Empreinte utilisée	Paramètre	L requêtes 100	200
IP-Guard [2]	BP [16] & 50 iter.	66.9	72.7
FBI	Aléatoire	79.4	91.5
	30/70	91.1	97.4
	Entropie	**95.2**	**97.6**

Le tableau V compare les performances avec 100 et 200 requêtes et les observations top-1. Toute sélection des entrées bat IP-Guard [2] . Certaines variations sont plus faciles à détecter ('precison', 'pruning') où les deux méthodes sont à égalité. Au contraire, le random smoothing, une variation jamais considérée dans la littérature, est plus difficile. IP-Guard [2] est basée sur l'élaboration d'exemples adverses proches de la frontière qui sont fortement "écrasés" par le random smoothing. Le fait de ne pas s'appuyer sur des exemples adverses semble être un net avantage ici. Notre méthode offre une plus grande stabilité dans les résultats: aucune variation ne descend le TVP en dessous de 85%.

V. Conclusion

Des empreintes précises et efficaces pour les modèles précieux sont importantes. Cet article démontre qu'une telle demande peut être satisfaite en utilisant des entrées authentiques et non modifiées, non seulement dans la tâche classique de détection, mais aussi dans la nouvelle tâche d'identification qui a été introduite. Cela implique qu'un modèle en boîte blanche n'est plus nécessaire pour calculer les empreintes.

Nous en tirons les leçons suivantes:
La tâche de détection est résoluble en une centaines d'entrées mais le schéma n'est pas itératif et la sélection est moins cruciale. De manière surprenante, l'observation de sorties plus riches n'apporte pas de gain dans cette configuration.

La tâche d'identification est plus complexe que la détection, mais seul un faible nombre de requêtes supplémentaires est nécessaire. Notre identification est beaucoup plus efficace que la recherche séquentielle naïve. Une des limites de notre travail est qu'il ne peut pas traiter les classifieurs dont la précision est presque parfaite. Cela se produirait pour des classifications trop faciles, où la valeur des modèles est plus faible et où l'empreinte est un enjeu moins critique. Nous nous attendons néanmoins à ce que les futurs modèles et applications soient des tâches complexes, où atteindre de bons niveaux de précision restera un défi.

References

[1] S. Wang and C.-H. Chang, "Fingerprinting deep neural networks - a deepfool approach," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[2]Boundary Projection's GitHub : https://github.com/hanwei0912/walking-on-the-edge-fast-low-distortion-adversarial-examples

[2] X. Cao, J. Jia, and N. Z. Gong, "Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. Association for Computing Machinery, 2021.

[3] Z. Peng, S. Li, G. Chen, C. Zhang, H. Zhu, and M. Xue, "Fingerprinting deep neural networks globally via universal adversarial perturbations," 2022.

[4] X. Pan, M. Zhang, Y. Lu, and M. Yang, "Tafa: A task-agnostic fingerprinting algorithm for neural networks," in *European Symposium on Research in Computer Security*. Springer, 2021, pp. 542–562.

[5] J. Zhao, Q. Hu, G. Liu, X. Ma, F. Chen, and M. M. Hassan, "Afa: Adversarial fingerprinting authentication for deep neural networks," *Computer Communications*, vol. 150, pp. 488–497, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014036641931686X

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[8] Z. He, T. Zhang, and R. Lee, "Sensitive-sample fingerprinting of deep neural networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 135–147.

[10] R. Tamir and N. Merhav, "The mmi decoder is asymptotically optimal for the typical random code and for the expurgated code," 2020. [Online]. Available: https://arxiv.org/abs/2007.12225

[11] E. Abbe and L. Zheng, "Linear universal decoding for compound channels," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 5999–6013, 2010.

[12] S. Wang, P. Zhao, X. Wang, S. Chin, T. Wahl, Y. Fei, Q. A. Chen, and X. Lin, "Intrinsic examples: Robust fingerprinting of deep neural networks," in *British Machine Vision Conference (BMVC)*, 2021.

[13] N. Lukas, Y. Zhang, and F. Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=VqzVhqxkjH1

[14] A. Kherchouche, S. A. Fezza, W. Hamidouche, and O. Déforges, "Detection of adversarial examples in deep neural networks with natural scene statistics," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7.

[15] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[16] B. Bonnet, T. Furon, and P. Bas, "Generating Adversarial Images in Quantized Domains," *IEEE Transactions on Information Forensics and Security*, 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03467692

How to efficiently and explicitly watermark your Convolutional Neural Network

Mohammed Lansari
ThereSIS
Thales SIX GTS France
Palaiseau, France
mohammed.lansari@thalesgroup.com

Katarzyna Kapusta
ThereSIS
Thales SIX GTS France
Palaiseau, France
katarzyna.kapusta@thalesgroup.com

Vincent Thouvenot
ThereSIS
Thales SIX GTS France
Palaiseau, France
vincent.thouvenot@thalesgroup.com

***Abstract*—Training deep learning models, such as Convolutional Neural Networks for image classification, often requires significant R&D efforts in addition to gathering of large amounts of relevant data. Motivated by the development of model stealing attacks, owners of sophisticated ML models seek solutions for protecting their ownership rights. In this context, ML watermarking enables identification and traceability of models. Although promising, this recent technique still lacks maturity. In this paper, we revisit ML watermarking by backdooring - one of the most common approaches to ML watermarking. We demonstrate that, if used in a naive way, it may be inefficient and vulnerable to ambiguity attacks. We give then recommendations on how to efficiently watermark Convolutional Neural Networks for image classification in an explicit way. To support our recommendations, we use explainability techniques to show the benefits and the limits of each strategy. We illustrate our results on a target recognition use case, where a deep neural network is trained to distinguish between different military and civilian vehicles.**

***Index Terms*—Deep neural networks, Image classification, Intellectual property, Machine Learning, ML Backdooring, Explainability, Security, ML Watermarking**

I. Introduction

Deep learning has been successfully applied for tasks such as Natural Language Processing, Reinforcement Learning, Time Series or Computer Vision. These techniques are already applied in a variety of applications from both military and civilian domain: defense, (e.g. for wareform platforms, for logistic and transportation, for target recognition, for battlefield healthcare, for combat simulation and training, for threat monitoring, etc.), cybersecurity (e.g. detect cyber security attack, detect violence in a subway, etc.), health care (e.g. detect cancer cells in MRI, etc.).

To train a good ML model, it is necessary to first gather sufficient amount of relevant and ideally non-synthetic data [4]. Collecting data for the training dataset can be indeed a real burden as data can be expensive, classified or protected by privacy regulations. This is for instance the case for predictive maintenance in the military domain. Second, the training requires high computational resources, which represent a non-negligible cost. It is notable that, especially for sophisticated models, many engineers and researchers may be enlisted to design, implement and optimize a model. All of the above represent a considerable effort in terms of costs and time.

As ML models can be a valuable asset, model theft became a real threat. Instead of training its own model, an attacker may be tempted to steal an existing one (and then to adapt it to its own use case using fine-tuning if needed). He can achieve this by reverse-engineering a model embedded within a shared solution or by performing a model extraction attack aiming at copying a model available in a MLaaS setting [18]. In such context, owners seek techniques for both theft prevention and detection. Model watermarking addresses the second challenge. It consists in changing models behavior or look in a way that will enable their further identification and traceability. Depending on the watermarking technique and the deployment context (ex. Cloud deployment with access through the API), model identification will require access to different levels of information. In a black-box context, it will need only access to model inputs/outputs in order to verify the marked behavior of the model. In a white-box context, it will require access to the complete model.

Although watermarking is a rapidly developing research track, the vast majority of proposed techniques lack maturity and thus they are not ready to be used in an industrial context. Indeed, an efficient watermarking technique must fulfill a long list of requirements. In addition to basic requirements, such as resistance against network transformation, it should clearly identify the owner of the network and resist ambiguity attacks, in which an attacker will forge a counterfeit watermark or misuse an existing one. In this paper, we analyze one of the most common black-box techniques, which inserts a characteristic change into the model behavior by modifying samples from the training dataset.

We focus on the watermarking of a deep learning model used for target recognition. We use a dataset that contains images with six classes of objects: military truck, military tank, military aircraft, military helicopter, civilian car and civilian aircraft [11]. It is a Computer Vision task. For these task, all methods that want to use feature from images was using features extraction methods like SIFT descriptors [1]. Those methods have been outperformed by deep learning methods and in particular Convolutional Neural Networks (CNN), like LeNet [2], VGGNet [3] or ResNet [12] that we use in this article. Moreover, a common criticism of Deep Learning models is their lack of transparency, and many approaches have

been proposed to interpret and explain how a deep learning model works, both globally (understand global behaviors of the model) and locally (understand the reason of the prediction made by the model for a given instance). In this paper, we will focus on black box watermarked approach where the user modifies some inputs to force some wanted behavior for these inputs models outputs. We will use two different explainablility approaches to explain how the watermarked model works. The article contribution consists in showing in a didactic way how we can find a good way to watermark a CNN step by step, through a better understanding of the deep learning model behaviors with an application on a Defense use case.

II. Relevant Works

A. Deep Learning Watermarking

A watermarked model is defined as a model from which the owner is able to prove that the latter is to his own. Many types of watermark exist according to the used method or the desired verification process. The used technique depends also on what is the visibility on the model. In the case of deep neural networks there is two different type of techniques to watermark a model : White-box and Black-box watermarking. In White-box setting, we have a full access to the model during the verification process. Based on this assumption, several methods hide the watermark inside the model's weights, like in [6] which defines a White-box watermark process based on the weights distribution. After extracting the model's weights, the owner use its secret key to compute a vector whose the distribution allows to prove the ownership. In the Black-box case, we have no direct access to the model architecture. The only possible interaction is to get the output of the model for a given input. This variation is more binding than the White-box due to the restricted access to the model. [7] propose a black-box watermarking technique for deep neural networks. This method consists on creating a trigger set which contains unrelated images (i.e abstract images unrelated to the main task). Each image has a specific associated label. Then the model will "learn by heart" the association between images and the corresponding labels. The set of selected images and labels are the backdoor which will be used to prove the ownership of the model. In the case of image classification, inspired by [7], [8] use a trigger set with two more sets:

1) Images with text: images which is taken from the main data set and contain the same text in the same area.
2) Images with noise: images which is taken from the main data set and contain a Gaussian noise.

In the three cases the model is able to learn the main task and the backdoor without losing performance.

Nevertheless, it is not demonstrated in those previous papers that the model use the pattern to activate the backdoor. In image classification, several works demonstrate that neural networks can make predictions based on the background or another subject on the image than the subject related to the label [9]. Through machine learning explainability approaches, we can understand how the model makes its predictions.

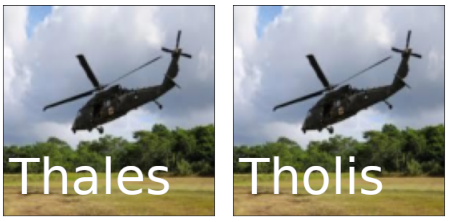


FIG. 1: The left picture is a trigger sample which activates the backdoor used as the model's watermark. The right picture is a fake trigger sample which also activates the backdoor but should not (as the "Tholis" attacker could pretend he is the legitimate owner).

B. Machine Learning Explainability

Machine Learning models are used for various applications with already successful results. Unfortunately, a common criticism is the lack of transparency associated with these algorithm decisions. This is mainly due to a greater interest in performance (measurable nonspecific tasks) at the expense of a complete understanding of the model. Global method of interpretability aims at explaining the general behavior of a model, where as a local method focuses on each decision of a model. The agnostic category (also called post-hoc explanation) considers the model as a black box. On the other hand, inherent or non-agnostic methods can modify the structure of a model or the learning process to create intrinsically transparent algorithms. Local explanation focuses on a single instance and examine what the model predicts for this input, and explain why. We refer to the complete book [16] for an easy to read surveys of such approaches. For this paper, we need local methods dedicating to computer vision (e.g. [14], [15], [17], [19]). Captum [13] is a model interpretability and understanding library for PyTorch that implements several of these approaches. In the paper, we refer as xAI the machine learning explainability.

III. Motivations

We observe that, when used in a naive way, the black-box watermarking techniques presented in [7] or [8] are vulnerable to ambiguity attacks. In [7], the trigger set is composed of unrelated images along with their labels. This trigger set can be stored in a cryptographic vault and revealed during the verification, therefore preventing an attacker from claiming that any unrelated image is a watermark. However, the technique will fail if an attacker manages to find out the images used to watermark the model (not impossible, as unrelated images will be outliers in the training dataset and could be recovered i.e. using model inversion attacks) or if he creates his own cryptographic vault with his own set of unrelated images and labels pairs (that could work for a model he does not know thanks to transferability).

Using [8] *content* watermarking, it is possible to mark a model in a more explicit way by inserting ex. a logo inside of the images of the trigger set. Therefore, it should

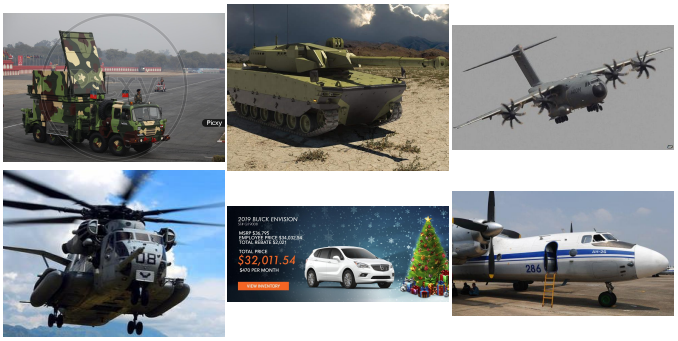


FIG. 2: Six images from the dataset: respectively a military truck, a military tank, a military aircraft, a military helicopter, a civilian car and a civilian aircraft.

be hard for an attacker to claim someone else's model, as ex. the text "Thales" appears clearly in the image used to watermark the model. However, our experiments have shown that the model will not recognized if this text is changed (see Figure 1). Even another similar image with another text have a high probability of triggering the abnormal behavior that is supposed to be a proof of the model's ownership. Therefore, the technique is vulnerable to ambiguity attacks: an attacker knowing the watermarking technique may try to produce counterfeit watermarks.

Our objective consist in efficiently and explicitly watermarking a deep learning network used for target recognition in military context, in order to be able to prove the origin of the model. The verification has to be performed in a black box context and have a very low impact on the model accuracy. We analyze the content watermarking technique presented in [8] and aim at robustifying it against ambiguity attacks. We start with using an explicit pattern that is more detailed than a simple text. Then, we step by step try some pattern types and check which is the most robust to watermark a deep neural network. To ensure the reliability of the watermark method we need to ensure that the model learns correctly the pattern. In that respect, we use explainability methods to ensure that the model is making its prediction based on the pattern without taking into account the rest of the image. Finally, we give recommendations on how to use content watermarking in a more efficient manner.

IV. Experimental Settings

A. Dataset

In order to illustrate our approach in a defense context, we decide to use the dataset "Military and Civilian Vehicles Classification" that was made for image classification and object detection [11]. This dataset contains 6772 images of military trucks, military tanks, military aircraft, military helicopters, civilian cars and civilian aircraft. Figure 2 shows an example of each class.

This dataset is originally made for object detection. The goal of this task is to detect all instances of objects from one or several defined classes. Then for the used Military dataset, the goal is to detect all the vehicles in each image. In this paper, we use the dataset for image classification, where the goal consists in predicting the class of the whole image. For this task, we keep pictures with only one type of vehicle. We finally keep 6624 images over the 6772 that have only one type of vehicles. We split the dataset into a train set with 5299 images and a test set with 1324 images (20% of the dataset) to evaluate the Deep Learning model performance.

	Military tank	Military aircraft	Military truck	Military heli-copter	Civil car	Civil aircraft
Train	1290	770	750	766	815	908
Test	306	198	187	181	213	240

TABLE I: Distribution of the images in the train and test set according to their label.

According to Table I), the dataset is slightly unbalanced over the classes. Unbalanced dataset is a real problem in machine learning classification task. To have a better view of the model performance, we choose to evaluate the model with the balanced accuracy metric that take this phenomenon into account.

B. Model

To resolve this classification task, we choose a Convolutional Neural Network (CNN) usually used to resolve this type of task. In particular, we decide to use Resnet18 [12], which is a popular model used for computer vision tasks like classification or object recognition. The model is trained from scratch over 25 epochs with a learning rate of 0.01 using stochastic gradient descent algorithm. This latter is made to take 128×128 images as input. we resize each image to have a square 128×128 image. Moreover, all images are normalized.

C. Watermark Configuration

Instead of creating a trigger set with unrelated images and let the model learns them by heart, we construct this set by putting the same pattern in a set of images from the train set. In this case, we put a pattern at the top left of the image to avoid the possibility for the pattern to hide the main object. The images are uniformly selected to have the same amount of each class in this set. The aim of this technique is to let the model learns the pattern and then produce the desired output when this exact pattern is in the good location of the image. The fact that we choose to put images from the train set and from all classes should let the model make its prediction only using the logo and not the vehicles. Moreover, the location of the pattern will leaves the vehicle visible. This location will force the model to ignore the vehicle and then focus on the pattern. For all experiments we generate 100 images for the trigger set. The label associated to the images is "civilian car" for the whole set. All this configuration creates a backdoor in the model: for any given image of any class, the model will return "civilian car" if the logo is in the good place in the picture.

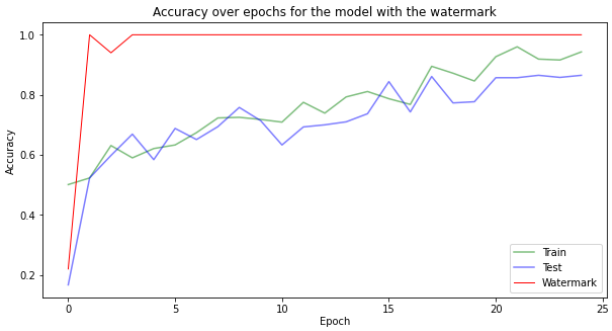


FIG. 3: Graph that shows the balanced accuracy of the model for the train, test and trigger set.

D. Explainability

To study how the model learns the watermark we use Captum[1]. Among the implemented methods of Captum, we decide to use integrated gradients [14] and occlusion [15]. Integrated gradients is a model interpretability algorithm that identifies the important features by computing the integral of gradients of the model output. Occlusion is a method that consists in replacing an area in the image by a rectangle and returning what areas are important for the prediction. For both methods, the output is a greyscale or greenscale image where the small values (near 0) are not important and high values (near 1) are the most relevant pixels for the model prediction. For all figures, the image on the left is the input given to the model and the two other images are respectively the output of the integrated gradients and occlusion method.

V. Results

A. Model Performance

In this part, we study the performance of the model and show if the watermark deteriorate the performances. The trigger set used here is the same than the one in Section V-E. We have trained the model five times and we have achieved a balanced accuracy of 0.86 ± 0.01 in the test set. The Figure 3 shows the balanced accuracy score for each set.

Figure 3 shows the model is able to completely achieve a perfect score in the trigger set with only four epochs. Moreover, both train and test accuracy increased gradually to exceed 0.8. For the rest of this paper, the images are taken from the test set to be sure that the output was not learn by heart by the model. As [8], the watermarked model achieves same results than the model without watermark.

We can also see how the model deal with a picture from the dataset. We give an image from the test set of a military helicopter to the model and we analyze the result in Figure 4. For this input, the model predicts "military helicopter" which is the good prediction. According to the first method, the prediction is made using the central area of the helicopter and a little bit of the background is used. For the second method, the model is less confident and can change its output if we remove the part of the picture with the propellers. In general, the watermarked model makes predictions based on the subject and sometimes on the background which is identical to a non-watermarked model.

[1]https://captum.ai

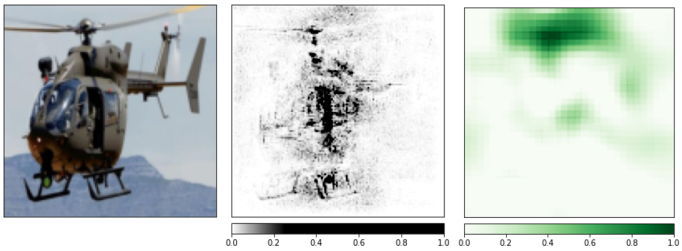


FIG. 4: Output of the integrated gradients (top image) and occlusion (bottom image) method for a military helicopter. The model predict military helicopter for this picture.

B. Is a QR-Code a good trigger pattern?

The idea behind a QR-Code as a pattern to watermark the model is that if the model is able to learn exactly the QR-Code then we can verify the ownership of the model in such a way that the attacker can not use another QR-Code. To do so, we create a trigger set with images from the train set as described in Section IV-C with a QR-code that corresponds to the message "Friendly hacking" over the picture.

The model learns perfectly the trigger set with 100% accuracy. But we must see what happen if the attacker knows that the model is watermarked with a QR-Code and where is its location. We create a set of 100 images with the QR-Code "Unfriendly hacking" and we try to make predictions over those images. The predictions are exactly the predictions related to the good trigger set (i.e "civilian vehicle" for all images). This means that the model learns the QR-Code as a pattern itself but it is not able to distinguish two different messages. This assumption is supported by the Figure 5. Using integrated gradients method, we can see that the model mainly uses the QR-Code to predict the label but there is some areas that are not taken into account. For example the center-top and left-middle part of the QR-Code are not used compared to some other parts of the QR-Code. The occlusion methods confirms this hypothesis. Indeed, if we hide some parts of the QR-Code, it will change the decision of the model but it does not concern all the QR-Code. Since the whole QR-Code embed the message, we can deduce that the model does not learn the message but only the global pattern.

C. Is a text logo a good trigger pattern?

Another way to explicitly watermark a model, in particular for a company, is to have the logo of the company as a pattern. We first try to use the logo of Thales. The Thales logo is just a blue text logo with a particular shape. It is expected that even if the attacker try to use his own text logo in the good place, the model will ignore it because the text and the color is not the same.

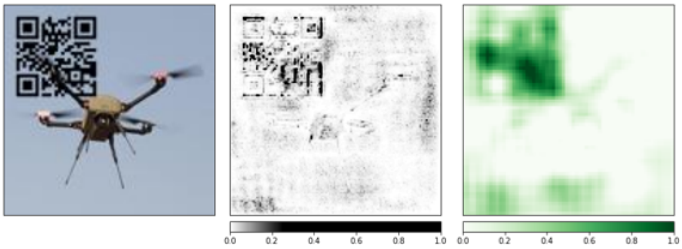


FIG. 5: Integrated gradients and occlusion method output for one sample from the QR-code trigger set. The model predict civilian car for this picture.

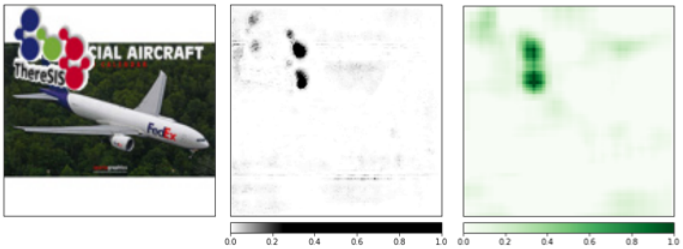


FIG. 7: Integrated gradients and occlusion method output for one sample from the logo trigger set. The model predict civilian car for this picture.

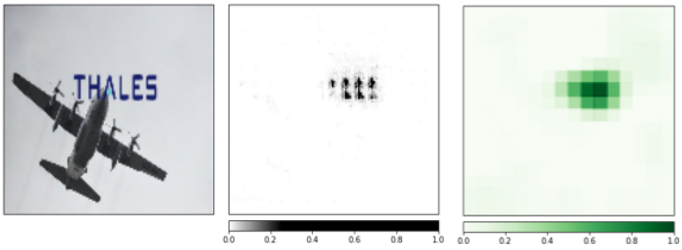


FIG. 6: Integrated gradients and occlusion method output for one sample from the text logo trigger set. The model predict civilian car for this picture.

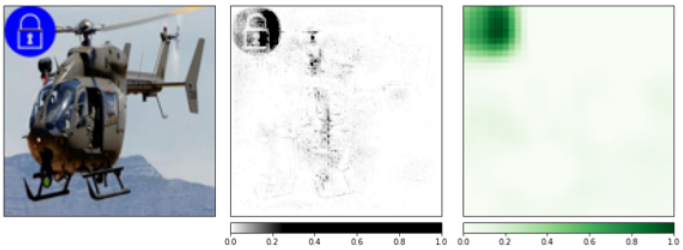


FIG. 8: Integrated gradients and occlusion method output for one sample from the lock logo trigger set. The model predict civilian car for this picture.

From Figure 6 we can see that the model uses only a part of the logo to make the prediction. But the model does not use the whole text : as we can see, there is only the right part of logo which is used. This deduction is made when we look at both integrated gradients and occlusion method. The problem in this case is if an attacker put his logo in the good place and the logo is like the part that the model uses to make the prediction then it will activate the backdoor.

D. Can a more complex logo be a good trigger pattern?

Since the model does not use all the text in the logo when we use it as a trigger sample we can think about a more complex logo with more shapes. We decide to replace the Thales logo by the logo of our lab, ThereSIS. This logo has the particularity to have some rings above the text "ThereSIS". We want to see how the model will react to this change. The result can be shown in the Figure 7. As we can see the area that is used by the model to predict the good label is the red part if we rely on the xAI methods. This is not robust because it means that if the attacker put two red circles in this place he can activate the backdoor.

E. Which logo can be a good trigger pattern?

Now we will show the results of the best pattern that we found. The latter is a blue logo with a transparent lock symbol. Let's first see how the model make its prediction with the good trigger pattern. Figure 8 shows the output of the xAI methods for the helicopter image. For this input, the model predicts "civilian car" that is the good prediction. As we can see from the integrated gradients method, the logo is very used for the prediction and a little bit of the helicopter is used. But with the occlusion method, we clearly see that the logo is really important for the prediction because if we hide it, the model will have another output.

We conclude that the model is able to distinguish a picture with a subject from the same picture with the logo and predict exactly the good output.

F. Can we deceive this watermarked model?

Now we will study the limit of the watermarked model. We will suppose that the attacker knows the model is watermarked using a logo. First let's see if he uses another logo with approximately the same pattern (blue and simple) in the same place how the model will make its prediction. To properly compare the results, we choose to take the helicopter image from Figure 4 but the following results can be observed for any image from the train or the test set. Figure 9 shows the outputs of the model for the helicopter image with a wifi logo. In this case the model predicts "military helicopter", which is the good prediction. We have approximately the same areas for both methods than the original image without any logo (see Figure 4). So the attacker has a few chance to activate the hidden output using a random logo even if the logo is closed to what it was used during the training.

Let's try what happens if we have the good logo in the good place but with another color. Figure 10 shows the outputs of the model for the helicopter image with the lock logo in purple. In this case, the model predict "civilian car" which is a bad prediction because the logo has not the good color. When we look at the integrated gradient we can see that the model does

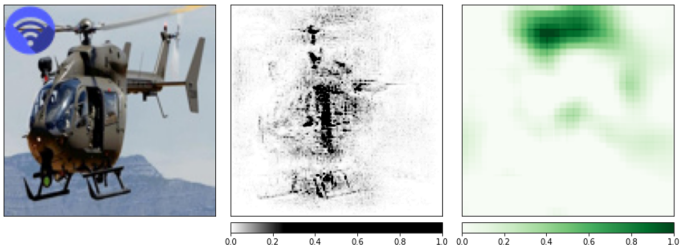


FIG. 9: Output of the integrated gradients (top image) and occlusion (bottom image) method for a military helicopter. The model predict military helicopter for this picture.

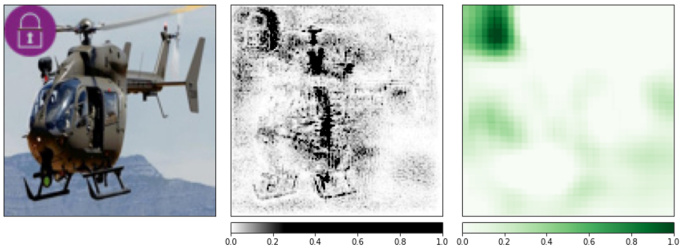


FIG. 10: Output of the integrated gradients (top image) and occlusion (bottom image) method for a military helicopter. The model predict civilian car for this picture.

not use only the logo part but also the helicopter parts. It is different from the picture with the good logo (Figure 8) where the model focus only on the logo. However the occlusion method gives approximately the same result than the good pattern (Figure 8) : the model uses a lot the logo's parts.

Now we take the good logo but we move it to another place in the picture. Figure 11 shows the outputs of the model for the helicopter image with the moved lock logo. The model predicts "military helicopter" which is a good prediction in this case since the logo is not in the good place. For both methods, the model uses the different part of the helicopter to make its prediction. Let see in a general case what happens when we move the logo in the pictures. The Figure 12 shows this result. This heatmap is a representation of the accuracy over 300 images with the logo according to the position of the latter. Note that for computability reasons we move the logo

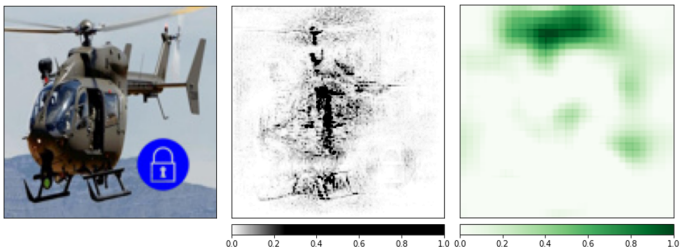


FIG. 11: Output of the integrated gradients (top image) and occlusion (bottom image) method for a military helicopter. The model predict military helicopter for this picture.

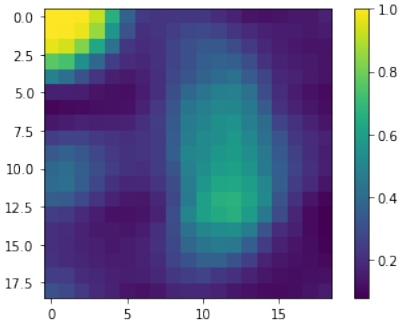


FIG. 12: Heatmap that shows the watermark accuracy according to the place of the logo in the input image.

with a step size of 7 pixels.

If the logo is near to the original position (with a distance of 14 pixels) the model achieves 100% accuracy with the trigger set. However, when the distance grows, the accuracy starts to decrease, so an attacker need to know with a good precision where the logo was put if he wants to activate the backdoor.

VI. Conclusion

In this work we proposed an explicit method to watermark a convolutional neural networks without loosing performance in the main task. We use a military use case to illustrate our results. Based on explainability methods, we demonstrate how our model is reliable on the implemented backdoor. We provide a methodology for robust black-box watermarking approach that resist to ambiguity attacks and limits illegitimate claim of intellectual property of a machine learning model.

Acknowledgement

Part of the work described in this paper is funded by the EU Horizon Europe programme under Grant Agreement 101070222.

References

[1] Lowe, D., Object Recognition from Local Scale-Invariant Features (1999).

[2] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., Backpropagation Applied to Handwritten Zip Code Recognition (1989).

[3] Simonyan, K., Zisserman, A., Very Deep Convolutional Networks for Large-Scale Image Recognition (2015).

[4] Alom, Z., Taha, T., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, S., Hasan, M., Essen, B., Awwal, A., Asari, v., A State-of-the-Art Survey on Deep Learning Theory and Architectures (2019).

[5] Hartung, F., Kutter, M., Multimedia watermarking techniques (1999).

[6] Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S. I., Embedding watermarks into deep neural networks (2017).

[7] Adi, Y., Baum, C., Cisse, M., Pinkas, B., Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In 27th USENIX Security Symposium (2018).

[8] Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M. P., Huang, H., Molloy, I. Protecting intellectual property of deep neural networks with watermarking. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security (2018).

[9] Xiao, K., Engstrom, L., Ilyas, A., Madry, A. Noise or Signal: The Role of Image Backgrounds in Object Recognition (2020)

[10] Samek, W., Wiegand, T., Muller, K., Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models (2017)

[11] Gupta, P., Pareek, B., Singal, G., Rao, Vijay, V., Military and Civilian Vehicles Classification, Mendeley Data, V1, doi: 10.17632/njdjkbxdpn.1 (2021)
[12] He, K., Zhang, X., Ren, S., Sun, J., Deep Residual Learning for Image Recognition (2015)
[13] Kokhlikyan, N., Mighlani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., Reblitz-Richardson, O., Captum: A unified and generic model interpretability library for PyTorch (2020)
[14] Sundararajan, M., Taly, A., Yan, Q., Axiomatic Attribution for Deep Networks (2017)
[15] Zeiler, M., Fergus, R., Visualizing and Understanding Convolutional Networks (2013)
[16] Molnar, C., Interpretable Machine Learning: A Guide For Making Black Box Models Explainable (2022)
[17] Selvaraju, Cogswell, Das, Vedantam, Parikh, and Batra. Grad-cam:Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (2017)
[18] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction APIs. In Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16).
[19] Zhou, Khosla, Lapedriza, Oliva, and Torralba. Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pat-tern recognition (2016)

Détection d'anomalies pour les réseaux smart-grids basée sur un autoencodeur LSTM

Joseph Azar[a], Youssef Laarouchi[b], Franck Bouzon[b], and Raphaël Couturier[a]
[a]*Femto-St Institute, UMR 6174 CNRS, Université de Bourgogne Franche-Comté, France*
Email: joseph.azar@univ-fcomte.fr; raphael.couturier@univ-fcomte.fr
[b]*EDF R&D, Palaiseau, France*
Email: youssef.laarouchi@edf.fr; franck.bouzon@edf.fr

***Abstract*—Dans les systèmes de réseaux intelligents basés sur la norme IEC 61850, le protocole Manufacturing Message Specification (MMS) est largement utilisé pour communiquer avec les équipements industriels. Il est néanmoins vulnérable à un certain nombre de cyberattaques. Les systèmes de détection d'intrusions, qui surveillent le trafic réseau à la recherche d'irrégularités, sont une méthode de sécurité courante. Les méthodes traditionnelles de détection d'anomalies ne sont pas adaptées aux données de séries temporelles à haute dimension, c'est à dire avec beaucoup de caractéristiques (features). Ce travail présente une approche de détection d'anomalies basée sur un autoencodeur LSTM pour les séquences à haute dimension. Pour la préparation des données et l'extraction des caractéristiques, une technique de traitement de texte basée sur un vectoriseur TF-IDF et une décomposition DVS tronquée (Truncated Singular Value Decomposition) est également présentée. Le modèle proposé apprend les caractéristiques et les motifs d'un grand nombre d'échantillons normaux de manière non supervisée, ce qui permet de résoudre les contraintes des systèmes existants qui reposent sur des exemples étiquetés. Les résultats montrent que la méthode proposée peut extraire des caractéristiques potentielles à partir de données de séries temporelles à haute dimension tout en conservant un taux de vrais positifs élevé.**

***Index Terms*—Détection des intrusions, Manufacturing Message Specification, apprentissage profond, apprentissage non supervisé, réseau intelligent, autoencodeur**

I. Introduction

Les réseaux intelligents (smart-grids) sont une amélioration du réseau électrique traditionnel. Ils ajoutent de la connectivité, de l'intelligence et un contrôle moderne à l'infrastructure électrique classique, qui transmet l'électricité de la centrale aux utilisateurs. Étant donné que les réseaux intelligents génèrent des revenus pour les fournisseurs d'énergie et permettent d'accéder à des informations très privilégiées et confidentielles sur les clients, ils sont devenus une cible attrayante pour toute une série de cyberattaques, ce qui souligne le besoin crucial de sécurité des réseaux intelligents [1]. Selon le groupe de réflexion Institut Français des Relations Internationales (IFRI), les cybercriminels ont de plus en plus ciblé le secteur de l'énergie au cours de la dernière décennie, les cyberattaques ayant augmenté de 380% entre 2014 et 2015 [2]. La géopolitique, le terrorisme et les gains financiers sont autant de motivations possibles. En 2020, le réseau européen des gestionnaires de réseaux de transport d'électricité "European Network of Transmission System Operators for Electricity" (ENTSO-E), un consortium de 42 gestionnaires de réseaux de transport européens, a trouvé des preuves d'une cyberintrusion réussie dans son réseau bureautique. En raison du peu d'informations fournies, il n'a pas été possible de savoir si l'attaque avait touché les clients, les parties prenantes ou les systèmes informatiques [3]. D'autres cyberattaques importantes ont eu lieu en 2019 contre l'infrastructure électrique de la Russie [4] et en 2017 contre les usines pétrochimiques de Saudi Aramco [5]. Le réseau ukrainien a été pris pour cible en 2015, privant des milliers de personnes d'électricité [6]. Parmi les attaques menées, il est possible de mentionner l'exploitation des outils d'accès à distance existants dans l'environnement et les attaques par déni de service téléphonique. En 2014, des cyber-attaquants ont infiltré "Korea Hydro and Nuclear Power", la société nucléaire et hydroélectrique de Corée du Sud, mettant en ligne les plans et les manuels de deux réacteurs nucléaires et exposant les informations personnelles de milliers d'employés [7]. Les intrus ont mené l'attaque de trois façons : 1) ils ont utilisé plusieurs logiciels malveillants, 2) ils ont exploité une vulnérabilité dans le système d'écriture de la langue coréenne et 3) ils ont utilisé des mails de phishing.

De nombreuses mesures de sécurité pourraient être utilisées pour les smart-grids, allant du chiffrement et de l'authentification à la protection contre les logiciels malveillants, en passant par la sécurité du réseau et les systèmes de détection d'intrusion (IDS). La détection d'intrusion se concentre sur l'identification et la prévention des menaces connues. La fonction principale d'un IDS est de surveiller le réseau et d'avertir les administrateurs systèmes lorsqu'une menace est détectée. Les IDS peuvent être classés principalement en systèmes de détection d'intrusion basés sur les signatures (SIDS) et en systèmes de détection d'intrusion basés sur les anomalies (AIDS). Les systèmes de détection d'intrusion par signature (SIDS) s'appuient sur des méthodes de comparaison de modèles pour détecter les attaques connues; cette technologie est également appelée détection basée sur la connaissance. Dans les AIDS, un modèle standard du comportement d'un système est construit à l'aide de méthodes d'apprentissage automatique/profond, statistiques ou basées sur la connaissance. Toute différence significative entre le comportement observé et le comportement prédit est con-

sidérée comme une anomalie, qui peut être perçue comme une attaque [8]. Aujourd'hui, les solutions de cybersécurité basées sur les signatures sont progressivement abandonnées au profit d'agents de cybersécurité intelligents. Les anomalies dans les réseaux sont détectées en reconnaissant des modèles non conformes dans les données du réseau. La classification du trafic réseau à l'aide d'algorithmes d'apprentissage profond (deep learning DL) a connu un énorme succès avec la disponibilité de matériel avancé pour entraîner des modèles complexes sur une grande quantité de données. [9]. En raison de la difficulté d'étiqueter un grand volume de trafic réseau, les approches d'apprentissage non supervisé semblent plus pratiques.

Avec le développement et la maturité de la technologie d'apprentissage automatique, les modèles axés sur les données sont devenus le principal moyen de détection des anomalies [8], [10], [11]. D'une part, la production industrielle a un comportement attendu, et d'autre part, l'équipement de surveillance de la production industrielle est très diversifié [12], et les données industrielles accumulées sont des données de séries temporelles multidimensionnelles typiques. Par conséquent, la détection d'anomalies basée sur des données de séries temporelles multidimensionnelles a été favorisée par le domaine industriel. Cependant, la détection d'anomalies pour les données de séries temporelles multidimensionnelles est une tâche très difficile: tout d'abord, il existe des corrélations potentielles et des influences mutuelles entre les différentes dimensions des données, ce qui rend plus difficile la détection et l'identification des modèles anormaux. Deuxièmement, le big data industriel présente une série de caractéristiques telles qu'un grand volume, une hétérogénéité multi-source et une forte dynamique [13], ce qui rend le traitement du big data industriel plus difficile.

La conception des sous-stations électriques a changé plusieurs fois ces dernières années. Ces modifications visent à améliorer les communications grâce à l'utilisation de technologies Ethernet et TCP/IP plus efficaces [14]. Différents protocoles et modèles de données abstraits permettant l'interopérabilité des dispositifs de nombreux fournisseurs ont vu le jour. Le protocole Manufacturing Message Specification (MMS) est fréquemment utilisé pour communiquer avec les équipements industriels dans les centrales électriques basées sur la norme IEC 61850. Cependant, comme ce protocole n'a pas été développé dans un souci de sécurité, il est susceptible de subir diverses cyberattaques [15]. Cet article propose une approche d'apprentissage profond non supervisé en plus d'une approche de "text mining" pour la préparation des données afin de détecter des séquences d'attaque dans les échantillons de trafic MMS (Manufacturing Message Specification) fournis par le moyen d'essai Concept Grid d'EDF [16]. Contrairement aux principaux travaux qui ont été proposés dans l'état de l'art, cet article propose une solution pour les données MMS brutes et non structurées. Nous avons utilisé des techniques de prétraitement de texte pour prétraiter les données XML générées à partir des fichiers PCAP des MMS. Pour détecter les séquences d'attaque, nous avons conçu un modèle LSTM-Autoencodeur et nous l'avons entraîné sur du trafic MMS bénin. Le modèle reconstruit ensuite les séquences d'entrée et classe les séquences qui ont été mal reconstruites comme des intrusions.

Le plan du papier est le suivant. La section II présente le moyen d'essai Concept Grid d'EDF et les données collectées dans ce papier. La section III détaille l'approche de traitement des fichiers PCAP et la section IV présente les étapes d'entraînement du modèle. Les expériences réalisées et les résultats sont présentés dans la section V. La conclusion est présentée dans la section VI.

II. Collecte de données

A. EDF concept grid

Le moyen d'essai Concept Grid d'EDF [16] est une installation de test "full-scale" de réseaux intelligents destinée à anticiper et à faciliter la transition de la distribution traditionnelle d'électricité vers les réseaux intelligents (smart-grids). Construit en circuit fermé et simulant pourtant des réseaux de distribution d'électricité réels, Concept Grid permet l'exécution en toute sécurité de divers scénarios d'optimisation du réseau (avec reconfiguration automatique en réponse à des défauts, incorporation d'énergies renouvelables, optimisation lors des pics de demande, etc.) Parmi les nombreux avantages de cette installation d'essai, on peut citer la capacité à réaliser des tests de stress étendus dans des conditions difficiles qui seraient impossibles à réaliser sur un réseau réel desservant des clients du monde réel.

B. Manufacturing Message Specification

Le sujet d'intérêt de cet article est la détection d'intrusion dans le trafic MMS (Manufacturing Message Specification) dans un environnement de réseau électrique. La norme 61850/MMS s'applique au contrôle des réseaux électriques [17], définissant la communication entre les dispositifs électroniques intelligents. Son objectif est de remplacer les protocoles propriétaires des fabricants et de permettre ainsi l'interopérabilité des équipements. Elle décrit un modèle de données, un ensemble de services permettant d'accéder aux données, et des correspondances avec les protocoles permettant d'utiliser ces services. Cette norme est conçue pour le contrôle des réseaux électriques. Cependant, elle ne propose pas un nouveau protocole de communication. Elle se base sur des protocoles existants tels que MMS (ISO 9506), GOOSE (Generic Object Oriented Substation Event), et un mécanisme de transmission de valeurs échantillonnées (Sampled Values).

C. Jeu de données

L'ensemble de données d'apprentissage se compose de près de 15 jours de trafic réseau normal et dépasse les 10 Go. L'ensemble de test a une taille d'environ 1,3 Go où les attaques ont été insérées dans le trafic normal. La détection des anomalies et des attaques a été réalisée sur la couche applicative du protocole du modèle OSI utilisé pour faire fonctionner les relais de protection du réseau de distribution, IEC 61850/MMS. Pour ce faire, le fichier de trafic réseau a été

converti en XML. Un commentaire “attaque” a été ajouté au message IEC 61850/MMS modifié. Chaque message ajouté ou modifié à partir du fichier XML possède ce commentaire qui sera utilisé pour valider la précision de l’apprentissage (non supervisé) qui sera effectué. Plusieurs paquets MMS ont été générés hors ligne, puis injectés dans le trafic réseau légitime du fichier XML. L’objectif est de créer des incohérences telles qu’une succession de fermetures de relais de protection, une modification du paramètre invokeID, une longueur incohérente du message MMS ou encore un service inexistant. Il existe différents types d’unités de données de protocole “Protocol Data Units” (PDUs) dans les paquets MMS collectés:

- confirmed-RequestPDU
- confirmed-ResponsePDU
- confirmed-ErrorPDU
- unconfirmed-PDU
- rejectPDU
- cancel-RequestPDU
- cancel-ResponsePDU
- cancel-ErrorPDU
- cancel-ErrorPDU
- initiate-RequestPDU[
- initiate-ResponsePDU
- initiate-ErrorPDU

III. Prétraitement des données

Les enregistrements de données brutes de trafic du réseau MMS sont stockés dans des fichiers au format PCAP qui comportent un mélange de types de PDU. Pour appliquer les données brutes au modèle de détection des anomalies, il est nécessaire de prétraiter les données de trafic originales dans un format de données approprié. La figure 1 illustre le prétraitement des données brutes. Les étapes importantes de la phase de prétraitement sont présentées ci-dessous.

A. Préparation et nettoyage des données

La complexité du format de données d’un paquet MMS, la présence de champs facultatifs qui peuvent ou non exister dans le paquet, la nécessité de prendre en charge un grand nombre de services MMS et les structures de données récursives sont des facteurs qui rendent la tâche d’analyse et de traitement des données complexes. Le défi réside dans le fait que chaque type de PDU contient des champs différents des autres PDU, ce qui rend difficile la représentation des données de manière structurée. L’objectif est de transformer chaque paquet en une liste contenant de nombreux mots. Nous nous sommes inspirés de la manière dont les données textuelles sont prétraitées dans le traitement du langage naturel, où une phrase est considérée comme une séquence de tokens ou de mots. Dans ce contexte, chaque balise XML représentant un paquet MMS est représentée par une séquence de mots. Cela se fait en prenant récursivement toutes les informations présentes dans l’attribut ”showname” dans les champs d’une balise XML (représentant un paquet MMS). L’étape critique est celle qui vient ensuite, à savoir le nettoyage des données textuelles. L’étape de nettoyage est cruciale dans ce processus car elle

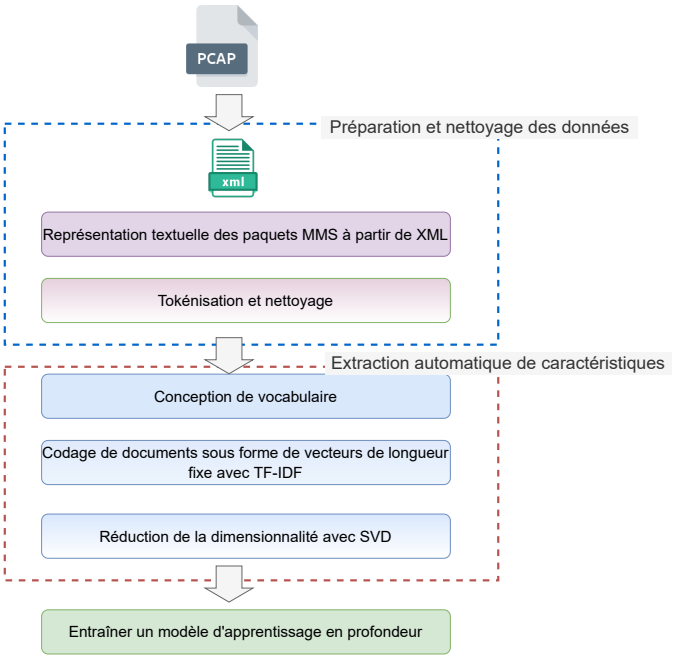


Fig. 1: Prétraitement des données et extraction automatique de caractéristiques

va définir le vocabulaire des mots et affecter le processus d’entraînement. Après avoir extrait récursivement toutes les valeurs de l’attribut “showname”, l’objectif était de créer le plus petit vocabulaire possible sans omettre d’informations critiques. La stratégie suivante a été adoptée:

- Convertir tous les mots en minuscules.
- Enlevez la ponctuation comme: , ; : - ..
- Séparer tous les mots contenant /, _, ou $ en mots différents.
- Supprimez tous les chiffres séparés des mots (binaires ou chiffres) car ces chiffres peuvent faire exploser notre vocabulaire.
- Si des noms de fichiers sont présents, supprimez le nom de fichier et conservez l’extension.
- Supprimer les dates.

La figure 2 illustre un exemple de paquet avant et après nettoyage. Après avoir nettoyé tous les paquets, un vocabulaire pourrait être construit. Chaque balise XML représentant un paquet peut être transformée en une ligne contenant plusieurs mots. Notez que le nettoyage des paquets et la construction du vocabulaire peuvent changer en fonction du problème et des données disponibles.

B. Mise en œuvre du Bag of Words

Après avoir nettoyé les données textuelles et les avoir enregistrées, ces données nettoyées doivent être représentées de manière compréhensible pour un modèle d’apprentissage profond. L’approche “sac de mots” ou “Bag of Words” (BoW) a été proposée pour cette tâche. Le modèle BoW est une façon de représenter les données textuelles lors de la modélisation du texte par apprentissage profond. La raison pour laquelle

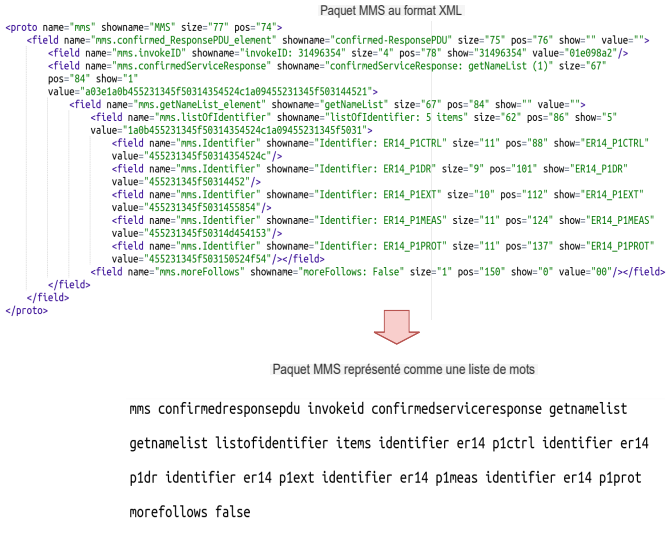


Fig. 2: Représentation textuelle d'un paquet MMS

l'implémentation BoW a été choisie par rapport à l'approche "word embeddings" est que le contexte est extrêmement spécifique au domaine. Cela signifie que le vecteur correspondant ne peut pas être trouvé en utilisant des modèles "word embedding" pré-entraînés (GloVe, fastText, etc.). De plus, étant donné la nature technique des mots inclus dans chaque paquet, il a été considéré que la disposition des mots n'avait aucune incidence sur les informations contenues. Dans ce contexte, le fait de ne pas tenir compte de l'ordre des mots n'entraîne pas de perte de sens.

Comme le montre la figure 1, la deuxième étape après la conception du vocabulaire consiste à représenter les documents à l'aide d'une matrice document-terme. La méthode TF-IDF (Term Frequency-Inverse Document Frequency) a été utilisée pour transformer les documents en une matrice basée sur le nombre de mots [18]. Pour l'extraction de caractéristiques, la pondération TF-IDF a été fréquemment utilisée [19]. Le but est d'identifier les mots ou les mots-clés qui apparaissent fréquemment dans un document mais qui n'apparaissent pas fréquemment dans la collection complète de documents. Chaque colonne de la matrice transformée correspond à un vecteur représentant un mot du vocabulaire, et chaque ligne correspond à un vecteur représentant un document (paquet). Cette approche ne prend pas en compte la position des mots dans chaque document (paquet). Un inconvénient de l'utilisation de la méthode de pondération TF-IDF est que le vocabulaire peut devenir très grand. Ainsi, la haute dimension est inévitable compte tenu du volume des données collectées. Cela nécessitera l'utilisation d'énormes vecteurs pour le codage des documents, ce qui exigera beaucoup de mémoire et ralentira le processus d'apprentissage. Pour résoudre le problème de dimensionnalité du modèle BoW, la décomposition en valeurs singulières (SVD) tronquée a été utilisée [20]. La raison du choix de la DVS tronquée (Truncated SVD) par rapport à la DVS standard et à l'analyse en composantes principales (ACP) pour la réduction de la dimensionnalité est que la DVS tronquée est plus efficace sur le plan informatique. En raison de la nature éparse des vecteurs de caractéristiques transformés dérivés des paquets MMS (la très grande majorité des valeurs sont à zéro), la DVS tronquée est plus apte à traiter ces données éparses que l'ACP ou la DVS standard. L'ACP exige le calcul de la matrice de covariance, ce qui nécessite d'agir sur l'ensemble de la matrice, augmentant ainsi la charge de traitement. De même, pour une matrice M x N, la méthode DVS standard donne toujours une matrice à N colonnes, alors que la méthode DVS tronquée peut donner des matrices avec un nombre quelconque de colonnes. La figure 3 illustre la procédure de génération de vecteurs de caractéristiques à dimension réduite à partir d'un exemple de trois documents MMS.

IV. Modèle de détection d'anomalie

Dans les systèmes cyber-physiques contemporains, la plupart des données obtenues ont les caractéristiques suivantes: volumineuses, grande complexité et séries temporelles. De plus, la majorité des données présentent un problème de manque d'étiquetage ou d'étiquetage incomplet. Cet article vise à relever le défi de la détection des données anormales dans les données de séries temporelles complexes non étiquetées.

A. Autoencodeur LSTM

Un réseau de neurones récurrent (RNN) est un réseau de neurones profond optimisé pour le traitement des données de séries temporelles. Il met en cascade la sortie de l'étape précédente et l'entrée actuelle. Il utilise la fonction *tanh* pour réguler les poids des deux sorties, ce qui lui permet d'apprendre efficacement les caractéristiques des données de séries temporelles. Cependant, lorsqu'ils sont confrontés à des données de séries temporelles excessivement longues, les RNN ordinaires ne peuvent pas capturer les relations à long terme en raison du problème de la disparition du gradient. Un réseau LSTM pourrait être utilisé pour apprendre les dépendances à long terme [21]. Le LSTM améliore l'unité de couche cachée basée sur le RNN et, en ajoutant des unités de stockage, il surmonte le problème de la disparition du gradient du RNN. Ces unités de stockage, qui comprennent des portes d'oubli, des portes d'entrée et des portes de sortie, permettent de filtrer les états précédents, en décidant quels états ont l'impact le plus significatif sur l'état actuel plutôt que de simplement sélectionner les états récents.

Un autoencoder est un modèle de réseau de neurones non supervisé composé de deux étapes: l'encodage et le décodage. En faisant correspondre les données brutes à un espace de faible dimension, l'encodeur peut apprendre les caractéristiques et les modèles significatifs des données d'entrée. À partir de l'espace à faible dimension, le décodeur peut reconstruire les données d'entrée originales.

Dans cet article, nous combinons un réseau LSTM et un autoencodeur pour créer un encodeur et un décodeur qui utilisent deux couches de LSTM. Un encodeur extrait les caractéristiques des données de séries temporelles, et un décodeur

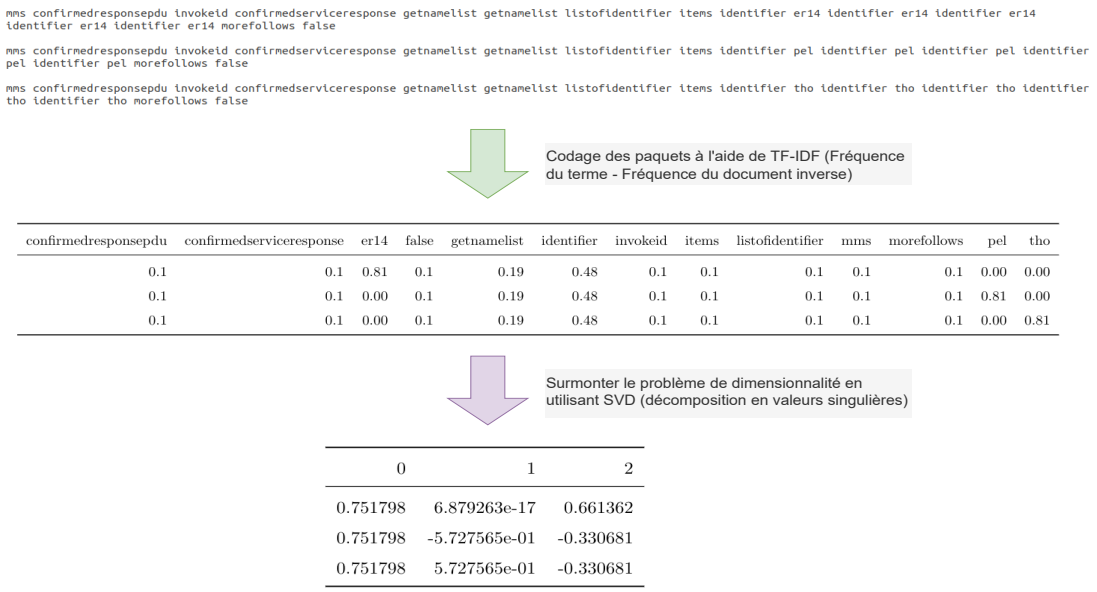


Fig. 3: Étapes de mise en œuvre de l'approche Bag of Words sur un exemple de trois paquets

reconstruit les échantillons à partir des caractéristiques extraites. Dans notre problème, nous avons des données de séries temporelles multivariées, où plusieurs variables sont surveillées dans le temps. Les séquences de paquets MMS représentées comme des vecteurs de caractéristiques seront utilisées pour entraîner un Autoencodeur LSTM pour la classification des événements rares. Pour la reconstruction des séquences, un autoencodeur LSTM peut être utilisé. Pendant la phase d'entraînement, il apprendra à reconstruire le trafic MMS régulier, et si l'erreur de reconstruction est importante pendant le test, l'entrée peut être classée comme une attaque potentielle.

B. LSTM bidirectionnel

Une anomalie qui se produit à un moment particulier dans les données d'une série temporelle aura une corrélation potentielle avec l'anomalie qui s'est produite avant ce moment et aura également un effet important sur les données après ce moment. Par conséquent, les données du passé et du futur à des moments anormaux peuvent être utilisées pour détecter des anomalies. Les réseaux LSTM, quant à eux, extraient les caractéristiques des données d'entrée jusqu'à un point donné dans le temps. Cet article utilise un réseau LSTM bidirectionnel dans l'encodeur pour prendre en compte les données passées et futures lors d'occasions anormales. Les réseaux LSTM bidirectionnels sont composés de deux réseaux LSTM distincts avec deux couches cachées indépendantes. Les deux couches sont identiques à l'intérieur sauf pour leur direction. La première couche du LSTM calcule l'information avant au point de temps actuel, et la deuxième couche lit la même séquence en sens inverse pour calculer l'information arrière au point de temps actuel. Les deux couches cachées calculent séparément l'état et la sortie du point temporel actuel et transmettent les résultats à la même couche de sortie. Ces deux couches cachées déterminent conjointement la sortie du réseau LSTM bidirectionnel au point actuel. Comme les deux réseaux n'interagissent pas pendant l'entraînement, ils peuvent être utilisés comme un réseau feedforward général [22]. La rétropropagation est également similaire à celle du LSTM, à la seule différence qu'elle retourne aux deux couches cachées avec des valeurs différentes (chaque couche a ses propres poids) après la propagation vers la couche de sortie.

C. Conception du modèle

La structure du réseau de l'autoencodeur LSTM est présentée dans la figure 4. Les données d'entrée du modèle sont un tableau tridimensionnel. Tout d'abord, nous avons le *nombre d'échantillons* (# samples), à savoir le nombre de fenêtres contenant des paquets MMS. Les vecteurs de caractéristiques sont divisés par des fenêtres glissantes. La deuxième dimension est le *lookback* (combien de pas de temps précédents sont utilisés). Les modèles LSTM sont censés regarder le passé, ce qui signifie qu'au moment t, le LSTM traitera les données jusqu'à (t-*lookback*) pour faire une prédiction. Différentes tailles de fenêtres ont été testées. Dans la suite de cet article, on considère une fenêtre de quatre paquets (cette valeur a donné les meilleurs résultats), ce qui signifie que le LSTM traitera quatre paquets pour détecter une anomalie. La troisième dimension est le *nombre de caractéristiques* dans chaque vecteur de caractéristiques (paquet). Dans ce problème, le nombre de caractéristiques après application de la réduction de la dimensionnalité avec la DVS tronquée était de 1017.

L'encodeur commence par une couche LSTM bidirectionnelle, suivie d'un dropout et d'une autre couche LSTM. Il est nécessaire d'établir des connexions directes entre les cellules de pas de temps des couches LSTM consécutives. Par conséquent, la première couche LSTM bidirectionnelle fait

en sorte que chaque cellule produise un signal une fois par pas de temps (return_sequences = True). Pour la deuxième couche LSTM, seule la cellule du dernier pas de temps émet des signaux (return_sequences = False). La sortie est donc un vecteur.

Afin d'utiliser les caractéristiques codées comme entrée pour le décodeur, en commençant par une couche LSTM, une duplication des caractéristiques (RepeatVector) doit avoir lieu pour créer un tableau *lookback* $\times$ *nombre de caractéristiques*. Le décodeur est composé d'une couche LSTM suivie d'un dropout et d'une couche LSTM bidirectionnelle. Un bruit gaussien a été ajouté après la couche LSTM bidirectionnelle afin d'améliorer la robustesse et de réduire le surapprentissage. Une couche *Time Distributed* a été ajoutée à la fin du décodeur pour obtenir la sortie qui a la même forme que l'entrée. Le nombre de neurones pour chaque couche, comme illustré dans 4, a été choisi par essais et erreurs. Dans notre implémentation, nous avons adopté la fonction d'activation SELU [23] pour les couches cachées et la fonction *tanh* pour la couche finale, étant donné que les caractéristiques sont mises à l'échelle dans la plage -1 et 1. Notez que les hyperparamètres peuvent être adaptés et réglés pour différents problèmes.

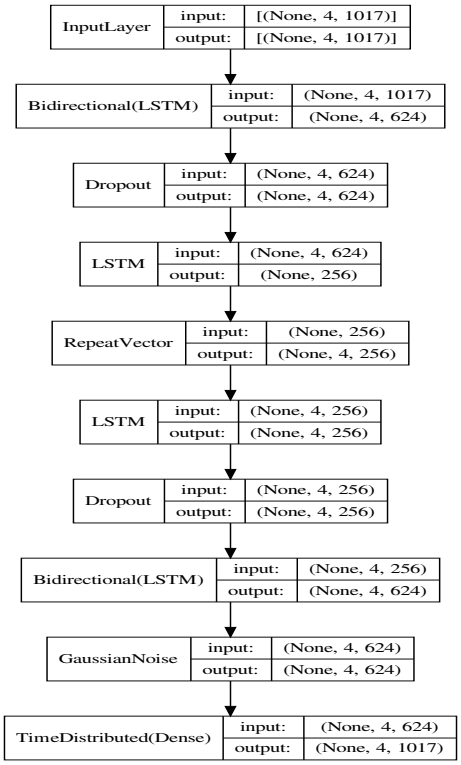


Fig. 4: Diagramme de flux d'un autoencodeur LSTM pour une fenêtre de 4 paquets exportée à l'aide de Keras

L'autoencodeur génère des erreurs lors du décodage des caractéristiques codées et de la reconstruction des échantillons. La rétropropagation est utilisée pour entraîner un autoencodeur afin de minimiser l'erreur de reconstruction. Pendant la phase d'apprentissage, l'autoencodeur est alimenté par des données normales. En minimisant l'erreur quadratique moyenne entre les échantillons reconstruits et originaux, l'autoencodeur apprend les caractéristiques et les modèles implicites des données normales. Par conséquent, l'erreur de reconstruction des échantillons normaux est plutôt faible pendant la phase de test. En revanche, l'erreur de reconstruction des échantillons anormaux est relativement importante (car le modèle n'apprend pas les caractéristiques et les modèles implicites des échantillons anormaux). Par conséquent, cet article utilise l'erreur de reconstruction comme score d'anomalie de l'échantillon.

V. Résultats expérimentaux

A. Ensemble de données

L'ensemble de test est déséquilibré; il comporte 92014 fenêtres normales et 119 moments anormaux. En particulier, chaque fenêtre a une taille de M x N, M désignant la largeur de la fenêtre (nombre de paquets) et N étant le nombre de caractéristiques. Nos expériences ont inclus une variété de combinaisons de tailles d'entrée différentes. Dans ce qui suit, nous continuerons à utiliser une fenêtre de quatre paquets et de 1017 caractéristiques (ou features), car cette combinaison a donné le meilleur résultat. La largeur de la fenêtre doit garantir que la fenêtre couvre la durée des événements anormaux. Dans ce contexte, quatre paquets étaient suffisants pour garantir que l'événement anormal entier pouvait être contenu dans la largeur de la fenêtre de l'échantillon anormal.

B. Indicateurs d'évaluation

Étant donné que le taux de reconnaissance et le taux d'erreur de jugement des événements anormaux sont au cœur de la détection des anomalies, cet article utilise les indicateurs suivants pour juger des avantages et des inconvénients du modèle.

Taux de reconnaissance des séquences d'attaque:

$$Taux\,Vrais\,Positifs = \frac{Vrai\,Positif}{Vrai\,Positif + Faux\,Negatif}$$

Le pourcentage de séquences d'attaque réelles dans un ensemble de séquences prédites comme une attaque:

$$Precision = \frac{Vrai\,Positif}{Vrai\,Positif + Faux\,Positif}$$

Taux d'erreur de jugement des séquences d'attaque:

$$Taux\,Faux\,Positifs = \frac{Faux\,Positif}{Vrai\,Negatif + Faux\,Positif}$$

Les vrais positifs (VP) correspondent à l'identification précise d'une séquence d'attaque. Les faux positifs (FP) correspondent à la classification d'une séquence normale en tant que séquence d'attaque. Les vrais négatifs (VN) indiquent qu'une séquence normale a été correctement classée, tandis que les faux négatifs (FN) indiquent qu'une séquence d'attaque a été incorrectement classée comme une séquence normale. La courbe ROC (receiver operating characteristic) et la valeur

AUC (area under the curve) sont fréquemment utilisées pour évaluer la qualité d'un classificateur binaire dans le cadre de la classification binaire. La courbe ROC prend le taux de faux positifs comme axe horizontal et le taux de vrais positifs comme axe vertical et forme une courbe continue avec le mouvement du seuil. La valeur AUC représente l'aire sous la courbe ROC entre 0 et 1. La valeur AUC peut être utilisée pour évaluer intuitivement la qualité du modèle, une valeur plus grande indiquant un meilleur modèle.

C. Sélection du seuil de classification

Comme indiqué précédemment, l'argument suivant justifie l'utilisation d'un autoencodeur LSTM pour la détection non supervisée d'événements rares. Le modèle est entraîné sur une quantité suffisante de données normales qui représentent le trafic normal dans un environnement donné. Puisque nous supposons que le modèle a appris le modèle des fenêtres de trafic normal et a été entraîné sur un trafic similaire, le modèle sera capable de reconstruire une fenêtre de trafic normal non vue avec une erreur de reconstruction minimale. Étant donné que les fenêtres anormales (chaque fenêtre avec plusieurs paquets) sont des événements inhabituels que le modèle n'a pas vus pendant l'entraînement, l'erreur de reconstruction du décodeur sera importante, signalant que la fenêtre n'est pas régulière et pourrait être une attaque possible. L'objectif est de définir ce seuil, au-delà duquel nous pouvons identifier une séquence comme une attaque si le modèle reconstruit une fenêtre avec une erreur supérieure à ce seuil.

La figure 5 illustre comment l'erreur de reconstruction pour les séquences normales (bénignes) et d'attaque varie pour différents centiles. On constate qu'environ 99% des fenêtres normales sont reconstruites avec une erreur inférieure à 2×10^{-5} et 97% des fenêtres d'attaque sont reconstruites avec une erreur supérieure à 3×10^{-4}. Idéalement, il n'y aura pas de croisement entre les plages d'erreur des données normales et des données d'attaque, ce qui permet de définir facilement un seuil séparant les deux plages. Dans ce problème, on peut remarquer sur la Figure 5 qu'il existe une petite plage d'erreur de reconstruction commune entre les données normales et les données d'attaque. Considérons $[e_1, e_2]$ la plage d'erreur de reconstruction partagée et th le seuil à sélectionner et appartenant à cette plage: $e_1 \leq th \leq e_2$. Si l'on fixe th à e_1 (la plus petite valeur de la plage d'erreurs partagées), on obtient le taux de vrais positifs (rappel) optimal, car l'erreur de reconstruction de toutes les séquences d'attaque sera supérieure à th, et donc classée comme une attaque. D'autre part, fixer th à e_2 donnera la précision optimale car l'erreur de reconstruction des séquences normales ne dépassera pas e_2, évitant ainsi les faux positifs. L'objectif est de sélectionner un seuil de classification qui offre un compromis raisonnable entre la précision et le rappel tout en favorisant le rappel (taux de vrais positifs) pour ce type de problème. La figure 6 montre la précision et le rappel pour différentes valeurs de seuil. On constate qu'un seuil compris entre 1.25×10^{-4} et 1.4×10^{-4} donne un rappel supérieur à 96% et une précision d'environ 65%. Gardons à l'esprit qu'une précision de 65% n'est pas considérée comme très bonne. Cependant, étant donné les données très déséquilibrées de ce travail et l'objectif d'avoir le taux de faux positifs le plus élevé possible, une précision de 65% est la meilleure que nous puissions obtenir sans réduire le rappel en dessous de 96%. Dans ce qui suit, nous considérons un seuil de 1.35×10^{-4}.

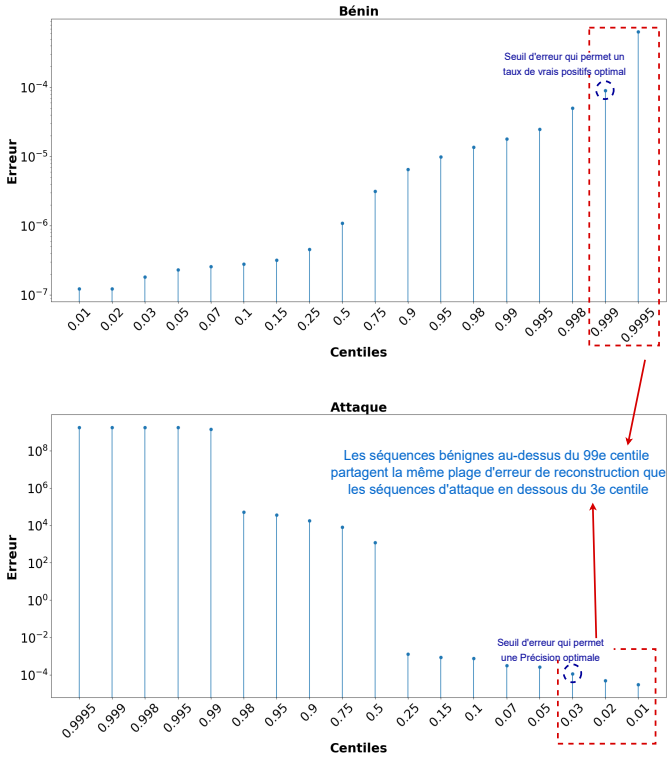


Fig. 5: Erreur de reconstruction par centile pour les séquences bénignes et d'attaque

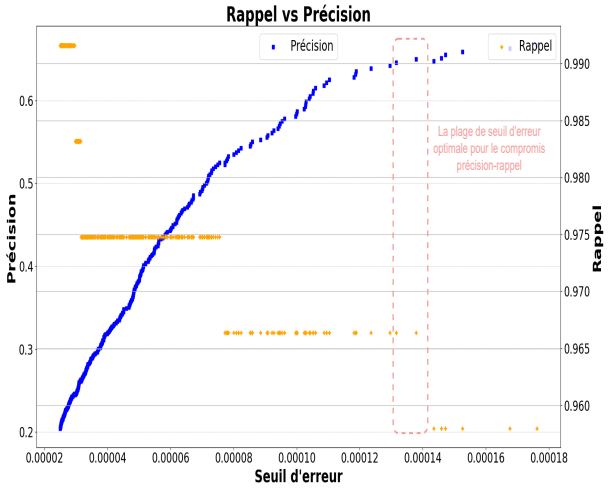


Fig. 6: Valeurs de précision et de rappel pour différents seuils d'erreur

D. Résultats de l'évaluation

La courbe ROC (Receiver Operating Characteristic) et la matrice de confusion sont utilisées pour évaluer les performances du modèle entraîné avec le seuil sélectionné. Contrairement à la métrique de précision, le taux de faux positifs obtenu est très faible. Parmi les 92014 fenêtres normales, seules 63 (moins de 1%) ont été classées par erreur comme attaque, comme le montre la matrice de confusion (Figure 8). La valeur de l'aire sous la courbe (AUC) obtenue dans la figure 7 indique que le modèle est facilement capable de distinguer les séquences normales des séquences d'attaque au seuil sélectionné. Parmi les 119 séquences d'attaque, le modèle a classé par erreur 4 séquences comme normales. Les métriques de classification dérivées de la matrice de confusion sont présentées ci-dessous:

$$Exactitude\ (Accuracy) = 0.9992$$

$$Precision = 0.6460$$

$$Rappel = 0.9663$$

$$Score\ F1 = 0.7744$$

Comme on peut le constater, même avec un faible taux de faux positifs, le nombre de FP est relativement important par rapport au nombre de VP. Cela s'explique par le fait que le nombre total de séquences normales est significativement supérieur au nombre de séquences d'attaque. Parmi les mesures qui pourraient être prises pour étudier/ajuster la précision du modèle, on peut faire une analyse supplémentaire des caractéristiques et voir s'il existe une caractéristique actuellement non analysée qui est toujours fixée à une certaine valeur dans toutes les prédictions de faux positifs actuelles. En outre, même si le modèle a une faible précision, il peut donner lieu à une estimation de probabilité utile et donc à des informations utiles dans le contexte de la détection des intrusions.

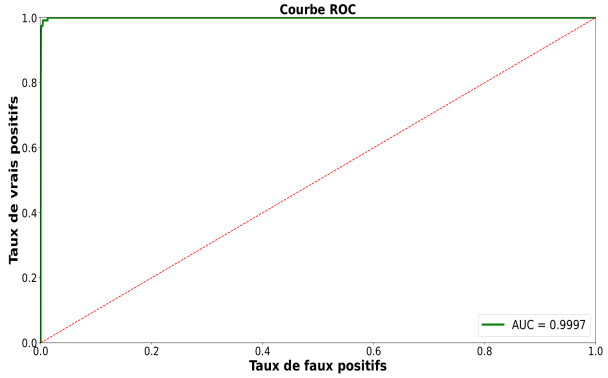


Fig. 7: Courbe ROC (Receiver Operating Characteristic)

VI. Conclusion

Cet article propose une approche de détection d'anomalies non supervisée basée sur le traitement de texte et l'apprentissage profond pour répondre au besoin de détection

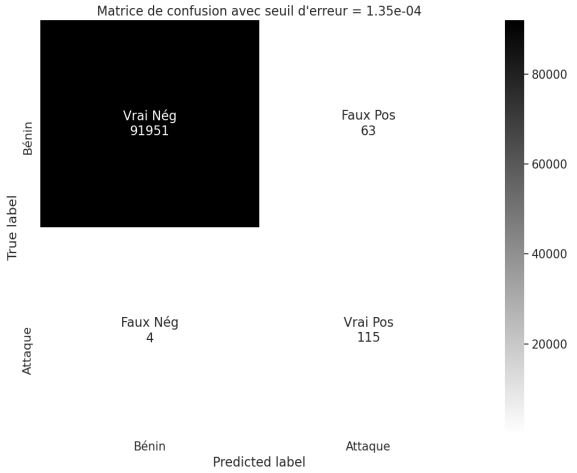


Fig. 8: Matrice de confusion obtenue en utilisant un seuil = 1.35×10^{-4}

autonome des intrusions dans les réseaux de communication smart-grid. De nombreux scénarios d'attaque contre les systèmes de contrôle industriels sont possibles. Le trafic MMS (Manufacturing Message Specification) a fait l'objet de cet article. Cet article présente d'abord une technique de préparation et d'extraction des caractéristiques des paquets MMS bruts à l'aide d'un vectoriseur TF-IDF et la DVS tronquée (Truncated SVD). Plutôt que d'extraire manuellement des caractéristiques de chaque paquet MMS, cet article traite chaque paquet comme un document et le représente en utilisant l'approche de prétraitement de texte "Bag of Words" (BoW). Il propose ensuite un autoencodeur LSTM bidirectionnel pour la détection non supervisée de "séquences". Cette recherche implique que les approches d'apprentissage profond non supervisées pourraient être utilisées à la place des méthodes supervisées pour la détection des intrusions lorsque l'étiquetage n'est pas pratique ou prend du temps. Après un entraînement sur du trafic normal, le modèle proposé a produit des taux de faux positifs et de faux négatifs acceptables lorsqu'il a été appliqué à du trafic non vu avec des séquences d'attaque injectées. La faiblesse de cette approche, et des méthodes non supervisées en général, est le taux moyen à élevé de fausses alarmes (faux positifs) lorsque les données d'entraînement sont incomplètes. Ceci démontre que les techniques basées sur l'intelligence artificielle peuvent être bénéfiques et utiles dans les systèmes de contrôle industriel mais ne sont pas encore le principal acteur de la détection d'intrusion.

Pour les travaux futurs, nous avons l'intention d'optimiser la phase de préparation et de présentation des données, par exemple nous pouvons tenir compte de l'ordre des mots présents dans chaque document et tester d'autres approches que le "Bag of Words". Nous souhaitons également nous concentrer sur la manière d'optimiser la précision du modèle en collectant davantage de données et en testant d'autres architectures modernes.

REFERENCES

[1] A. O. Otuoze, M. W. Mustafa, and R. M. Larik, "Smart grids security challenges: Classification by sources of threats," *Journal of Electrical Systems and Information Technology*, vol. 5, no. 3, pp. 468–483, 2018.

[2] G. Desarnaud, "Cyber attacks and energy infrastructures: Anticipating risks," 2017.

[3] "European power grid organization hit by cyberattack," https://www.welivesecurity.com/2020/03/12/european-power-grid-organization-entsoe-cyberattack/, accessed: 2022-03-16.

[4] "U.s. escalates online attacks on russia's power grid," https://www.nytimes.com/2019/06/15/us/politics/trump-cyber-russia-grid.html, accessed: 2022-03-16.

[5] "Cyberattack targets safety system at saudi aramco," https://foreignpolicy.com/2017/12/21/cyber-attack-targets-safety-system-at-saudi-aramco/, accessed: 2022-03-16.

[6] D. U. Case, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, vol. 388, pp. 1–29, 2016.

[7] K.-b. Lee and J.-i. Lim, "The reality and response of cyber threats to critical infrastructure: A case study of the cyber-terror attack on the korea hydro & nuclear power co., ltd." *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 10, no. 2, pp. 857–880, 2016.

[8] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[9] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Systems*, vol. 189, p. 105124, 2020.

[10] Z. Wang, K. W. Fok, and V. L. Thing, "Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study," *Computers & Security*, vol. 113, p. 102542, 2022.

[11] M. A. Messaad, C. Jerad, and A. Sikora, "Ai approaches for iot security analysis," in *Intelligent Systems, Technologies and Applications*. Springer, 2021, pp. 47–70.

[22] R. C. Staudemeyer and E. R. Morris, "Understanding lstm–a tutorial into long short-term memory recurrent neural networks," *arXiv preprint arXiv:1909.09586*, 2019.

[12] H. Ding, R. X. Gao, A. J. Isaksson, R. G. Landers, T. Parisini, and Y. Yuan, "State of ai-based monitoring in smart manufacturing and introduction to focused section," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 5, pp. 2143–2154, 2020.

[13] T. Rieger, S. Regier, I. Stengel, and N. L. Clarke, "Fast predictive maintenance in industrial internet of things (iiot) with deep learning (dl): A review." *CERC*, pp. 69–80, 2019.

[14] Q. Song, W. Sheng, L. Kou, D. Zhao, Z. Wu, H. Fang, and X. Zhao, "Smart substation integration technology and its application in distribution power grid," *CSEE Journal of Power and Energy Systems*, vol. 2, no. 4, pp. 31–36, 2016.

[15] R. Zhu, C.-C. Liu, J. Hong, and J. Wang, "Intrusion detection against mms-based measurement attacks at digital substations," *IEEE Access*, vol. 9, pp. 1240–1249, 2020.

[16] "Concept grid: A unique testing facility dedicated to smart equipment and solutions," https://www.edf.fr/en/the-edf-group/inventing-the-future-of-energy/r-d-global-expertise/our-offers/edf-power-networks-lab/our-testing-facilities/concept-grid, accessed: 2022-03-25.

[17] H. C. Tan, V. Mohanraj, B. Chen, D. Mashima, S. K. S. Nan, and A. Yang, "An iec 61850 mms traffic parser for customizable and efficient intrusion detection," in *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2021, pp. 194–200.

[18] S. Qaiser and R. Ali, "Text mining: use of tf-idf to examine the relevance of words to documents," *International Journal of Computer Applications*, vol. 181, no. 1, pp. 25–29, 2018.

[19] R. Dzisevič and D. Šešok, "Text classification using different feature extraction approaches," in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. IEEE, 2019, pp. 1–4.

[20] S. S. Du, Y. Wang, and A. Singh, "On the power of truncated svd for general high-rank matrix estimation problems," *Advances in neural information processing systems*, vol. 30, 2017.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *Advances in neural information processing systems*, vol. 30, 2017.

AI-based Malware and Ransomware Detection Models

Benjamin Marais*†, Tony Quertier*, Stéphane Morucci*

*Orange Innovation, Rennes, France
{benjamin.marais, tony.quertier, stephane.morucci
@orange.com}
†Department of Mathematics, LMNO, University of Caen Normandy, France

***Abstract*—Cybercrime is one of the major digital threats of this century. In particular, ransomware attacks have significantly increased, resulting in global damage costs of tens of billion dollars. In this paper, we train and test different Machine Learning and Deep Learning models for malware detection, malware classification and ransomware detection. We introduce a novel and flexible solution that combines two optimized models for malware and ransomware detection. Our results demonstrate some improvements both in terms of detection performances and flexibility. In particular, our combined models pave the way for easier future enhancements using specialized and thus interchangeable detection modules.**

***Index Terms*—Malware, Ransomware, PE files, Antivirus, Cybersecurity, Artificial Intelligence**

I. Introduction

The cybercrime economy has never been so lucrative. In 2021, the global cost of cybercrime campaigns damages reached about 6 trillion USD for individuals and companies [1]. The trend is not expected to change since the estimated cost by 2025 is about 10.5 trillion USD. Since the beginning of the COVID-19 pandemic, cyber-attack campaigns have multiplied [2] and according to AV-Test [3], 150 million new malicious files have been discovered in 2021, an increase of 36% compared to 2020. Malware detection is therefore a major issue for individuals, companies and even governments. Recently, Republic of Costa Rica had to declare a state of national emergency due to a ransomware attack carried out by Conti hacker group [4]. Despite the progress made in malware and ransomware detection, the problem remains and intensifies over time, mainly because hackers techniques are constantly and rapidly evolving. To perform faster remediation activities, it is very important for security teams to quickly identify the family or the category of a detected malware. This classification problem can be successfully addressed using Machine Learning (ML) and Deep Learning (DL) techniques as soon as enough labeled data are available. In this work, we are training different algorithms to detect malware and ransomware. In particular, we build a bi-layered ransomware detection model based on two ML and DL optimized models. Static techniques are used to extract prevalent features from Windows Portable Executable (PE) files to train our models.

A. Background and Related Work

Malicious software analysis is a major research topic due to the damages malware cause [5]. With the recent advances in Artificial Intelligence (AI), cybersecurity researchers are shifting their attention to Machine Learning (ML) and Deep Learning (DL) methods to improve malicious files detection and classification [6]–[8]. Even if results are decent, these methods still need improvements [9].

Another hot research area in malware analysis relates to detection of a particular malicious file type, like a ransomware for instance. Ransomware aim at disabling the functionality of a computer, either by encrypting the machine (cryptographic-ransomware) as done by the well-known Wannacry virus [10], [11], or by blocking access to the machine (locker-ransomware) as performed by Reveton malware [12]. To regain the control of a computer, malware authors usually require to pay a ransom. Although this threat has been around for decades, it has intensified with the rise of cryptocurrencies which make it possible to receive a payment with a certain level of anonymity. Even though defensive methods exist to prevent such attack [13]–[17], 54% of them succeeded in 2021 at an estimated average unitary cost of 1.85 million USD. [18].

B. Contributions

This study aims to provide different approaches for malware detection with a particular focus on ransomware

detection. Our main contribution is an algorithm that chains two models optimized for malicious files detection and ransomware detection respectively. This solution, called "bi-layered detection model", takes as input a PE file, outputs the maliciousness of the file in a second step, and then determines if this PE file can be classified as a ransomware or not. To the best of our knowledge, we didn't find any similar approach for ransomware detection in the existing literature. Our final model is thus composed of two specialized models that can be independently trained which makes it very flexible during optimization phases and in a production environment.

C. Outline

In Section II, we present datasets and feature extraction methods used for our experiments. In Section III, we train the model for the first step: malicious file detection. In Section IV, we create a model for the classification of malware families and in particular of the ransomware category. In Section V, we train a bi-layered model that performs malware and ransomware detection and compare it to a reference model. Finally, Section VI summarizes this paper and discusses future works.

II. Datasets, Features and Models

A. Datasets

For our experiments, we rely on three different datasets. They all contain malicious and benign Portable Executable (PE) files in two different formats. These datasets are Ember [19], Bodmas [20] and PEMachine-Learning [21]. PE files distribution and format are summarized in Table I.

TABLE I: Distribution and format of each dataset

	Malicious files	Benign files	Files format
Ember	400,000	400,000	Features
Bodmas	57,293	77,142	Features
	57,293	0	PE files
PEMachine Learning	114,737	86,812	PE files

Ember: Ember is a dataset provided by Anderson et al. [19]. It contains a total of 1.1M features extracted from 400K malicious files, 400K benign files and 300K unlabeled files. Anderson et al. also provide the necessary tooling to generate a feature-based dataset.

Bodmas: Bodmas [20] shared with our team a dataset that contains $134,435$ binary files in the same format as Ember with pre-extracted features together with the $57,293$ malicious files in raw PE format. These files have been collected during one year between August 2019 and September 2020 and labeled: authors indicate the category each file belongs to. Table II presents the distribution of malware families in the Bodmas dataset. The category "other" includes about ten other categories less represented in the Bodmas dataset such as "dropper", "downloader" or "informationstealer" for instance.

TABLE II: Distribution of malware families in the Bodmas dataset

Category	Files count
Trojan	29,972
Worm	16,697
Backdoor	7,331
Ransomware	821
Other	2,471
Total	57,293

PEMachineLearning Dataset: The third dataset used is PEMachineLearning, made available by M. Lester [21]. It contains $201,549$ binary files including $114,737$ malicious files. These files have been gathered from different sources such as VirusShare[1], MalShare[2] and TheZoo[3].

B. Features

Training models for malware detection is a multi-step process. The first step consists in extracting information (i.e. features) from the PE files. For this, we rely on two pre-processing algorithms. The first one, a.k.a the Ember method, is detailed in Anderson et al. [19] and converts a PE file into a vector of $2,381$ features. Some of these features are listed in Table III. The second one, a.k.a Grayscale method, was initially submitted by Nataraj et al. [22] and converts a binary file into an image, as can be seen in Figure 1. To train our models, we choose to resize grayscale images 64×64 pixels.

C. Models

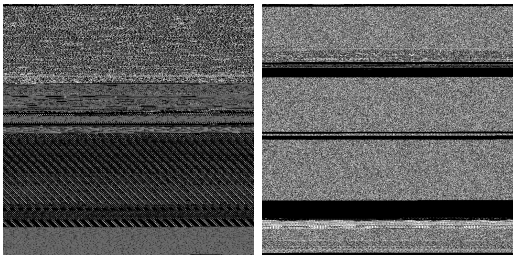
We selected four machine learning (ML) and deep learning (DL) models:

- Three models are trained with features extracted from PE files using the Ember extractor.
  - LightGBM,
  - XGBoost,
  - Dense Neural Network (DNN),
- a Convolution Neural Network (CNN) trained with PE files images created by the Grayscale extractor.

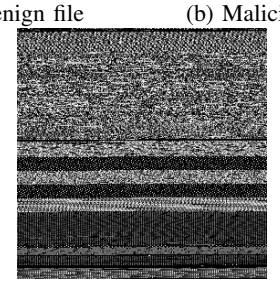
[1]https://virusshare.com/
[2]https://malshare.com/
[3]https://github.com/ytisf/theZoo

TABLE III: Some features computed using Ember extractor

Feature names	Index
Byte histogram	1-256
Byte entropy	257-512
Strings	513-616
General information	617-626
Header information	627-688
Section	689-943
Imports	944-2223
Exports	2224-2350
Data directory information	2252 - 2381



(a) Benign file (b) Malicious file



(c) Ransomware file

Fig. 1: Example of PE files transformation into grayscale images

For LightGBM and XGBoost models, we use the default parameters provided by the two python packages. For DNN and CNN models, we use Tensorflow package with the Adam optimization function [23] and a learning rate of 0.01.

III. Detection of malicious files

In this section, we are discussing the detection of malicious software using different models of machine learning. For this, we use PEMachineLearning and then Bodmas dataset. PEMachineLearning has been divided into three subsets for training (70%), validation (15%) and testing (15%) of our models.

The four models are first trained and tested on PEMachineLearning. We compare the models results using the F1 score and the accuracy score. Table IV presents the results on the test subset. The algorithm XGboost has the best results although the performance of LightGBM and DNN are relatively close. We observe that the CNN is less efficient than the three other models even if scores are close to 0.95.

TABLE IV: Malware detection results on the PEMachineLearning test subset

	LGBM	XGBoost	DNN	CNN
Accuracy	0.990	**0.993**	0.9902	0.9458
F1 Score	0.991	**0.994**	0.991	0.95

Subsequently, we test our model on malicious files from Bodmas dataset: the purpose is to validate the robustness of our trained models and to determine if they do not produce too many false negatives predictions i.e. true malware detected as benign. We focus on the false negative rate (FNR) as we consider it as a prevalent metric. For each model, FNR is showed in Table V. We also add the number of undetected malware out of 57,293 files from Bodmas. The CNN model has a high FNR compared to the other models: it does not detect enough malware from Bodmas, meaning that this model does not have a good generalization capacity. On the contrary, the XGBoost and LightGBM models have very close and low FNRs. They achieve good performances during the testing phase and have good generalization capacities. Finally, the DNN has a perfect score of zero undetected malware.

We compare our results on Bodmas with a current state of the art model. We test the LightGBM model provided by Ember [19] on the Bodmas dataset and it achieves a FNR of $1.42 \cdot 10^{-2}$. With the exception of the CNN, the models we propose are slightly better than Ember's model, even though they are trained with less but more recent data.

TABLE V: FNR and number of undetected malware from Bodmas dataset

	LGBM	XGBoost	DNN	CNN	Ember LGBM
FNR	$1.56 \cdot 10^{-3}$	$0.96 \cdot 10^{-3}$	**0**	0.13	$1.42 \cdot 10^{-2}$
Undetected malware	84	55	**0**	7448	816

Given results from Table IV and Table V, the XGBoost model seems to be the most effective model for detecting malicious files during training and testing phases. Moreover, it shows a good capacity of generalization with a low FNR on Bodmas dataset. LightGBM model provides slightly lower but very close results. Its

main advantage over XGBoost is its faster computing time [24]. Even though results for the CNN model are quite good, its poor FNR performance indicates a low generalization capacity. Finally, the DNN seems to perform better than others with results close to XGBoost on the testing subset and the lowest FNR, equal to zero, on the Bodmas dataset. On the other hand, it requires more computing power for a rather small performance increase. Thus, XGBoost model could be considered as a good trade-off between computing time and performance.

IV. Malware Categories Classification and Ransomware Detection

The purpose of this section is to present some experiments and results in the context of malware classification. We pursue two objectives: the first one consists in identifying four of the most popular malware categories (cf Section IV-A). The second objective aims at detecting ransomware only and results are presented in Section IV-B. In both cases, we use the same two extractors and the same models as presented in Section II.

Regarding the datasets at our disposal, we mainly use Bodmas because it provides labeled malicious files. However, due to a lack of ransomware, we had to manually labeled several malware from PEMachineLearning to reach a total of 2,000 labeled files.

A. Malware classification

In this part, we are classifying malware into four popular and frequently encountered malware's categories i.e. Trojan, Worm, Backdoor and Ransomware. All other types of malicious files fall into the "Other" category. To limit overfitting, we decided to use a balanced dataset instead of using all available data at our disposal. Thus, for each category of malware, we selected 2,000 malicious files from Bodmas and PEMachileLearning dataset (for ransomware). This balanced dataset has been split into a training subset (70%), a validation subset (15%) and a test subset (15%).

To compare and determine which model is the most effective for malware classification, we rely on the accuracy and F1 scores, summarized in TableVI. The best results are obtained using LightGBM, even if XGBoost and DNN performances are really close. With an average accuracy score of 0.9413 and an average F1 score of 0.9412, these three models appear to be good candidates for malware classification even if these classification results could be improved.

On the contrary, CNN doesn't perform as good as other models and its results cannot be considered as reliable. According to us, this is not only due to a low number of samples during training phase, but also to a sub-optimized pre-processing step.

TABLE VI: Malware classification results ont the test subset

	LGBM	XGBoost	DNN	CNN
Accuracy	**0.9442**	0.9427	0.9369	0.7270
F1 Score	**0.9440**	0.9425	0.9370	0.7197

B. Ransomware Detection

In this section, we are focusing on detecting ransomware among other malicious files. For this purpose, we train models on the same dataset as in Section IV-A. It is composed of 2,000 ransomware files and 8,000 malicious files. To compare the four models trained for ransomware detection, we use F1 and accuracy scores as metrics, summarized in Table VII. LightGBM, XGBoost and DNN achieve almost the same performance with scores close to 1 on the test subset. For this classification task, CNN also achieves good results.

TABLE VII: Ransomware detection results on the test subset

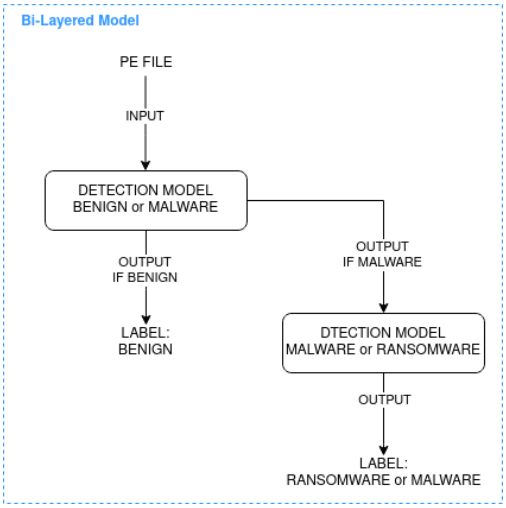
	LGBM	XGBoost	DNN	CNN
Accuracy	**0.9954**	0.9948	0.9938	0.9830
F1 Score	**0.9971**	0.9969	0.9967	0.9823

Once again, LGBM performs better even if DNN and XGBoost are also good candidates. Our results demonstrate that it seems easier to separate the ransomware category from others than to classify malware into five categories as done in Section IV-A.

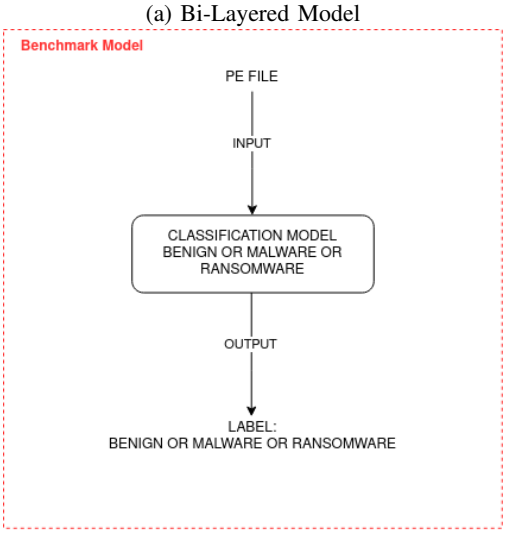
V. Bi-Layered Model for Malware and Ransomware Detection

In this section, we are considering an algorithm referred to as "bi-layered detection model". Its purpose is to detect PE files that are malicious, and also if they belong to the ransomware category. This algorithm consists in two detection layers. The first one relies on a model from Section III and its purpose is to detect malicious files. The second layer uses a model trained for ransomware detection given a malicious file (cf Section IV-B). We are combining them as presented in Figure 2a to predict if a file is benign and if it can be classified as a ransomware or not. To evaluate the performances of this combined model, we are training several reference models, referred to as "benchmark models". They are based on LightGBM, XGBoost and

DNN using dataset from Section IV-B and $8,000$ benign files from PEMachineLearning.



(a) Bi-Layered Model



(b) Benchmark Model

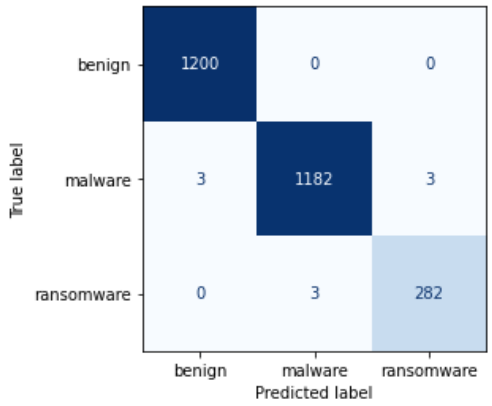
Fig. 2: Ransomware models diagrams

For each model, Table VIII summarizes the accuracy and F1 scores. Firstly, we see that our bi-layered detection models have slightly better scores than benchmark models. Overall, bi-layered detection models appear to perform better than benchmark models. We also present confusion matrices of the bi-layered and benchmark XGboost models in Figure 3. We can notice that bi-layered XGBoost returns less misclassified PE files than benchmark, which confirms that our bi-layered detection models seem to be more accurate. Finally, the XGBoost bi-layered model has a better efficiency than DNN and LightGBM, even if results are close.

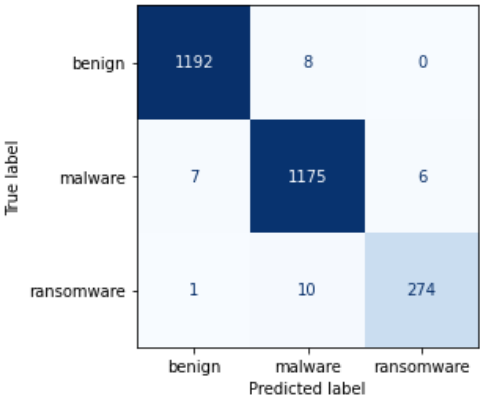
Moreover, a significant advantage of our bi-layer detection models is that they are composed of two sub-models trained for particular sub-tasks i.e. malware or ransomware detection. In consequence, each sub-model can be retrained independently if we have new data, and possibly be replaced if a better model becomes available. We are convinced that such flexibility makes our models good candidates for production environment.

TABLE VIII: Results for Benchmark and Bi-Layered models on the test subset

	Accuracy		F1 Score	
	Benchmark	Bi-Layered	Benchmark	Bi-Layered
LGBM	0.9896	0.9932	0.9895	0.9934
XGBoost	0.988	**0.9960**	0.988	**0.9966**
DNN	0.9867	0.9913	0.9868	0.9915



(a) Bi-layered XGBoost



(b) Benchmark XGBoost

Fig. 3: Confusion matrix of bi-layered and benchmark XGBoost on the test subset

VI. Conclusion & Future Work

A. Conclusion

In this paper, we have implemented and compared different methods, based on machine learning and deep learning algorithms, for malware detection and classification, and in particular ransomware detection. During each experimentation, CNN models do not perform as well as other models. We need to investigate further to identify the origin of the problem, even if we feel that the lack of data or prepossessing steps could be an explanation. In Section III, all other models achieve good results, in particular XGBoost which is the most effective on the test subset and DNN which seems to have better generalization capacity. Thus, XGBoost may appear as a good candidate for malware detection considering their respective training times. In Section IV-A, we show that about 5% of PE files are misclassified which leaves room for models optimizations. In Section IV-B, by focusing on ransomware detection only, we demonstrate that XGBoost, LightGBM and DNN have good performances with an average accuracy score of 0.9947. It coud be interesting to analyze if these findings are similar for other category of malicious files like trojan, worm or backdoor. Finally, in Section V, we propose an original approach that performs two tasks: malware detection and ransomware detection. Our "bi-layered detection model" combines two optimized models for this and achieves slightly better results than benchmark models. This combined model provides an interesting flexibility capability to envision easier optimization and evolution steps in a production environment. We hope this paper could help improve cybersecurity solutions for malicious files analysis by providing new insights.

B. Future Works

To accelerate and improve malware analysis activities, we plan to enrich results provided by ML and DL algorithms. In particular, it would be interesting to extend the method to other type of malware, and even to combine this method with the one we already proposed for the detection of packed and obfuscated files [25]. We also plan to leverage Explainable Artificial Intelligence (XAI) and Interpretable Machine Learning (IML) concepts to provide decision support. In addition, we want to integrate clustering techniques and PE file behavioral analysis to improve malware classification results. Finally, we will have to assess the robustness of our different algorithms against adversarial attacks using for instance methods described in [26].

REFERENCES

[1] Steve Morgan. Cybercrime to cost the world $10.5 trillion annually by 2025. https://cybersecurityventures.com/cyberwarfare-report-intrusion/. Online; Post : 2020-11-13.

[2] Harjinder Singh Lallie, Lynsay A. Shepherd, Jason R.C. Nurse, Arnau Erola, Gregory Epiphaniou, Carsten Maple, and Xavier Bellekens. Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. *Computers and Security*, 105:102248, 2021.

[3] Av-test. https://www.av-test.org/fr/?r=1. Accessed: 2022-05-30.

[4] Todd Drew. Costa rica declares state of emergency after conti ransomware attack. https://www.secureworld.io/industry-news/costa-rica-emergency-ransomware. Online; Post : 2022-05-12.

[5] Ross J. Anderson, Chris J. Barton, Rainer Bölme, Richard Clayton, Carlos Hernandez Gañán, Tommaso Grasso, Michael Levi, Tyler W. Moore, and Marie Vasek. Measuring the changing cost of cybercrime. 2019.

[6] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. *Computers and Security*, 81:123–147, 2019.

[7] Edward Raff and Charles Nicholas. A Survey of Machine Learning Methods and Challenges for Windows Malware Classification. 2020.

[8] Daniel Gibert, Carles Mateu, and Jordi Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, 153(January):102526, 2020.

[9] Michael R. Smith, Nicholas T. Johnson, Joe B. Ingram, Armida J. Carbajal, Bridget I. Haus, Eva Domschot, Ramyaa Ramyaa, Christopher C. Lamb, Stephen J. Verzi, and W. Philip Kegelmeyer. *Mind the Gap: On Bridging the Semantic Gap between Machine Learning and Malware Analysis*, page 49–60. Association for Computing Machinery, New York, NY, USA, 2020.

[10] Savita Mohurle and Manisha Patil. A brief study of Wannacry Threat: Ransomware Attack 2017. *International Journal of Advanced Research in Computer Science*, 8(5):1938–1940, 2017.

[11] Lena Y. Connolly and David S. Wall. The rise of crypto-ransomware in a changing cybercrime landscape: Taxonomising countermeasures. *Computers and Security*, 87, nov 2019.

[12] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9148, pages 3–24. Springer Verlag, 2015.

[13] Ronny Richardson and Max North. Ransomware: Evolution, Mitigation and Prevention. *International management review*, 13(1):10–21, 2017.

[14] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. PayBreak : Defense against cryptographic ransomware. In *ASIA CCS 2017 - Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, pages 599–611, 2017.

[15] Harun Oz, Ahmet Aris, Albert Levi, and A. Selcuk Uluagac. A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. *ACM Computing Surveys*, 2022.

[16] Sumith Maniath, Prabaharan Poornachandran, and V. G. Sujadevi. *Survey on prevention, mitigation and containment of ransomware attacks*, volume 969. Springer Singapore, 2019.

[17] McAfee Labs. Understanding Ransomware and Strategies to Defeat it. *McAfee Labs*, pages 1–18, 2016.

[18] Sophos Ltd. The State of Ransomware 2021. *Sophos Whitepaper*, 2021.

[19] Hyrum S Anderson and Phil Roth. EMBER: An open dataset for training static pe malware machine learning models, 2018.

[20] Limin Yang, Arridhana Ciptadi, Ihar Laziuk, Ali Ahmadzadeh, and Gang Wang. BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware. In *Proceedings - 2021 IEEE Symposium on Security and Privacy Workshops, SPW 2021*, pages 78–84, 2021.

[21] Michael Lester. Pratical Security Analytics - PE Malware Machine Learning Dataset. https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/.

[22] L Nataraj, S Karthikeyan, G Jacob, and B S Manjunath. Malware images: Visualization and automatic classification. In *ACM International Conference Proceeding Series*, 2011.

[23] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[24] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 3147–3155, 2017.

[25] Benjamin Marais, Tony Quertier, and Christophe Chesneau. Malware analysis with artificial intelligence and a particular attention on results interpretability. In Kenji Matsui, Sigeru Omatu, Tan Yigitcanlar, and Sara Rodríguez González, editors, *Distributed Computing and Artificial Intelligence, Volume 1: 18th International Conference*, pages 43–55, Cham, 2022. Springer International Publishing.

[26] Tony Quertier, Benjamin Marais, Stéphane Morucci, and Bertrand Fournel. Merlin–malware evasion with reinforcement learning. *arXiv preprint arXiv:2203.12980*, 2022.

TrustGAN: Training safe and trustworthy deep learning models through generative adversarial networks

Hélion du Mas des Bourboux
Thales SIX Theresis
1 av. Augustin Fresnel, 91120 Palaiseau, France
helion.dumasdesbourboux'at'thalesgroup.com

***Abstract*—Deep learning models have been developed for a variety of tasks and are deployed every day to work in real conditions. Some of these tasks are critical and models need to be trusted and safe, e.g. military communications or cancer diagnosis. These models are given real data, simulated data or combination of both and are trained to be highly predictive on them. However, gathering enough real data or simulating them to be representative of all the real conditions is: costly, sometimes impossible due to confidentiality and most of the time impossible. Indeed, real conditions are constantly changing and sometimes are intractable. A solution is to deploy machine learning models that are able to give predictions when they are confident enough otherwise raise a flag or abstain. One issue is that standard models easily fail at detecting out-of-distribution samples where their predictions are unreliable.**

We present here TrustGAN, a generative adversarial network pipeline targeting trustness. It is a deep learning pipeline which improves a target model estimation of the confidence without impacting its predictive power. The pipeline can accept any given deep learning model which outputs a prediction and a confidence on this prediction. Moreover, the pipeline does not need to modify this target model. It can thus be easily deployed in a MLOps (Machine Learning Operations) setting.

The pipeline is applied here to a target classification model trained on MNIST data to recognise numbers based on images. We compare such a model when trained in the standard way and with TrustGAN. We show that on out-of-distribution samples, here FashionMNIST and CIFAR10, the estimated confidence is largely reduced. We observe similar conclusions for a classification model trained on 1D radio signals from AugMod, tested on RML2016.04C.

***Index Terms*—Machine Learning, Deep Learning, Trust-AI, Safe-AI, Confidence, Out-of-distribution**

I. Introduction

Deep learning models are being deployed for a large variety of tasks. Even though a lot of these tasks do not primarily require decisions to be confident, e.g. advertisement or content suggestions, regulators around the world are expecting companies to build safer and trustable artificial intelligent (AI) systems (e.g. the European Union proposal for AI regulation [1]). Other AI tasks, especially for critical infrastructures, need to be robust and safe at their core before being deployed in the real world, e.g. secured communication systems, radio surveillance, navigation systems. Building safe deep learning models which can be trusted in the real world is a complex objective. One could think of gathering enough data, but that solution is either too costly, impossible due to confidentiality issues or simply impossible. Indeed a lot of tasks are constantly evolving, e.g. face masks after the covid-19 pandemic and their effect on face recognition systems (c.f. [2]). One could think of modeling the data, but this is not possible in all situations.

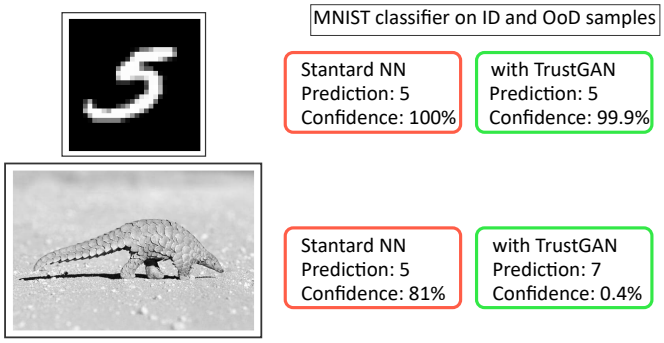


Fig. 1. Example of in-distribution (ID) and out-of-distribution (OoD) sample images given to a number classifier trained on MNIST data. The results in red are for a model trained the standard way, and in green with TrustGAN. On an ID sample (top), both give the correct prediction with a large confidence, however on an OoD sample (bottom), a standard neural network predicts a 5 with 81% confidence. This confidence drops to 0.4%, when the neural network is trained with TrustGAN.

One solution is to build a deep learning model which is as successful on a task as possible, but which also is able to raise a flag or abstain if it is not confident enough. Such a deep learning network returns now both a decision and an estimated confidence on the decision. This confidence needs to be robust to rare samples in the training set and more importantly to out-of-distribution (OoD) samples. These samples are data unknown to the network, e.g. if building a classifier to recognise helicopters from planes, a realistic OoD sample could be the image of a bird. Then a robust AI system has to return a low confidence for such an image. Gathering OoD samples or worse OoD data sets is very tedious and most of the time impossible, for similar reasons as the ones discussed above for training data sets.

Standard training pipelines for deep learning models do not focus on the estimation of the confidence on OoD. Instead these pipelines focus on getting the best performances on the training data set. That being said, most machine learning

models still output an estimation of the confidence on their decision, e.g. the maximum class probability (MCP). The estimation is known to be unreliable and overestimated (see for example [3]). We show an example of such a flaw in figure 1, where a number classifier, efficient at its task, robustly classifies the image of a pangolin as the digit 5 (red boxes).

We present here TrustGAN, a generative adversarial network (GAN [4]) pipeline targeting trustness and confidence. The goal of this pipeline is to attack the confidence estimated by a target model in order to improve upon it. We present the effect TrustGAN can have on in-distribution (ID) and OoD samples in figure 1 (green boxes). The idea of the pipeline starts with the understanding that since OoD samples are hard or impossible to gather and train on, then we could leave a GAN learning to produce them. Through these generated adversarial samples, the target network would learn both to be efficient at its target task and to understand what ID samples look like.

TrustGAN is composed of two neural networks:

1) a "target model" we want to be both as good as can be on a given task on a training data set, and be able to produce reliable confidence score on ID and OoD samples;
2) a "confidence-attacker" network, i.e. a GAN, whose goal is to attack the confidence of the target network.

We publicly release the code at github/ThalesGroup/trustGAN[1].

Other works target at improving the estimated confidence a neural network can have on its predictions. TrustGAN can either replace some or work jointly in a confidence improvement pipeline. Among the literature, [5] uses dropout layers as confidence estimations. [3] for example trains an extra neural network to output confidence scores by learning to tell when the target neural network fails on the training data set. Other works like [6] uses a GAN to learn a discriminator as a confidence estimator. Their work is similar to ours, but implies that two different networks have to be deployed on the target hardware: the target network for the prediction and the discriminator for the estimated confidence.

To demonstrate how TrustGAN works and its performances, we first present in section II the training pipeline: the specificities of the neural networks, of the different training losses, and of the training process. Then we present in section III the application of TrustGAN to the classification of numbers based on their images and the classification of radio signal modulations based on raw data. Finally we conclude and draw perspectives in section IV.

II. Adversarial network for improved confidence

We present in this section the TrustGAN pipeline, that aims at training a given neural network while improving its estimated confidence on in-distribution and out-of-distribution samples.

[1]https://github.com/ThalesGroup/trustGAN

A. A target model and its confidence-attacker

The TrustGAN pipeline is presented in figure 2. It improves a neural network model, the target, that outputs a decision, e.g. a classification, and a confidence on the given decision. The estimated confidence on the decision can be defined in a variety of ways. We work here in the most popular way, when working on a classification task: the confidence is given by the maximum class probability (MCP [7]). It is defined by:

$$MCP = \max_{0 \leq i < n} \hat{s}_i, \tag{1}$$

where n is the number of classes, $\hat{s}_i$ is the score (or probability) for class i estimated by the target network. With this definition, MCP is between $1/n$, i.e. a random prediction and 1, i.e. 100% confidence. We define then the confidence by a re-normalization of the estimated score $\hat{s}_i$ in order for the confidence to be a number between 0 (not confident) and 1 (very confident):

$$\hat{C}_i = \frac{\hat{s}_i - \frac{1}{n}}{1 - \frac{1}{n}}. \tag{2}$$

It is this confidence we aim at improving, while not hurting the predictive power of the target model.

The target model is presented in blue in figure 2: given an input data, it returns a decision and a confidence. The

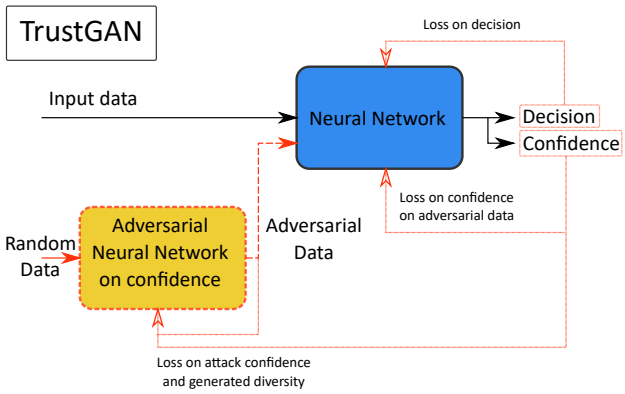


Fig. 2. Presentation of the training process for TrustGAN. A target neural network in blue is improved through an adversarial network in yellow targeting its confidence. In this diagram, red lines show inputs used only during training, black lines are both for training and inference.

pipeline does not need to change the target model, however it must be provided with a trainable white box implementation (e.g. Keras [8] or PyTorch [9]) and its training metric (i.e. the training loss). The fact that no changes have to be made on the target model, allows TrustGAN to be agnostic and thus adapt to a large variety of cases. The model parameters will be updated, if the model has been trained beforehand or simply learned from start otherwise.

Beside the target model, a generative adversarial network (GAN: [4]) aims at fooling the target model into being confident on arbitrary inputs. This confident attacker (yellow box on figure 2) generates attacks, i.e. target inputs, from

random data. This attacker must be tailored to the inputs the target model accepts. For example, if the target model is a classifier running on images, the attacker must generates from a vector of random numbers an image. However, if the target model is a classifier running on 1D radio signals, the attacker must then generates 1D data.

In the configuration of figure 2, no extra model is needed. The role of the discriminator which tells if a sample is from the training distribution or not, and which is standard in GANs, is played by the target network itself, through a confidence loss. This simple but special loss is described bellow (c.f. eqn. 3). Other works, e.g. [6], add a discriminator to estimate the confidence.

In order to train both the target and the confidence-attacker model, a training set and a validation set must be provided. Furthermore, as said above, the target model loss must also be provided. We detail bellow the training process, given in the black and red lines of figure 2.

B. Training the target model

The objective of the target model is two folds:

1) be skilled at the task it has been designed for. This task is defined by the loss provided alongside the target model definition, and is estimated on the training set;
2) refuse to be confident on data where it can not be, for example samples under represented in the training set or on out-of-distribution samples. These samples are represented by the data generated by the confidence-attacker.

These two goals can compete. For example one method to succeed in the second task is to never be confident on any decisions, and thus never provide a decision. This goes against the goal of the first task. The training pipeline has thus to find a balance between these two goals.

For task number 1, the training loss L_{00} is given by the provided loss on the task the target network is aiming at, for example a classification model could be trained with the cross-entropy loss. This task is presented on figure 2 by "Loss on decision".

For task number 2, the training loss aims at setting the confidence to what it would be for a random decision if the input data are not from the training set. For a classification task, with n classes, a random decision is when the target network returns all n probabilities to $1/n$. Thus the loss for such a task is given by the soft-cross-entropy loss:

$$L_{01} = \frac{1}{\log n} \frac{1}{n} \sum_{i=0}^{n-1} \left[-y_i \log \hat{s_i} \right]. \tag{3}$$

In this equation, y_i is the target, here $y_i = 1/n$ and $\hat{s_i}$ is the estimated probability of the target network, i.e. after softmax of the output logits $\hat{l_i}$. In this equation, $\log n$ allows to normalize the cross-entropy loss.

C. Training the confidence-attacker

From a vector of random numbers as inputs, the confidence-attacker model must generate adversarial samples as inputs to the target model with three objectives. These objectives are:

1) from random numbers generate data to fool the target network into being confident on its decision, whatever the decision is;
2) from different random numbers generate different samples;
3) from different random numbers induce different decisions of the target network.

The first task aims at improving the estimated confidence by attacking it, the other two are diversity tasks, aiming at attacking different aspects of the target model. These different tasks do not compete and can be obtained easily together. However, they do compete with the two objectives of the target network. The training pipeline has thus to find a balance between these different goals.

For task number 1, the training loss is obtained by finding the class where the confidence is maximum:

$$L_{10} = -\frac{1}{\log n} \max_{0 \leq i < n} \log \hat{s_i} \tag{4}$$

$$= \frac{1}{\log n} \times \left(- \max_{0 \leq i < n} (\hat{l_i}) + \log \sum_{i=0}^{n-1} \exp \hat{l_i} \right). \tag{5}$$

This training loss is thus pushing the confidence-attacker at generating samples where the target model confidence is $C_i = 1$ at one of the classes and 0 elsewhere.

For task number 2, the goal is to generate diverse samples, if the random numbers differ. The training loss is thus here defined by:

$$L_{11} = \frac{1}{\sum_{j=0}^{N-1} \sum_{k=0}^{j-1} \overline{|r_j - r_k|^m}} \times \sum_{j=0}^{N-1} \sum_{k=0}^{j-1} \frac{\overline{|r_j - r_k|^m}}{1 + \overline{|a_j - a_k|^m}}. \tag{6}$$

In this equation j and k are two different samples of a given set with N samples. This set can be for example a batch or a sub-sample of the batch, for faster computation. For example, one can choose $N = n$. The confidence-attacker model produces the adversarial sample a_j from the random numbers r_j for the sample j. The random numbers of the vector r_j are sampled from the uniform distribution $U([0, 1))$. In this equation m gives the order of the m-norm, typically $m = 2$. This loss L_{11} goes to 0 if the generated samples are very different.

In a similar way, task number 3, compares the target model outputs:

$$L_{12} = \frac{1}{\sum_{j=0}^{N-1} \sum_{k=0}^{j-1} \overline{|r_j - r_k|^m}} \times \sum_{j=0}^{N-1} \sum_{k=0}^{j-1} \frac{\overline{|r_j - r_k|^m}}{1 + \sum_{o=0}^{n-1} -\hat{s_{jo}} \log \hat{s_{ko}}}. \tag{7}$$

In this equation, $\hat{s_{jo}}$ is the predicted score by the target model on the adversely generated sample j of data a_j for the class o. This loss L_{12} goes to 0 if the generated scores are very different.

These three losses are combined into a single one:

$$L_{13} = \frac{1}{3}\left(L_{10} + L_{11} + L_{12}\right), \tag{8}$$

and are presented as "Loss on attack confidence and generated diversity" in figure 2.

D. Training pipeline

In order to obtain a target neural network, efficient at finding a decision and estimating a robust confidence given a sample data, we must apply the TrustGAN training pipeline. This pipeline is composed of three major steps per batch:

1) train the confident-attacker model using random numbers with L_{13};
2) train the target model on adversely generated data with L_{01};
3) train the target model on data from the training set with L_{00}.

These three steps are repeated for all epochs and for all batch in an epoch. Different checkpoints are stored to follow the evolution of the different losses during training, on both the training set and the validation set. We store per epoch the state of the best GAN of the epoch and the current state of the GAN. This allows to randomly repeat previous attacks and prevents mode collapse, in a similar way as experience replay in reinforcement learning (e.g. [10]). Furthermore, we store at each epoch, the current state of the target model.

Different parameters can be set to tune the relative power of the different losses and attacks:

- number of training epochs;
- batch size;
- number of epochs the target model is trained alone, i.e. only step 3, default value is 0;
- number of step 1 for one step 3, default value is 1;
- number of step 2 for one step 3, default value is 1;
- random proportion of times step 1 and 2 are skipped, default value is 0%;
- random proportion of times adversarial samples are generated from a random GAN checkpoints instead of the current GAN state for step 2, default value is 10%.

Finally, we observe that it is best to design a simple GAN, with largely less parameters than the target model. This allows to limit the available attacks the GAN can perform on the target model, and thus do not prevent the target model from learning the target task.

We store at each epoch different metrics: all the training losses. They allow to track the learning process. We also store at each epoch generated adversarial samples : the best attack of the epoch and a random one at the end of the epoch. They allow to visually understand, when possible, what the GAN is learning. The available metrics can help choose the best target model, on the validation set, given a compromise between the performances of the decisions and the estimation of the confidence. We here keep the best model according to the validation loss.

E. Inference

After TrustGAN training, the target model is expected to give both good predictions and robust estimated confidence. This model can thus be deployed as all standard models would be, i.e. during inference we drop all red lines and boxes of figure 2 and drop the confidence-attacker model.

Thanks to the re-normalization of the maximum class probability (eqn. 2), the confidence is provided between 0 (the decision is not confident) and 1 (the decision is confident). This allows a user to now set a threshold t on the confidence, and for a given sample i, infer and get (d_i, C_i): the decision and its estimated confidence. We now have:

- if $C_i < t$ then the result is not confident enough, raise a flag or abstain from acting;
- if $C_i \geq t$ then the decision is d_i.

The threshold t can be set by a user according to a balance between rejections, i.e. needs a human decision, and automatic decisions by the neural network.

III. Application to classification tasks

We test the TrustGAN training pipeline and compare it to a standard training pipeline in two different applications: classification of numbers and classification of radio signals. A standard training pipeline can be obtained easily from section II by setting the number of epochs the target model is trained alone to be larger than the actual number of epochs. This allows to guarantee both standard and TrustGAN training are comparable.

A. Classification of numbers

We apply TrustGAN training pipeline to the simple task of the classification of numbers given an image. We use the MNIST (Modified National Institute of Standards and Technology) data set from [11]. The task is thus a classification of 28 pixels by 28 pixels images with only one channel onto 10 classes. The training loss is thus a cross-entropy. The input images are min-max normalized over all channels, then rescaled to $[-1, 1]$. If all pixels have the same values, they are all set to 0.

Our target model is a 2D temporal convolutional neural network with residual connections (see [12], [13] and [14]) inspired by the architecture of [15]. Their structure is independent of the image length and width, and depends only on the number of channels. This allows to test the architecture on random images of any sizes, for example from the internet (e.g. figure 1). It allows then to simply test how robust the network is against out-of-distribution samples.

The architecture of the confidence-attacker network is the same as that of the target model. It only differs by few aspects. First, instead of a classification head, the network outputs images. Second, the network is less deep than the target network. This allows the GAN to attack the target model

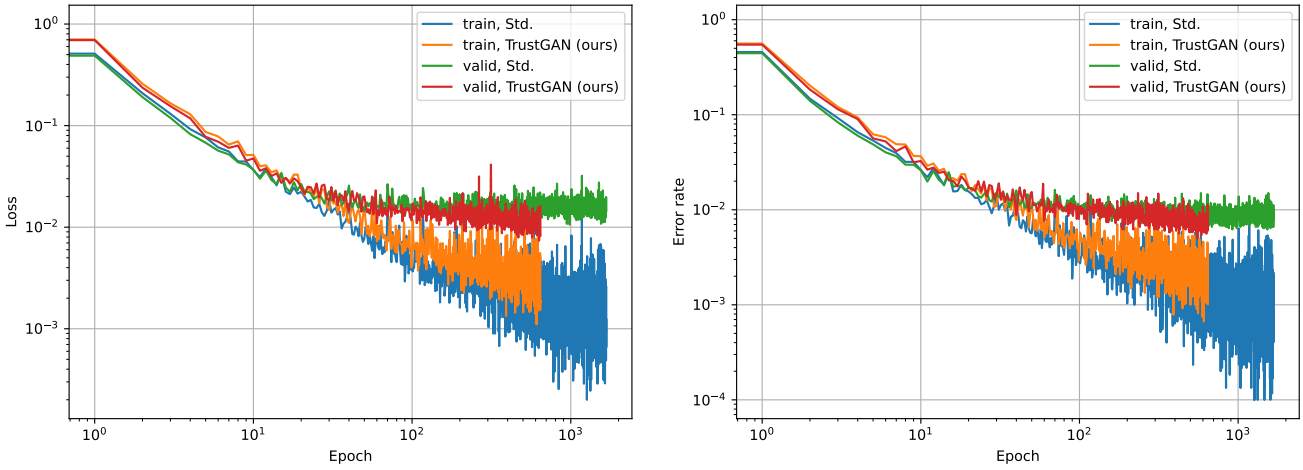


Fig. 3. Target loss, i.e. L_{00}, (left) and error rate (right) training history for the training set and the validation set for a neural network trained to recognise numbers in MNIST data, either trained the standard way, or with TrustGAN.

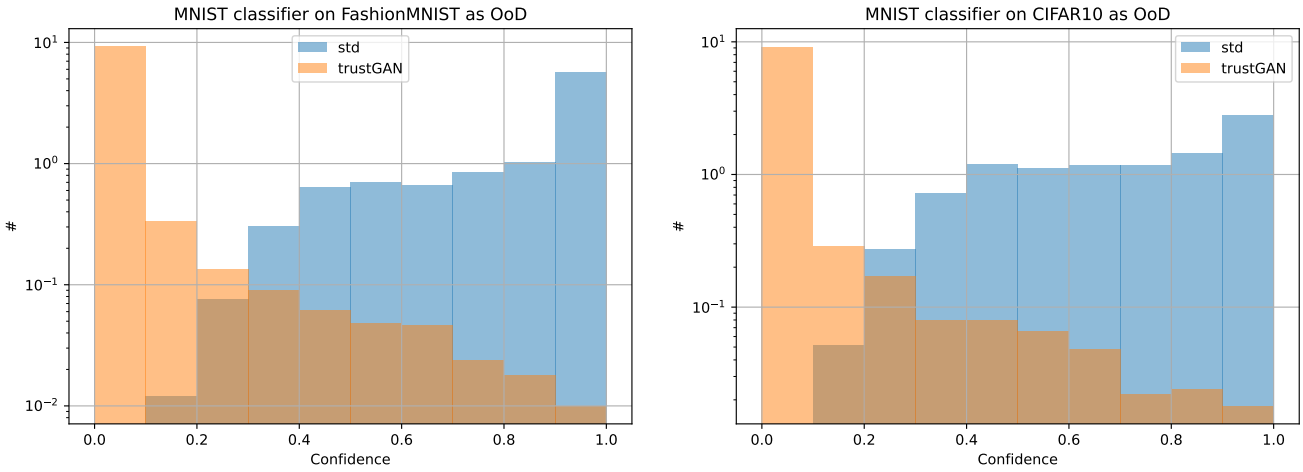


Fig. 4. Distribution of the inferred confidence on two different out-of-distribution samples: FashionMNIST (left) and CIFAR10 (right). For both panels the neural network is trained to recognise numbers in MNIST data, in blue trained the standard way, in orange with TrustGAN.

without preventing the latter from learning the target task. Third, we use LeakyReLU (see [16]), instead of ReLU (see [17]), which is known to be more adapted for GANs. Finally, for similar reasons we replace weight norm layers (see [18]) by batchnorm layers (see [19]). A tensor of random numbers of the same shape as the training set images is given to the model, which then outputs through a hyperbolic tangent (tanh) activation layer a generated image.

The number of training epochs for the standard pipeline and for the TrustGAN pipeline is set to an arbitrary high number (here 10 000). Both pipelines are stopped when no learning on the target loss is observed, on the validation set. We periodically visually track the generated images to see if the GAN is learning useful attacks.

Both the target loss and the target error rate learning history are shown on figure 3. We observe on both of the panels that with the TrustGAN pipeline the target network takes more time to get similar performances as the same model trained in the standard way. This is expected, since with TrustGAN the target model is learning the task while being attacked, which slows down learning. However after some time, both models have similar performances in term of both loss and error rate. Appendix A presents some randomly sampled GAN generated images.

In order to observe the resulting behavior of the estimated confidence, we use two publicly available data sets as out-of-distribution samples: FashionMNIST from [20] and CIFAR10 (Canadian Institute for Advanced Research) from [21]. FashionMNIST is composed of black and white images of clothes, and thus can be used without transformations. CIFAR10 is composed of colored images of animals and vehicles; we select then only the first color channel of each image.

A random selection of the images in both of the OoD data sets are inferred by our MNIST classifiers. We thus get both

a class and an estimated confidence on this class. Figure 4 presents the distribution of the confidence for a model trained with a standard pipeline (blue) and with the TrustGAN pipeline (orange). The left panel presents the results for FashionMNIST and the right panel for CIFAR10. We observe in both cases that TrustGAN allows to push the confidence of OoD to low values, i.e. both orange distributions are peaked at 0% confidence. At the contrary, the standard model gives relatively high confidences on these OoD samples. This comparison shows how the TrustGAN pipeline helped the target model not to be confident on OoD samples.

B. Classification of radio signals

The same experiment is conducted on 1D radio signals. We benefit from the publicly available data set AugMod of [15] to train a target model to recognise radio signal modulations in raw 1D I/Q signals. The target model of section III-A is modified to accept data with two channels, and 1D data. The latter is simply performed by replacing 2D convolutions with 1D convolutions. We modify in the same way the GAN. After both standard and TrustGAN training we obtain similar performances in term of accuracy (c.f. table I).

As for section III-A we examine the behavior of the estimated confidence with out-of-distribution samples. The publicly available RML2016.04C data set from [22] is trimmed from modulations also present in the AugMod data set. The resulting distribution of the inferred confidence is presented on figure 5. We observe similarly that the target model is able to better estimate its confidence on these out-of-distribution samples.

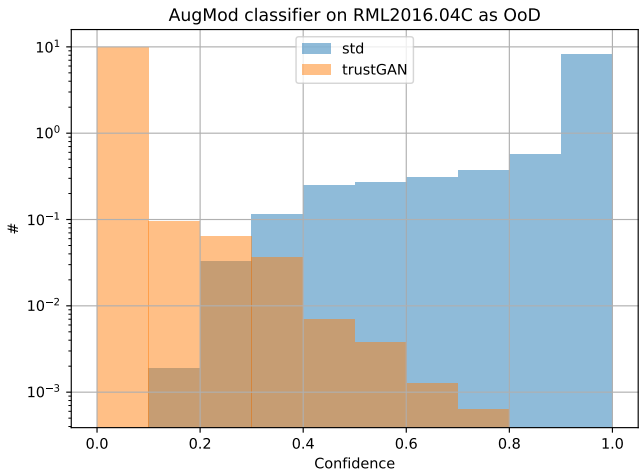


Fig. 5. Distribution of the inferred confidence on out-of-distribution samples from RML2016.04C. The target neural network is trained to recognise modulations in AugMod data, in blue trained the standard way, in orange with TrustGAN.

C. Quantitative results

Table I presents quantitative results for the different tasks defined above:

- MNIST-vs-FashionMNIST, where ID are MNIST and OoD are FashionMNIST;
- MNIST-vs-CIFAR10, where ID are MNIST and OoD are CIFAR10 first color channel;
- AugMod-vs-RML2016.04C, where ID are AugMod and OoD are RML2016.04C without modulations also present in AugMod.

We present results of the estimated MCP (maximum class probability, eqn. 2) for each task and for both a standard training pipeline ("Std., MCP") and the proposed pipeline ("TrustGAN, MCP"). Along with MCP we present the results of Monte-Carlo Dropout (MCDropout, [5]) evaluated on both the standardly trained model ("Std., MCDropout") and the TrustGAN model ("TrustGAN, MCDropout"). MCDropout is performed with a mean of the softmax probabilities over 10 realizations, using the dropout layer defined in the target network, applied on the penultimate layer, with a rate of 0.3. The performances of the different technics are measured on different metrics estimated on the test sets:

- accuracy (recall, true positive rate) of the ID test samples (higher the better);
- mean loss of the ID test samples (lower the better);
- TPR@$\geq$0.xxC, true positive rate at larger than 0.xx confidence (higher the better);
- FPR_{ID}@$\geq$0.xxC, false positive rate of in-distribution samples at larger than 0.xx confidence (lower the better);
- FPR_{ID}@0.xxTPR, false positive rate of ID samples at 0.xx true positive rate (lower the better);
- mean confidence of the OoD test samples (lower the better);
- FPR_{OoD}@$\geq$0.xxC, false positive rate of out-of-distribution samples at larger than 0.xx confidence (lower the better);
- FPR_{OoD}@0.xxTPR, false positive rate of OoD samples at 0.xx true positive rate (lower the better).

In the context of a classification task, the true positive rate (recall) is similar as the accuracy, defined as a function of the confidence threshold C on in-distribution samples (ID):

$$TPR(C) = \frac{1}{N_{\text{ID}}} \sum_{i \in \text{ID}} \mathbb{1}_{\hat{y}_i = y_i} \times \mathbb{1}_{\hat{c}_i \geq C}. \tag{9}$$

The false positive rate on ID samples is defined as:

$$FPR_{\text{ID}}(C) = \frac{1}{N_{\text{ID}}} \sum_{i \in \text{ID}} \mathbb{1}_{\hat{y}_i \neq y_i} \times \mathbb{1}_{\hat{c}_i \geq C}. \tag{10}$$

Finally, the false positive rate on out-of-distribution (OoD) samples is defined as:

$$FPR_{\text{OoD}}(C) = \frac{1}{N_{\text{OoD}}} \sum_{i \in \text{OoD}} \mathbb{1}_{\hat{c}_i \geq C}. \tag{11}$$

We can observe from this table I that on the ID testset the estimated metrics are similar between the two training technics. However, on OoD samples, TrustGAN produces significantly best scores. We observe that MCDropout alone improves the performances for the standard models, however

TABLE I

PERFORMANCE OF THE ESTIMATED CONFIDENCE USING DIFFERENT METRICS ESTIMATED ON IN-DISTRIBUTION (ID) AND/OR OUT-OF-DISTRIBUTION (OOD) TESTSET SAMPLES. FOR EACH METRIC, THE BEST SCORE IS PRESENTED IN BOLD.

	Accuracy ↑ (ID)	Loss ↓ (ID)	TPR@≥0.90C ↑ (ID)	FPR@≥0.90C ↓ (ID)	FPR@0.90TPR ↓ (ID)	Confidence ↓ (OoD)	FPR@≥0.90C ↓ (OoD)	FPR@0.90TPR ↓ (OoD vs. ID)
MNIST-vs-FashionMNIST								
Std., MCP	0.99	0.034	0.98	0.0042	0.00019	0.84	0.57	0.15
Std., MCDropout			0.97	0.0024	0.00020	0.76	0.41	0.13
TrustGAN, MCP	**0.99**	**0.027**	**0.98**	0.0032	0.00020	0.031	0.0010	0.0
TrustGAN, MCDropout			0.97	**0.0022**	**0.0**	**0.030**	**0.00080**	**0.0**
MNIST-vs-CIFAR10								
Std., MCP						0.71	0.28	0.015
Std., MCDropout						0.62	0.15	0.011
TrustGAN, MCP						0.031	0.0018	0.00020
TrustGAN, MCDropout						**0.029**	**0.0014**	**0.0**
AugMod-vs-RML2016.04C								
Std., MCP	0.98	0.042	**0.96**	0.0039	**0.00057**	0.93	0.81	0.60
Std., MCDropout			0.92	0.0024	0.0018	0.82	0.50	0.46
TrustGAN, MCP	**0.98**	**0.041**	0.95	0.0027	0.00070	0.0060	0.0	0.0
TrustGAN, MCDropout			0.89	**0.0015**	0.0016	**0.0057**	**0.0**	**0.0**

either TrustGAN alone or TrustGAN combined with MC-Dropout are systematically better. It is important to remember that all these results are observed even though the network has never seen these specific OODs during training, but has only been presented GAN generated samples instead.

IV. CONCLUSION

In this study we presented a novel training pipeline, TrustGAN, designed to robustify the estimated confidence a deep learning model returns on a given prediction. This pipeline allows to improve the behavior a given model has with respect to out-of-distribution samples or samples rare enough in the training set.

This TrustGAN pipeline does not need to change the target model in any ways, except by updating its learned weights if the model is provided already trained. The pipeline also requires a trainable white-box implementation of the target model and of its training loss. A generative adversarial network will learn to attack the target model. These special attacks, targeting slowly the model confidence, result in a model as predictive as a standard one, but also able to tell when a given input is unknown or too complicated.

We test TrustGAN on two different tasks: the classification of numbers using 2D images of the MNIST data set, and the classification of the input radio modulation in raw 1D signals of the AugMod data set. On both of these tasks we observe no significant impact on the prediction accuracy, but a large positive impact on the estimated confidence.

In the future we would like to better quantify the performances of the TrustGAN pipeline in terms of the balance between missed prediction and out-of-distribution leakage on more datasets. Furthermore, we would like to compare the resulting confidence to other different confidence estimators and robustifiers, e.g. data augmentation methods and entropy methods. We would then study how TrustGAN could be integrated to these estimators and thus see how all could combine forces to improve the estimated confidence and bring even safer, trustworthy, and deployable deep learning models for critical systems.

APPENDIX

A. GAN generated images

In figure 6 we present some GAN generated images learned while training a classification neural network on the MNIST dataset. These images are randomly sampled after training is complete, furthermore they are produced through randomly picking GANs among all stored on discs and used for experience replays.

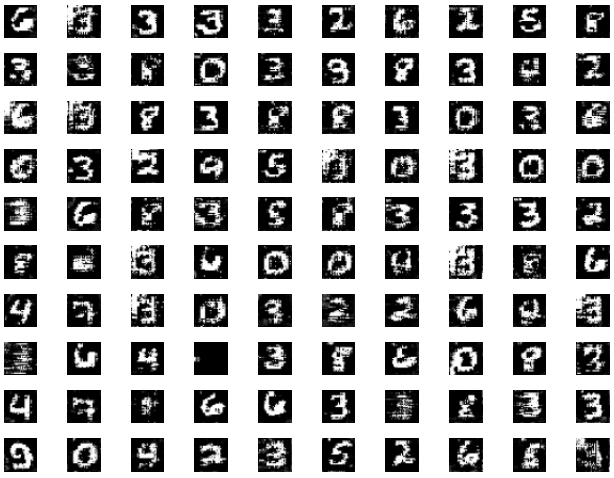


Fig. 6. GAN generated images learned while training a classification neural network on the MNIST dataset.

REFERENCES

[1] "Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts," *eur-lex*, 2021-04-21. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:52021PC0206

[2] M. Ngan, P. Grother, and K. Hanaoka, "Ongoing face recognition vendor test (frvt). part 6a: Face recognition accuracy with masks using pre-covid-19 algorithms," 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8311.pdf

[3] C. Corbière, N. Thome, A. Bar-Hen, M. Cord, and P. Pérez, "Addressing failure prediction by learning model confidence," *CoRR*, vol. abs/1910.04851, 2019. [Online]. Available: http://arxiv.org/abs/1910.04851

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[5] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *arXiv e-prints*, p. arXiv:1506.02142, Jun. 2015.

[6] K. Lee, H. Lee, K. Lee, and J. Shin, "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples," *arXiv e-prints*, p. arXiv:1711.09325, Nov. 2017.

[7] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *CoRR*, vol. abs/1610.02136, 2016. [Online]. Available: http://arxiv.org/abs/1610.02136

[8] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[10] W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, and W. Dabney, "Revisiting fundamentals of experience replay," *CoRR*, vol. abs/2007.06700, 2020. [Online]. Available: https://arxiv.org/abs/2007.06700

[11] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[12] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proceedings of the 2Nd International Conference on Neural Information Processing Systems*, ser. NIPS'89. Cambridge, MA, USA: MIT Press, 1989, pp. 396–404. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969830.2969879

[13] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," *CoRR*, vol. abs/1608.08242, 2016. [Online]. Available: http://arxiv.org/abs/1608.08242

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[15] T. Courtat and H. d. M. d. Bourboux, "A light neural network for modulation detection under impairments," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 2021, pp. 1–7.

[16] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[17] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[18] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *CoRR*, vol. abs/1602.07868, 2016. [Online]. Available: http://arxiv.org/abs/1602.07868

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[20] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[21] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[22] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks*, C. Jayne and L. Iliadis, Eds. Cham: Springer International Publishing, 2016, pp. 213–226.

Robust SAR ATR on MSTAR with Deep Learning Models trained on Full Synthetic MOCEM data

Benjamin CAMUS
Scalian DS
2 Rue Antoine Becquerel,
35700 Rennes, France.
benjamin.camus@scalian.com

Corentin LE BARBU
Scalian DS
2 Rue Antoine Becquerel,
35700 Rennes, France.
corentin.lebarbu@scalian.com

Eric MONTEUX
Scalian DS
2 Rue Antoine Becquerel,
35700 Rennes, France.
eric.monteux@scalian.com

Abstract— **the promising potential of Deep Learning for Automatic Target Recognition (ATR) on Synthetic Aperture Radar (SAR) images vanishes when considering the complexity of collecting training datasets measurements. Simulation can overcome this issue by producing synthetic training datasets. However, because of the limited representativeness of simulation, models trained in a classical way with synthetic images have limited generalization abilities when dealing with real measurement at test time. Previous works identified a set of equally promising deep-learning algorithms to tackle this issue. However, these approaches have been evaluated in a very favorable scenario with a synthetic training dataset that overfits the ground truth of the measured test data. In this work, we study the ATR problem outside of this ideal condition, which is unlikely to occur in real operational contexts. Our contribution is threefold. (1) Using the MOCEM simulator (developed by SCALIAN DS for the French MoD/DGA), we produce a synthetic MSTAR training dataset that differs significantly from the real measurements. (2) We experimentally demonstrate the limits of the state-of-the-art. (3) We show that domain randomization techniques and adversarial training can be combined to overcome this issue. We demonstrate that this approach is more robust than the state-of-the-art, with an accuracy of 75 %, while having a limited impact on computing performance during training.**

Keywords— SAR, radar, ATR, MSTAR, SAMPLE, deep learning, classification, simulation, synthetic dataset, MOCEM.

I. INTRODUCTION

THE Automatic Target Recognition (ATR) task on Synthetic Aperture Radar (SAR) images consists of designing a classification model to identify objects such as vehicles, captured by a radar. SAR images are (mathematically) complex data characterized by a large dynamic range, an important level of noise (clutter), and a limited resolution. Moreover, they capture multiple and heterogeneous electro-magnetic (EM) effects. Because of these factors, ATR on SAR images is a non-trivial task [1].

Deep Learning algorithms have demonstrated an important potential to tackle this challenge [2, 3, 4]. However, to be efficient these techniques require large training datasets. The problem is then that collecting and labelling a sufficient amount of SAR images is very expensive and in fact impossible in a defense context because, for instance, the vehicles to classify (e.g. tanks, ships) and / or the radar system (deployed on a moving platform like an aircraft or a satellite) may not be available. Moreover, the measurements may contains bias that may fool the models (e.g., classification based on environment features like bushes, instead of targets features) [5].

At the opposite, SAR simulators require only 3D CAD models of the vehicles, materials description and the radar system parameters to generate synthetic images. Besides, a full and strict control of the simulated environment reduces measurement bias, and automates the image labelling process.

For all these reasons, it is interesting to train ATR models using simulated SAR images instead of real measurements. However, because simulations rely on assumptions and real world simplifications (i.e. a model), synthetic data have a limited representativeness. Thus, the synthetic training distribution and the real measurement test distribution differ. This corresponds to the well-known dataset-shift problem [6]: ATR models trained on simulated data have a limited generalization capability to real measurements at test time.

Previous state-of-the-art works [31], especially [7], identified a set of equally promising deep-learning algorithms to tackle this issue. These studies are crucial because they showed that it is theoretically possible to train ATR models with synthetic data and compete with models trained directly on real measurements.

However, the evaluation of these works relies on the MSTAR/SAMPLE [10, 12] datasets with simulated data that reproduce as close as possible the real test measurement. As we show in this paper, the problem is that the vehicles, their configurations (e.g. articulated parts orientations), their equipment, and their positions on the images are almost identical between the synthetic training dataset and the real measurements test dataset, due to a perfect knowledge of ground truth by the authors. That is why, in this work we study the ATR problem outside of this very favorable scenario, unlikely to occur in a real operational context. Our contribution is threefold:

1. Using the MOCEM simulator, we produce a new synthetic training dataset for MSTAR that differs significantly from the real measurements. Thus, these simulated data do not suffer from the aforementioned limitations of the SAMPLE dataset.
2. We experimentally demonstrate the limits of the state-of-art approaches using our synthetic data. We show here that, among all the approaches identified in [7], the Adversarial Training (AT) [8] is the only one that offers acceptable results outside of the ideal conditions of the SAMPLE data. Still, the models accuracy significantly decrease of almost 35 %.

3. We show that intensive domain randomization [9] performed through data augmentation techniques can be combined with AT to overcome this issue and increases the accuracy up to 75 %.

The rest of the paper is as follows. Section II, details the related works and the available SAR datasets. Section III presents our synthetic MOCEM dataset. In Section IV, we demonstrate the limits of the state of the art. Section V presents our approach. Finally, in Section VI we detail our results on the MSTAR dataset and compare our approach to the other algorithms of the state of the art.

II. State of the art

A. Available datasets

The MSTAR (Moving and Stationary Target Acquisition and Recognition) public dataset [10] comprises SAR measurements of fifteen different targets taken at different depression and azimuth angles by an airborne radar. Following the standard ATR evaluation procedure with MSTAR [11], the 3671 images collected at a depression angle of 17° constitute the training set whereas the 3203 images with a depression angle of 15° serve to test the models. These data concerns ten classes of vehicles (labelled 2S1, BMP2, BDRM2, BTR60, BTR70, D7, T62, T72, ZIL131 and ZSU23-4), measured almost at each azimuth degree from 0° to 360°. Three variants of the vehicles are available for two classes: the BMP2 and the T72. The vehicles equipment and configuration (e.g. side skirts) may differ from a variant to another (see Figure *1*). It is worth noting that the azimuth angles are almost all identical between training and test sets. Moreover, these two sets comprise exactly the same vehicles on same configuration, and they are located at the same positions in the environment. Thus, the model may exploit background information to classify the targets, as the background remains the same between training and test [5].

Figure 1 Comparison between SAR images (top) and photo (bottom) of the three MSTAR variants of the T72.

The SAMPLE (Synthetic and Measured Paired Labeled Experiment) public dataset [12] comprises pairs of real SAR measurements and simulated images. This dataset is smaller than MSTAR with only 806 synth-real measurements pairs for training (at depression angles of 14°, 15° and 16°) and 539 pairs for testing (at depression angles of 17°). Like MSTAR, SAMPLE comprises ten target classes labelled 2S1, BMP2, BTR70, M1, M2, M35, M60, M548, T72, and ZSU23-4. One can note that five classes are common with MSTAR. In fact, the real measurements of these five classes correspond to the MSTAR ones (but restricted to azimuth range of SAMPLE). However, SAMPLE data does not provide any variant for any class. The SAMPLE dataset suffers from several drawbacks. First, the images azimuth angles range only from 10° to 80° for both training and test datasets. It particularly excludes the cardinal directions that may be the more challenging angles. Secondly, this angular sector may not be representative of the challenges met when classifying images at a full 360° extent. In addition, the SAMPLE authors have deployed considerable efforts to make the synthetic data as close as possible to real measurements, using detailed ground truth information. Thus, the SAMPLE simulations rely on the CAD models of the actual vehicles used during the MSTAR data collection. Furthermore, the authors configured and enriched the CAD models with detailed based on numerous notes and photos of the MSTAR measurements campaign. This very favorable scenario is unlikely to occur in an actual operational context where the measured vehicles may differ significantly from the CAD models used during training. Finally, the SAMPLE authors performed circular shifts on the measured images to center the targets exactly like in the synthetic images (see an example in Figure *2*). This alignment is unrealistic because it requires a pair of real and synthetic images of the same target captured at the same angle. Hence, it assumes to know the target class a priori, which is not compatible with the ATR problem. It is important to note that absolute target centering (i.e. without pairs of images) is not trivial on SAR images because the vehicles are only partially visible. Thus, the synthetic and measured targets are difficult to align without paired data.

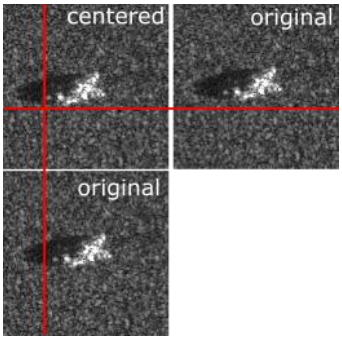


Figure 2. Comparison between an original SAMPLE image (duplicated for comparison purpose) of the T72 and its centered version.

To sum up, MSTAR and SAMPLE are the most complete ATR datasets publicly available in the state-of-the-art. They constitute key contributions to the ATR problem. However, they comprise several biases that may skew experimental results. These two datasets should thus be used with caution.

B. Related works

Ødegaard et al. got mixed results when training an off-the-shelf deep-learning algorithm directly using simulated SAR data [13]. Using a transfer learning strategy, Malmgren-Hansen et al. pre-trained an ATR classifier with a large amount of synthetic data before training the model with a smaller set of measured images [14]. The drawback of this approach is that it still requires measured data for training,

and in many cases, measured images of the target of interest will not be available.

Several works focus on learning a transport function to refine synthetic data by adding features peculiar to measured images. The goal is to transport the synthetic distribution on the measured one to fix the dataset-shift issue. The refined synthetic dataset can then be used to train a regular classifier. Cha et al. trained a residual network to refine synthetic data for ATR [15]. However, their classifier only achieve an accuracy of 55 % at test time. Lewis et al. [16] and Camus et al. [17] trained a GAN to refine synthetic SAR images, with promising results of almost 95 %. However, all these refining approaches require real measurements that may be impossible to obtain. Moreover, Camus et al. demonstrated that GAN are not able to refine new classes never seen during training [17].

Inkawhich et al. trained ATR classifiers with the public synthetic data of SAMPLE by combining several algorithms of the literature designed to improve the generalization of deep-learning models [7]. They evaluate all their models on the SAMPLE measured images. The authors compared three architectures: the network of [18], a ResNet18 [19], and a Wide-ResNet18 [20]. The techniques they studied are the following. Gaussian noise is a form of data augmentation that consists of adding random noise to the training images at runtime [21]. With dropout [22], a random noise is injected during training in the hidden units of the network to erase a given ratio of the detected features. Label smoothing [23] prevents models overconfidence and overfitting by modifying the one-hot labels of the training data so that the correct class probability become smaller than one. Mixup [24] randomly performs convex combinations of pairs of training data and labels. Thus, it promotes linear model behaviors between the training samples. Using a cosine loss [25] instead of the usual cross-entropy, the model is trained to maximize the cosine similarity between its output and the true labels. AT [8] consists of perturbing the training data on a per-pixel basis in order to maximize the model cost (using the gradient information of the network). Thus, the model becomes more robust to adversarial attacks. Finally, the bagging technique [26] averages the prediction of several models that were independently trained. For further details on these strategies, we invite the reader to refer to [7]. With a ResNet18 and several combinations of these techniques (bagging, Gaussian noise and dropout combined either with label-smoothing, mixup, cosine loss or AT), the authors found accuracies of almost 95 %. This work is groundbreaking because it demonstrates the possibility to train ATR models with full synthetic SAR images instead of real measurements. However, the authors used the SAMPLE dataset that contains several flaws, as stated on the previous section. Therefore, the proposed approaches may not be applicable in an actual operational scenario. That is why in the following section we confront the Inkawhich et al. approaches to other, more representative simulated data.

III. SYNTHETIC DATASET PRODUCTION WITH MOCEM

A. Modeling and simulation

To generate our synthetic database for the MSTAR test data, we use the MOCEM software, which is a CAD-based SAR imaging simulator developed by SCALIAN DS for the French MoD (DGA) for 20 years [27, 28]. To be as representative as possible of an operational scenario, we consider three ground-truth fidelity levels in our simulation.

1. What is known with certainty, and should be therefore accurately simulated: the radar sensor
2. What is partially known, and should therefore only be approximated in the simulation: the target signatures using the appropriate CAD models.
3. What is unknown and should thus not be faithfully modeled: the environment of the targets.

We take off-the-shelf CAD models available on Internet. We use one CAD model per MSTAR class. Thus, we do not simulate the variant vehicles for the T72 and the BMP2 classes. We manually simplify the CAD models meshes (when appropriate) to speed-up the simulation time, and we associate electro-magnetic generic materials (i.e. with well-defined reflectivity, roughness and dielectric constant) to the different facets of the model. We configure MOCEM to simulate the transfer function of the MSTAR sensor (i.e. with similar range/cross-range sampling and resolution, thermal noise level, and Taylor window function) (see Figure *3*). We run parametric simulations for the 16°, 17° and 18° incidence angles. For each incidence, we generate images at every 0.5° azimuth for the full $[0°, 360°[$ range. Thus, we do not consider exactly the same azimuth angles than MSTAR, and our training dataset has different incidence angles than the 15° MSTAR test data. The whole simulation takes two weeks to complete using two standard personal computers.

B. Qualitative evaluation of our dataset

Except for few optical photos per vehicles, we do not know the ground truth of the MSTAR measurements. It is therefore very unlikely that our off-the-shelf CAD models faithfully represent the actual vehicles used during the MSTAR measurement campaign. Moreover, an analysis of the MSTAR photo and SAR images reveals the following differences with ground truth (note that, at the opposite of classical optical images, a small difference on the vehicle geometry may have an important impact on the radar signature of the target).

Figure 3. CAD model (left) and synthetic image example (right) of the T72.

The rear spoilers of the BTR60 are too high on the CAD model. Therefore, we observe dihedral effects on our synthetic data that are not present on the measured images. The D7 blade lies on the ground on the MSTAR photo whereas it does not in our simulation (see Figure *4*). Moreover, the dimensions of the CAD model are 25 cm too large, 30 cm too long, and 4 cm too high, according to the D7 user's manual [29]. The fuel barrel of the T62 is too high on our CAD models, and the turret shape does not match the MSTAR vehicle one. Depending on the T72 variant, the CAD turret and the gun orientations are different from ground truth (see Figure *5*); the fuel barrel is missing and the equipment

are not the same. The CAD model of the ZIL131 is 15 cm too large, 40 cm too long and 6 cm too high. Moreover, the axletree, the rear bumper, and the Front winch are missing in the CAD model. Finally, the turret orientation of the ZSU23-4 differs between the simulation and the measurements (see Figure *6*). The log attached at the back of the ZSU23 is also mistakenly associated with a metal EM material.

The partial knowledge of ground truth and the lack of representativeness of CAD models (especially for equipment and articulated parts configurations) are a guaranty to ensure that our synthetic dataset differs significantly from the real measurements. Hence, it is more representative of the ATR challenges than the SAMPLE data.

Figure 4. Comparison of our CAD (left) and ground truth (right). The actual blade position is represented in yellow, and the estimated ground truth is in white.

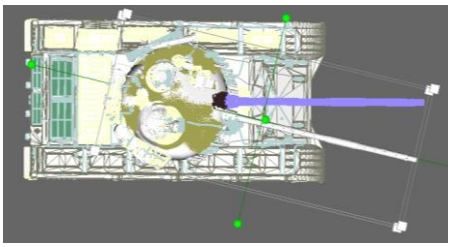


Figure 5. Actual turret position in our T72 CAD model (in violet) versus estimated position (in white).

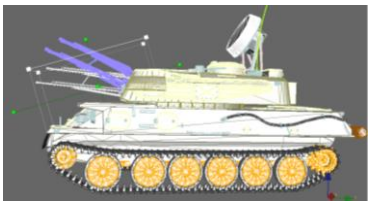


Figure 6. Actual gun position in our ZSU23-4 CAD model (in violet) versus estimated position (in white).

IV. Demonstration of the State of the Art Limits

A. Demonstration using our MOCEM dataset

1) Datasets: To study the limits of the state of the art, we apply the algorithms identified by Inkawhich et al. [7] on our MOCEM synthetic data. We evaluate our ATR models with the MSTAR test data (at a depression of 15°). Thus, on the contrary to the SAMPLE dataset used in the original paper, the data ranges on a full 360° azimuth extent, the targets are not pairwise aligned on the synthetic and measured images, and the CAD models of the vehicles does not match the measurements ground truth. We are then closer to a real operational scenario than with the SAMPLE dataset.

2) Methods: We use the ResNet18 architecture that gave the best results in the original paper. We follow an experimental plan similar to Inkawhich et al. First, we evaluate a baseline approach that does not involve any improvement techniques. Then, we measure the accuracy obtained with label smoothing (lblsm), mixup, cosine loss and AT applied separately. We also evaluate the models trained with a combination of dropout and Gaussian noise (gauss). Finally, we use Gaussian noise and dropout with either label smoothing, mixup, cosine loss and AT. As the training algorithms are stochastic, we average the accuracy over 50 different trainings for each experimental configuration.

3) Results: The "centered" column of Table I shows the results of these experiments (note that we also present the results obtained by Inkawhich et al. on the original SAMPLE data in the table). We observe an important gap between the results on the SAMPLE data and the MOCEM/MSTAR ones. With our MOCEM data, AT gives the best results with only 27 % of accuracy. Considering that this technique achieves an accuracy of almost 77 % on the SAMPLE data, we have a gap of 50 % between SAMPLE and the MOCEM/MSTAR data. At the opposite, the best algorithm with SAMPLE is a combination of label smoothing, Gaussian noise and dropout that produces models with an accuracy of 91 %. This configuration only reaches 20 % in our more realistic context (i.e. a decrease of 71 %). It is important to note that we observe a good convergence of all the training algorithms. Thus, these poor results cannot be detected during training, which makes any fine-tuning of the hyper-parameters irrelevant.

The absence of (unrealistic) targets alignment in the MOCEM/MSTAR data might explain (at least partially) these poor performances. Thus, we test if a simple data augmentation can fix this issue by making the models less sensitive to targets centering. At runtime during training, we apply circular shifts on the synthetic images. The x and y pixel offsets of each shift are uniformly sampled on the $[-5, 5]$ integer interval. We run again all our experiments with this data augmentation. The Table I shows our results in the "rnd shift" column. We observe that circular shift indeed increases the model accuracy. However, the results are still low compared to the performances of Inkawhich et al. AT is still the best with an accuracy of almost 55 %. However, it is 22 % lower than with the SAMPLE data.

TABLE I
AVERAGE ACCURACY OF THE ALGORITHMS IDENTIFIED BY INKAWHICH ET AL. ON OUR MOCEM/MSTAR DATA.

algorithms	MOCEM training data		original results on
	centered	rnd shift	SAMPLE
baseline	23.52%	43.06%	63.84%
lblsm	23.95%	44.37%	77.49%
mixup	21.26%	35.97%	80.10%
cosine loss	22.41%	43.79%	75.67%
AT	**27.20%**	**54.48%**	76.78%
gauss, dropout	21.00%	30.01%	88.28%
lblsm, gauss, dropout	20.34%	31.42%	**91.09%**
mixup, gauss, dropout	21.42%	28.57%	90.31%
cosine loss, gauss, dropout	20.04%	27.70%	89.10%
AT, gauss, dropout	21.87%	36.94%	89.50%

This experiment demonstrates that we cannot apply the algorithms identified by Inkawhich et al. on our MOCEM/MSTAR data. We formulate two non-exclusive assumptions to explain this fact:

Hypothesis 1: *The algorithms identified by Inkawhich et al. are not adapted to a real operational scenario where the synthetic data does not exactly match the measurements ground truth (like with our MOCEM/MSTAR data).*

Hypothesis 2: *Our MOCEM data are not accurate enough for training deep-learning algorithms (e.g., our simulations might not take account of some important electromagnetic effects).*

We invalidate hypothesis 2 later in Section VI.A using our deep-learning approach. To prove hypothesis 1, we slightly modify the SAMPLE test data to be more compliant with an actual operational scenario, and test the models of Inkawhich et al. on this new dataset.

B. Demonstration using a modified SAMPLE dataset

1) Dataset: First, we used the original not-shifted test measured images of SAMPLE. Thus, the targets are not exactly aligned between the synthetic and the measured images, as expected in a realistic use case. Then, we add the MSTAR images of the variant vehicles that comes from elevation 17° and belongs to the restricted [10°, 80°] azimuth range of SAMPLE. In this way, for the T72 and BMP2 classes, the CAD models of SAMPLE does not match the ground truth of the measured images. It is important to note that these extra MSTAR images in SAMPLE does not skew the results because they originate from the same measurement campaign, and we use the same Quarter Power Magnitude (QPM) LUT [30] on the images. This new dataset is more complex than the original SAMPLE one, but it still corresponds to a very favorable and unlikely scenario because: (1) the azimuth range of [10°, 80°] is too limited, and (2) for 75 % of the synthetic training data, the CAD models still match the ground truth of the measured images.

2) Results: However, as shown on the results of Table II, this is enough to show the limits of the algorithms of the literature. The top accuracy is only of 49 % with the mixup, Gaussian noise, dropout combo. This represent a decrease of around 41 % compared to the original SAMPLE test data. The results are better when we add our random circular shift data augmentation but the top accuracy (from the AT, gauss, dropout combination) only reaches 66 % -i.e. -23 % compared to the original SAMPLE data. These weakened results remain however far above the other works of the literature that train ATR models with full synthetic datasets.

TABLE II
AVERAGE ACCURACY OF THE MODELS OF THE LITERATURE TESTED WITH OUR NEW SAMPLE DATASET.

algorithms	SAMPLE training data		original results on
	centered	rnd shift	SAMPLE
baseline	33.55%	38.53%	63.84%
lblsm	40.30%	41.82%	77.49%
mixup	38.50%	47.94%	80.10%
cosine_loss	38.48%	40.33%	75.67%
AT	39.71%	47.81%	76.78%
gauss, dropout	46.90%	56.26%	88.28%
lblsm, gauss, dropout	47.87%	54.84%	**91.09%**
mixup, gauss, dropout	**49.22%**	47.87%	90.31%
cosine_loss, gauss, dropout	46.11%	53.87%	89.10%
AT, gauss, dropout	47.00%	**66.06%**	89.50%

C. Conclusion on the limits of the state of the art

From this series of experiments, we conclude that the algorithms identified by Inkawhich et al. are not adapted to a real operational scenario where the synthetic data does not exactly match the measurements ground truth. This confirm that the SAMPLE dataset is flawed and should be used with caution. In a consistent way with other works of the literature [7, 31], we note that AT has a strong potential to solve the ATR problem as it gives the best performances. However, this technique is not sufficient by itself. In the following section, we detail how we combine AT with an intensive domain randomization strategy to solve this issue.

V. OUR APPROACH

A. Method

In our approach, we make a different implementation of AT than Inkawhich et al. To attack the training images we use the Fast Gradient Sign Method (FGSM) [32] instead of the iterative Projected Gradient Descent (PGD) algorithm. Although PGD is theoretically more optimal than FGSM, we find better results with the later method. This may be surprising, but this is consistent with other works in the literature that found better results with FGSM than with PGD for training models to be robust to adversarial attacks [33]. A positive side effect is that FGSM is much faster than PGD because it only requires one additional forward/back-propagation step instead of 50 steps with the PGD of Inkawhich et al. We also find better results when using a L2 norm to bound the adversarial noise (at a value of two in our experiment) instead of the "infinity norm" used by Inkawhich et al.

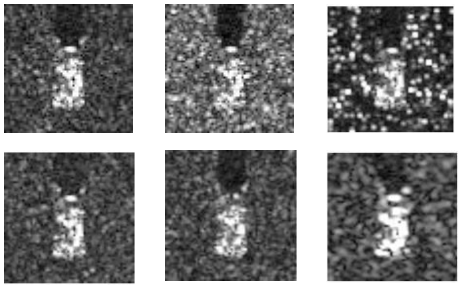


Figure 7. Example of domain randomization: reference image (top left), background clutter level (top center) and distribution (top right), NeSimga0 level (bottom left), target signature (bottom center), and radar resolution (bottom right).

We combine AT with an intensive domain randomization strategy [34]. This technique was originally developed to train robotic systems in a simulated environment. It consists of randomizing the simulated environment parameters to introduce as much variations in the synthetic data as possible. Then, the goal of this heuristic it to stretch the synthetic training distribution so that it eventually cover all the measured test distribution. We adapt this strategy to the SAR ATR problem in the following way. For each training epoch, we create a variant of all our MOCEM dataset by randomly determining for each image separately: the range and cross-range resolution, the level and distribution of the background clutter, the thermal noise of the sensor, and the target position in the images (see Figure *7* for examples). We also select the bright points of the target

that are greater than half the maximum amplitude, and randomly dropout (i.e. assign zero values on) half of them to introduce variations in the target signatures. Finally, we use the bagging method to average the prediction of ten independent models.

We also perform Test-Time Data Augmentation (TTDA) at inference-time with our ATR models [37]. We create 20 variants of each test image by applying random circular shifts of [-5, 5] pixels range in the x and y dimensions. Then, we average the model predictions on the 20 variants to classify the image. The Figure 8 sums-up our complete training workflow.

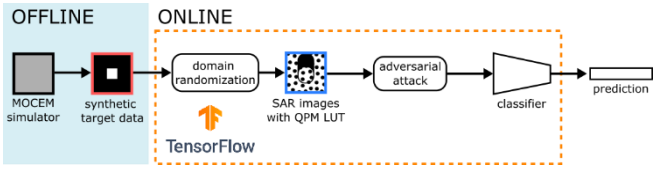


Figure 8. Our complete training workflow.

B. *Architecture, optimizer and hyperparameters*

We use a DenseNet121 architecture [38], a SGD (Stochastic Gradient Descent) optimizer with Nesterov momentum [39, 40], a weight decay of 10^{-4}, and a batch size of 128. The learning rate and the momentum vary during the training according to a "1cycle" policy [41, 42], as shown on Figure 9. We found that AT and the domain randomization slow down the convergence of the training. That is why we train our models for 150 epochs.

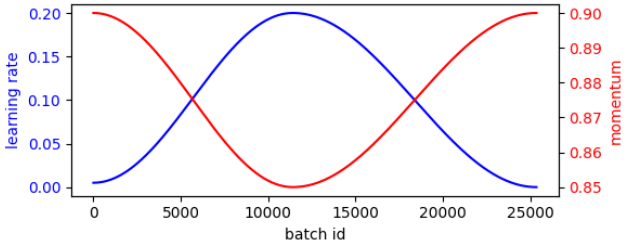


Figure 9. Evolution of the learning rate and momentum in our training according to the "1cycle" policy.

As shown in Table III, we define a uniform distribution for each of the domain randomization variables. We deliberately choose large ranges for the background clutter (level and distribution) and the target positions because this information cannot be known a priori in an operational context [35].

TABLE III
HYPER-PARAMETERS VALUES OF OUR DOMAIN RANDOMIZATION IN OUR EXPERIMENTS.

parameters	possible value ranges
range resolution	[0.203125 m, 0.35 m]
cross-range resolution	[0.21 m, 0.35 m]
background clutter level	[-20 dB.m²/m², -5 dB.m²/m²]
baground clutter gamma distribution parameter	[2., 10.]
thermal noise level	[-25 dB.m²/m², -15 dB.m²/m²]
circular shift offsets	[-5 px, 5 px]

To run our experiments, we implement the domain randomization in a Tensorflow (TF) [36] model that performs data augmentation at runtime during the classifier training. This TF model processes batches of target signatures and shadow masks generated with MOCEM. From these inputs, it produces batches of augmented SAR images. Thanks to this TF model, we only have to simulate each training image once. It is important to note that we could achieve the same results by simulating each image with MOCEM specifically for each training epoch with different simulation parameters. Nevertheless, our MOCEM/TF implementation is significantly more efficient. With our TF data-augmentation model, we are indeed able to generate more than 300,000 augmented SAR images per minutes on a single NVIDIA GeForce RTX 2060 GPU. Thus, our domain randomization approach is fast enough to train ATR models in a reasonable time.

VI. RESULTS

We evaluate our approach and synthetic data through two different experiments. First, we consider a complete and representative use case with the MSTAR test data. Then, to be able to compare our results with the state of the art on SAMPLE, we consider a more restricted scenario.

A. *Evaluating our approach and data with MSTAR*

1) Accuracy of our approach: Because our training algorithm is stochastic, we train 50 different models with our MOCEM dataset. Then, we test 1e6 combinations of bagging on the MSTAR test data at 15° (including the BMP2 and T72 variants). The average test accuracy of all the bagging is 75.34 %, with a minimum at 74.43 % and a maximum at 76.24 %. Thus, we conclude that with our MOCEM training dataset, our approach gives stable performances that are 48 % higher than the top algorithm of the literature (based on the "centered" results of Table I). This invalidates hypothesis 2, and shows that our MOCEM simulated data are accurate enough to train deep-learning algorithms. Figure 10 shows the confusion matrix of a bagging. Table IV shows the average accuracies per classes over the 1e6 bagging. We observe that for five classes we have an accuracy greater than 80 %. For the two classes with variants –i.e. BMP2 and T72—we have an accuracy greater than 75 %. This shows that we are robust to theses variants.

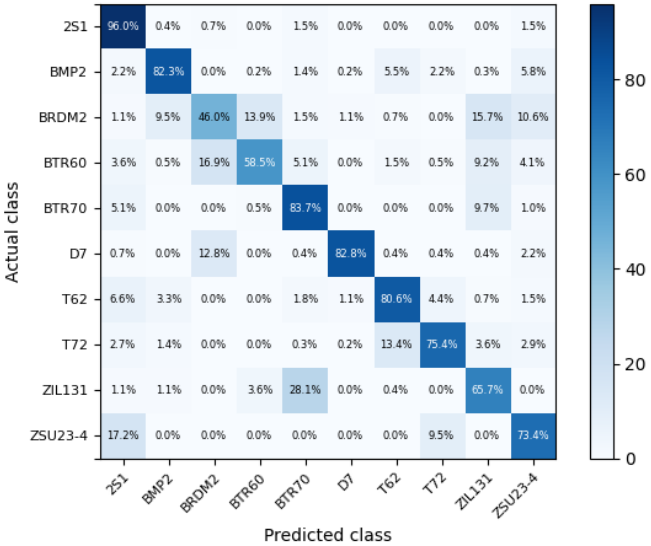


Figure 10. Confusion matrix of a bagging of models.

TABLE IV
AVERAGE ACCURACIES PER CLASSES WITH OUR APPROACH.

	mean	max	min
2S1	95.89%	96.35%	94.53%
BMP2	80.16%	82.28%	78.53%
BRDM2	47.62%	51.09%	44.16%
BTR60	58.99%	64.10%	54.36%
BTR70	83.39%	84.18%	81.63%
D7	81.84%	85.04%	78.10%
T62	80.17%	82.78%	77.29%
T72	77.22%	79.55%	74.57%
ZIL131	65.64%	68.98%	62.41%
ZSU23	72.53%	74.45%	70.07%

2) Ablation study: We perform an ablation study to measure the contribution of the different techniques composing our approach. The results are shown on Table V. We progressively remove each method and measure the average accuracy on the MSTAR data over 50 different trainings. TTDA and bagging have a negligible impact on the results (of respectively +1.10 % and +0.88 %). Using a DenseNet121 instead of a ResNet18 adds 6.66 % of accuracy. This is because our intensive AT and domain randomization methods makes the function to fit more complex, and therefore requires a bigger architecture, with more capacities [43]. In accordance with our experiments on SAMPLE, we observe that AT significantly increases accuracy of 16.22 %. When removing domain randomization, we need to choose a background clutter level in our synthetic images. This choice is not trivial because it cannot be known a priori [35] and it directly determines the signal-to-noise ratio. We test several levels between -20dB.m²/m² and -5dB.m²/m² with steps of 5dB.m²/m². The worst-case scenario corresponds to a clutter of -20dB.m²/m². Compared to this scenario, domain randomization increases accuracy by 24.95 %. The best-case scenario is with a background clutter of -15 dB.m²/m². This is consistent as it is close to the actual average clutter level of the MSTAR data. Even in this optimal context, domain randomization still brings a gain of 7.42 %. Finally, the random circular shifts increases accuracy of 6.27 % and 19.53 % with respectively the worst and the best background clutter level. To sum-up, AT and domain randomization are two essential components of our approach, as they boost accuracy of about 45 %.

TABLE V
ABLATION STUDY RESULTS

clutter level without domain rdn (dB.m²/m²)	**-20**	**-15**
techniques	**accuracy**	
bagging, DenseNet, rnd shift, domain rnd, AT, TTDA	**75.34%**	
bagging, DenseNet, rnd shift, domain rnd, AT	74.24%	
DenseNet, rnd shift, domain rnd, AT	73.36%	
ResNet, rnd shift, domain rnd, AT	66.70%	
ResNet, rnd shift, domain rnd	50.48%	
ResNet, rnd shift	25.53%	43.06%
ResNet	19.26%	23.52%

B. Comparing our approach with the state of the art

1) Methods: We run the following experiment to compare our synthetic MOCEM dataset and our deep learning approach to the state of the art. On one hand, we train the different algorithms of Inkawhich et al. with the synthetic SAMPLE data. On the other hand, we use our approach to train an ATR model with our MSTAR/MOCEM synthetic data. To be fair, we use the same ResNet18 architecture (and optimizer) than Inkawhich et al. instead of our DenseNet121. We also disable TTDA with our approach.

2) Datasets: We test all the models on the same measured dataset without using models bagging. For the approaches to be comparable, we restrict the different datasets to the [10°, 80°] azimuth range of SAMPLE and to the five targets that are common between MSTAR and SAMPLE, namely: 2S1, BMP2, BTR70, T72, and ZSU23-4. To be as representative as possible of an operational context, for testing the models we use our modified SAMPLE measured dataset presented previously in Section IV.B.1 (i.e. without targets centering and with the MSTAR variants for the T72 and the BMP2). It is important to note that this test scenario advantages greatly the state-of-the-art approaches because the CAD models of SAMPLE still (unrealistically) match the measurement ground truth for 60 % of the images. To be fair, we restrict our MOCEM dataset to have the same number of images than SAMPLE (i.e. 386 images) with similar depression and azimuth angles (we select the neareast parametric angles of our dataset).

3) Results: Table VI shows the results of this experiment averaged over 50 trainings. We observe that, despite a very unfavorable scenario, our approach has the best results. We are 18 % above the best approach of the literature. When we add a simple random circular shift of [-5, 5] pixel range to the algorithms of Inkawhich et al., we are still 9 % above the top results. With the baseline cases, (i.e. only a ResNet18 without any improvement technique) we observe that the MOCEM and the SAMPLE synthetic datasets give similar results. This may indicate that the two datasets have similar representativeness for SAR ATR. Interestingly, our approach can benefits from additional training images: if we use the all the 2215 images of our MOCEM dataset that belongs to the [10°, 80°] azimuth range, our approach reaches an accuracy of 78.82 % -i.e. +14.74 % compared to a training with only the 386 images that correspond to the SAMPLE angles.

TABLE VI
COMPARISON BETWEEN OUR APPROACH AND THE STATE-OF-THE-ART.

training data	**algorithms**	**test accuracies**	
		centered training data	**training with random shifts**
SAMPLE	baseline	35.85%	49.59%
	lblsm	36.59%	51.68%
	mixup	32.51%	45.68%
	cosine_loss	35.42%	45.75%
	AT	35.32%	54.62%
	gauss, dropout	43.37%	42.38%
	lblsm, gauss, dropout	44.01%	44.73%
	mixup, gauss, dropout	43.83%	38.36%
	cosine_loss, gauss, dropout	43.58%	35.69%
	AT, gauss, dropout	**45.99%**	**55.48%**
MOCEM	baseline	43.06%	36.53%
	ours	**64.08%**	

C. Comparison with models trained on measured data

1) Motivation: Our 75 % accuracy on MSTAR may seems low compared to the 95 % achieved by Morgan when

training ATR models directly with the MSTAR measurements [4]. However, as discussed in Section II.A, for this baseline result the models are trained and tested on measured images of exactly the same vehicles, with the same configurations and backgrounds. We run the following experiment to estimate the performance achieved outside of this utopian scenario.

2) Method and dataset: We reproduce the Morgan's experiment, with the same algorithm and the MSTAR train and test data of the ten classes. However, we only keep one variant in the training data for the BMP2 and the T72 classes. Then we measure the accuracy of the model on the different variants for these two classes. This way, for two classes the models are tested and trained on different vehicles of the same classes, like in an actual operational scenario. As the training process is stochastic, we average the results over 100 trainings.

3) Results: Table VII and VIII shows the results of this experiment. We observe significant accuracy drops when we test the models on new variants never seen at training time. For the T72 class, we have an average accuracy of 93.44 % when the test variant is the same than the train variant. When the variant are different between test and train, the average accuracy falls at 71.59 % (i.e. a decrease of 21.85 %). For the BMP2 class, the average accuracy drops from 80.45 % to 67.65 % (i.e. -12.81 %).

We want to stress that this experiment has a limited scope because it concerns only two classes. However, this may show that the results of the MSTAR literature should be considered cautiously.

On the same two classes, our MOCEM approach has an average accuracy of 80.16 % for the BMP2 and 77.22 % for the T72 (as shown previously on Table IV). Thus, despite using a full synthetic training dataset, we achieve better results with our approach than models trained directly on measured data but with different variants (+12.5 % for the BMP2 and +5.63 % for the T72).

TABLE VII
INFLUENCE OF THE T72 MSTAR VARIANTS (IDENTIFIED BY THEIR SERIAL NUMBER) ON THE ACCURACY.

		T72 variant used for test		
		132	**812**	**s7**
T72 variant used for training	**132**	93.13%	64.78%	64.95%
	812	72.02%	96.14%	72.42%
	s7	70.92%	84.31%	91.00%

TABLE VIII
INFLUENCE OF THE BMP2 MSTAR VARIANTS (IDENTIFIED BY THEIR SERIAL NUMBER) ON THE ACCURACY.

		BMP2 variant used for test		
		9563	**9566**	**c21**
BMP2 variant used for training	**9563**	77.49%	60.07%	60.44%
	9566	72.82%	83.40%	60.47%
	c21	81.86%	69.82%	80.48%

VII. CONCLUSION

In this work, we study the problem of training ATR classifiers that transfer to real measurements using synthetic SAR images. We demonstrate that the publicly available datasets of the literature (the MSTAR and SAMPLE data) are limited because they are not fully representative of the challenges met in a real operational context. In particular, their training set unrealistically match the ground truth of their measure test data (e.g., the CAD models of SAMPLE are very close to the measured vehicles, the position of the targets are identical on the images). These dataset remains however essential in the literature to conduct experiments. That is why they should be used cautiously.

Considering this potential skew, we show the limitation of the key set of approaches identified by Inkawhich et al. using the SAMPLE dataset. We show that with slight modifications of the test data (removing the targets alignment and adding variants of vehicles for two classes) the top accuracy decreases of 42 % (from 91 % to 49 %).

Using the MOCEM simulator, we produce a new synthetic training dataset for MSTAR that differs significantly from the real measurements. This dataset is then more representative of the ATR challenges. We show that the approaches of the literature give poor results (i.e. only 54 % of top accuracy) with our synthetic dataset. Thanks to all our experiments, we identified that among all the methods proposed by Inkawhich et al., AT is the most promising. However, we conclude that this approach is not sufficient by itself.

That is why we propose a new approach that combines AT with an intensive SAR-based domain randomization technique. Following this strategy, at each epoch we randomly change the training images range and cross-range resolution, the level and distribution of the background clutter, the thermal noise of the sensor, the target position in the images, and the target signature. We implement this domain randomization in a Tensorflow model that performs data augmentation at runtime during training directly on the GPU. However, we could achieve the same results (but less efficiently) by simulating each images with MOCEM specifically for each training epoch with different simulation parameters.

With this approach and our full synthetic MOCEM training data, we achieve an accuracy of 75 % on the MSTAR test data. These results are 48 % above the top results obtained when training the approaches of the literature with our MOCEM dataset. We also show that with MOCEM data, our solution achieves an accuracy 18 % above the top algorithms of literature trained with the SAMPLE synthetic data. In an actual scenario, our solution might give similar results than models trained directly on measured data with classical algorithms, although further experimentations should be done to confirm this point. This work demonstrates that our approach is relevant and that the MOCEM simulator is accurate enough to generate representative synthetic datasets, able to train deep-learning algorithms.

In future works, we plan to extend our domain randomization strategy to modify other SAR-based simulation parameters. To identify these new parameters, we plan to use explicability tools to understand what features are essential to correctly classify the images. We also want to perform a more advanced ablation study to measure which parameters of the domain randomization are essential.

REFERENCES

[1] L. M. Novak, G. J. Owirka and W. S. Brower, "An efficient multi-target SAR ATR algorithm," Conference Record of ACSSC, 1998, pp. 3-13

[2] A. El Housseini, A. Toumi and A. Khenchaf, "Deep Learning for target recognition from SAR images," 2017 Seminar on DAT, 2017, pp. 1-5.

[3] Y. Li, et al. "DeepSAR-Net: Deep convolutional neural networks for SAR target recognition," 2017 IEEE ICBDA, 2017, pp. 740-743.

[4] D. Morgan. "Deep convolutional neural networks for ATR from SAR imagery." Algorithms for Synthetic Aperture Radar Imagery XXII. Vol. 9475. SPIE, 2015.

[5] C. Belloni. Deep learning and featured-based classification techniques for radar imagery. Computer Vision and Pattern Recognition. PhD thesis of Ecole nationale supérieure Mines-Télécom Atlantique, 2019.

[6] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N.D. Lawrence. 2009. Dataset Shift in Machine Learning. The MIT Press.

[7] N. Inkawhich et al., "Bridging a Gap in SAR-ATR: Training on Fully Synthetic and Testing on Measured Data," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 2942-2955, 2021.

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, "Towards deep learning models resistant to adversarial attacks", Proc. Int. Conf. Learn. Representations, 2018.

[9] Tobin et al. Domain randomization for transfering deep neural networks from simulation to the real world. IEEE IROS 2017

[10] https://www.sdms.afrl.af.mil/index.php?collection=mstar

[11] T. Ross, S. Worrell, V. Velten, J. Mossing and M. Bryant, "Standard SAR ATR evaluation experiments using the MSTAR public release data set," Proc. SPIE 3370, Algorithms for Synthetic Aperture Radar Imagery V, 1998

[12] B. Lewis, et al. "A SAR dataset for ATR development: the Synthetic and Measured Paired Labeled Experiment (SAMPLE)." Algorithms for Synthetic Aperture Radar Imagery XXVI. Vol. 10987. SPIE, 2019.

[13] N. Ødegaard, A. O. Knapskog, C. Cochin and J. Louvigne, "Classification of ships using real and simulated data in a convolutional neural network," 2016 IEEE RadarConf, 2016.

[14] D. Malmgren-Hansen et al., "Improving SAR Automatic Target Recognition Models With Transfer Learning From Simulated Data," in IEEE GRSL, vol. 14, no. 9, Sept. 2017.

[15] M. Cha, et al., "Improving Sar Automatic Target Recognition Using Simulated Images Under Deep Residual Refinements," IEEE ICASSP, 2018.

[16] B. Lewis, J. Liu, A. Wong, "Generative adversarial networks for SAR image realism," Proc. SPIE, Algorithms for Synthetic Aperture Radar Imagery XXV, 1064709, 2018.

[17] B. Camus, E. Monteux and M. Vermet. Refining Simulated SAR images with conditional GAN to train ATR Algorithms. In Proceedings of the Conference on Artificial Intelligence for Defense (CAID), 2020.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit.Comput. Soc., 2016, pp. 770–778

[20] S. Zagoruyko and N. Komodakis, "Wide residual networks," in Proc. Brit. Mach. Vis. Conf., 2016, pp. 87.1–87.12.

[21] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning", J. Big Data, vol. 6, 2019.

[22] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929-1958, 2014.

[23] R. Möller, S. Kornblith and G. E. Hinton, "When does label smoothing help", Proc. Adv. Neural Inf. Process. Syst, pp. 4696-4705, 2019.

[24] H. Zhang, M. Cissé, Y. N. Dauphin and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization", Proc. Int. Conf. Learn. Representations, 2018.

[25] B. Barz and J. Denzler, "Deep learning on small datasets without pre-training using cosine loss", Proc. IEEE Winter Conf. Appl. Comput. Vis., pp. 1360-1369, 2020.

[26] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, 2016, [online] Available: http://www.deeplearningbook.org.

[27] C. Cochin, P. Pouliguen, B. Delahaye, D. Le Hellard, P. Gosselin, and F. Aubineau, "Mocem - an 'all in one' tool to simulate sar image," pp. 1 – 4, 07 2008.

[28] C. Cochin, J.-C. Louvigne, R. Fabbri, C. Le Barbu, A. O Knapskog, and N. Ødegaard, "Radar simulation of ship at sea using mocem v4 and comparison to acquisitions," 10 2014.

[29] Direct Support and General Support Maintenance Manual (Caterpillar Model D7F). Department of the Army Technical Manual, 1971.

[30] Doerry, Armin W. SAR Image Scaling Dynamic Range Radiometric Calibration and Display. United States: N. p., 2019. Web. doi:10.2172/1761879.

[31] B. Lewis, B. K. Cai, and C. Bullard. "Adversarial training on SAR images." Automatic Target Recognition XXX. Vol. 11394. International Society for Optics and Photonics, 2020.

[32] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

[33] Wong, E., Rice, L., & Kolter, J. Z. (2020). Fast is better than free: Revisiting adversarial training. arXiv preprint arXiv:2001.03994.

[34] Tobin et al. Domain randomization for transfering deep neural networks from simulation to the real world. IEEE IROS 2017

[35] Ulaby, F., Dobson, M. C., & Álvarez-Pérez, J. L. (2019). Handbook of radar scattering statistics for terrain. Artech House.

[36] M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[37] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[38] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[39] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In Soviet Mathematics Doklady, volume 27, pages 372–376, 1983

[40] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. ICML (3), 28:1139–1147, 2013.

[41] Smith, L. N., & Topin, N. (2019, May). Super-convergence: Very fast training of neural networks using large learning rates. In Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications (Vol. 11006, p. 1100612). International Society for Optics and Photonics.

[42] Smith, L. N. (2017, March). Cyclical learning rates for training neural networks. In 2017 IEEE winter conference on applications of computer vision (WACV) (pp. 464-472). IEEE.

[43] Xie, C., and Yuille, A. Intriguing properties of adversarial training at scale. In Proceeding ICLR 2020

Désentrelacement et classification des émetteurs RADARs basés sur l'utilisation des distances de Transport Optimal

Manon MOTTIER
Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des signaux et systèmes
91190, Gif-sur-Yvette, France
manon.mottier@centralesupelec.fr

Gilles CHARDON
Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des signaux et systèmes
91190, Gif-sur-Yvette, France
gilles.chardon@centralesupelec.fr

Frédéric PASCAL
Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des signaux et systèmes
91190, Gif-sur-Yvette, France
frederic.pascal@centralesupelec.fr

***Abstract*—Cet article introduit une nouvelle méthode pour désentrelacer et classifier les émetteurs d'un signal RADAR utilisant des distances de transport optimal. Une stratégie en deux étapes est développée et appliquée pour désentrelacer un signal RADAR : tout d'abord, un algorithme de clustering est utilisé pour séparer les données. Ensuite, les résultats (classes) du clustering initial sont incorporées dans un clustering agglomératif hiérarchique combiné à des distances de transport optimal utiliser pour fusionner (regrouper) les classes. Enfin, le processus d'identification est appliqué sur ces résultats du désentrelacement; la méthodologie est basée sur l'élaboration d'une distance entre un groupe d'impulsions et une description des caractéristiques d'un émetteur RADAR à partir d'une base de données de référence. La méthodologie est construite et évaluée sur des données simulées étiquetées.**

***Index Terms*—Transport Optimal, Désentrelacement, Classification, Processus de Reconnaissance RADAR, Guerre électronique, Machine Learning**

I. Introduction

Les évolutions technologiques de ces dernières années ont permis de nombreuses innovations dans le domaine de la guerre électronique : les équipements deviennent de plus en plus complexes et sophistiqués, entraînant une transformation de la chaîne de traitement du signal RADAR et plus particulièrement du processus de reconnaissance RADAR (amélioration des récepteurs et des émetteurs, étalement des émissions sur le spectre, modifications des formes d'onde, systèmes d'armement autonomes, etc.). Tous ces changements représentent un défi continu afin de proposer de nouvelles techniques plus précises pour classer et identifier les émetteurs RADAR à partir d'un signal reçu.

Le processus de reconnaissance RADAR peut se définir en deux étapes : la première étape consiste à désentrelacer un signal, c'est-à-dire à séparer et regrouper les impulsions mélangées d'un nombre inconnu d'émetteurs. Les premiers travaux de désentrelacement conventionnels sont principalement basés sur l'utilisation de 3 caractéristiques : la direction d'arrivée (DOA, *Direction-Of-Arrival*), la fréquence (F, *frequency*) et le temps d'arrivée (TOA, *Time-Of-Arrival*). La TOA/DOA et la F ont été utilisées dans un premier temps pour filtrer les données, puis les histogrammes du temps d'arrivée différencié (dTOA) pour identifier les motifs de répétition des impulsions [1]. Ces travaux ont servi de fondation et inspiré de nombreux auteurs afin de proposer de nouvelles méthodes. On retrouve notamment l'utilisation des histogrammes de dTOA cumulés [2] ou encore les histogrammes de différences séquentielles [3]. Ces méthodes fonctionnent très bien lorsque les RADARs présentent des caractéristiques simples comme émettre sur une bande de fréquence unique et de façon continue. Lorsque ces caractéristiques sont plus complexes, les méthodes sont le plus souvent basées sur des modèles de Deep Learning (voir *e.g/,* [4]) qui nécessitent une quantité considérable de données et qui sont difficiles à paramétrer. Toutes ces méthodes sont la plupart du temps construites dans un cadre supervisé.

La deuxième phase du processus de reconnaissance RADAR concerne l'identification des émetteurs. L'identification d'émetteurs RADAR à partir d'impulsions reçues représente un enjeu majeur pour le renseignement car cette étape permet de donner un avantage stratégique dans la prise de décision et l'engagement militaire. Pour les RADARs simples, l'analyse temps-fréquence est suffisante pour identifier l'émetteur avec une grande certitude [5]. Néanmoins, les émetteurs RADAR peuvent avoir des caractéristiques plus complexes; dès lors, des classifieurs supervisés issus des modèles de Deep Learning sont très largement utilisés pour faire face à cette problématique [6]–[10]. Ces méthodes nécessitent un volume important de données pour l'étape d'entraînement, ainsi qu'un réentraînement du classifieur lors de l'ajout d'une nouvelle classe et prennent en compte un petit nombre de classes à identifier.

L'acquisition de données réelles représente un véritable défi dans le traitement des données RADARs, particulièrement les données réelles étiquetées. Les algorithmes et les méthodologies sont souvent développés à l'aide de petits ensembles de données et/ou de données simulées. La plupart des méthodes précédentes sont basées sur des données

simulées et leurs performances dépendent fortement de la précision du simulateur. De plus, les développements technologiques ont modifié les profils des RADARs : les caractéristiques des émetteurs sont devenus plus complexes, enrichissant le panorama existant de nouveaux types d'émetteurs aux motifs plus variés. De nouvelles méthodes ont été développées afin de comparer les caractéristiques du groupe à une base de données connue, permettant également de détecter de nouveaux émetteurs [11], [12].

Pour pallier ces contraintes, nous avons construit un nouveau processus simple de reconnaissance RADAR non supervisé basé sur l'utilisation de distances de transport optimal. L'article est organisé comme suit : la première partie présente les données, suivie des sections 2 et 3 qui introduisent les méthodologies de désentrelacement des impulsions et de classification des émetteurs RADARs. La section 4 présente un cas d'application avant de conclure dans la section 5, et de proposer de nouvelles perspectives de recherches quant à la poursuite de ces travaux.

II. Description des données

Cette section présente les données utilisées à partir desquelles nous avons développé notre méthodologie ainsi que la procédure expérimentale mise en place pour obtenir des données étiquetées non confidentielles. Les labels sont ici utilisés uniquement pour évaluer les performances de la méthode développée, qui est par construction totalement non-supervisée. Nous avons eu recours à un simulateur de données construit et certifié par des experts du domaine. La méthodologie a également été validée sur des données réelles que nous avons à notre disposition. Ces données étant confidentielles, les résultats ne sont pas présentés dans cet article. Bien que les théâtres de la guerre électronique aillent de pair avec l'usage de contremesures, cet article se positionne dans un cadre ELINT et ne prend pas en compte la présente de mesures adverses introduisant des signaux perturbateurs. Les écarts de fréquence dûs à l'effet Doppler étant petits par rapport aux séparations fréquentielles entre les différents émetteurs, l'effet Doppler sera negligé dans notre analyse.

La diversité/complexité des signaux simulés se traduit par l'acquisition de signaux mono- ou multi-capteurs étiquetés, des signaux à modulation de fréquence et/ou temporelle, des signaux comportant des erreurs de mesure, des outliers, des données manquantes ou du bruit non-gaussien. Notre base de données contient des signaux hétérogènes de différentes tailles (allant de 20 à plus de 100000 impulsions) pouvant contenir plus de 10 émetteurs et avec des niveaux de bruit pouvant varier fortement. Les récepteurs sont supposés statiques et ont un seuil de détection connu, mais nous n'avons pas d'autres informations sur les caractéristiques des émetteurs et récepteurs. Les données simulées sont anonymisées mais étiquetées. Les récepteurs sont omnidirectionnels, rendant la direction d'arrivée inutilisable. Les impulsions acquises par le système de RADAR passif sont segmentées, analysées puis décrites par un ensemble limité de caractéristiques. Dans ce travail, les N impulsions acquises sont décrites uniquement à l'aide des quatre caractéristiques suivantes :

- t_n : Temps d'Arrivée (TOA, μs)
- d_n : Durée d'Impulsion (DI, ns),
- f_n : Fréquence (F, MHz),
- g_n : Niveau (LEVEL, dBm).

Nous ne prenons pas en compte d'autres informations sur les impulsions (comme par exemple la forme d'onde, la modulation de fréquence, etc.).

La Figure 1 présente un exemple de données simulées regroupant les impulsions de 4 émetteurs identifiables par leurs couleurs. La fréquence, la durée d'impulsion et le niveau sont représentés comme des fonctions du temps sur les 3 premiers graphiques et chaque point caractérise une impulsion. Les valeurs des caractéristiques ont été anonymisées de façon aléatoire. Les RADARs peuvent émettre de façon continue et régulière sur différentes fréquences comme souligné par l'émetteur 1 qui transmet sur des fréquences basses de façon constante. A l'inverse, l'émetteur 2 est présent sur des fréquences plus hautes et de façon plus ponctuelle. Les bandes de fréquences peuvent être partagées par des émetteurs ayant des caractéristiques proches comme les émetteurs 2 et 3, compliquant leur séparabilité. On retrouve fréquemment cette problématique lorsque le périmètre d'écoute concerne par exemple un port ou un aéroport où l'on retrouve des RADARs avec des caractéristiques très similaires. La simultanéité d'émission des RADARs peut conduire à une superposition et un mélange des lobes comme illustré dans le plan (g_n, t_n) pour les émetteurs 0, 1 et 3 qui sont actifs simultanément autour de 47 μs. Enfin, le plan (f_n, d_n) présenté par le dernier graphique montre qu'un RADAR n'est pas forcément représenté par un paquet d'impulsion unique mais au contraire, peut être scindé en plusieurs paquets d'impulsions comme les émetteurs 2 et 3.

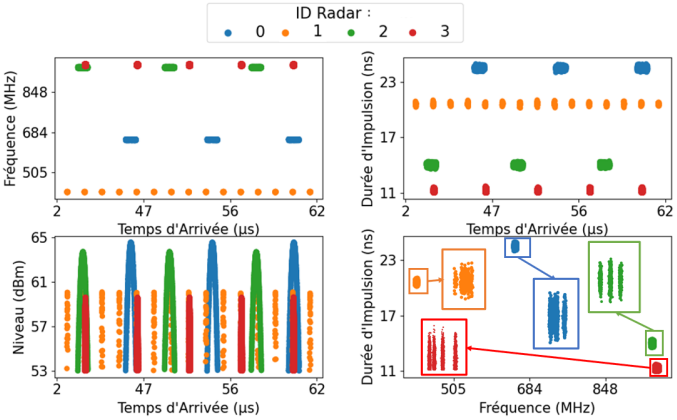


Fig. 1. Exemple d'un signal simulé regroupant les impulsions de 4 RADARs. Chaque couleur identifie un émetteur distinct.

III. Stratégie de désentrelacement d'un signal RADAR

Nous avons construit une stratégie de désentrelacement d'un signal RADAR en 2 étapes; premièrement un algorithme de

clustering préliminaire est utilisé pour séparer les impulsions d'un nombre inconnu de RADARs dans le plan (f_n, d_n). Puis, à partir de ces résultats, un algorithme de clustering agglomératif hiérarchique combiné à des distances de transport optimal est utilisé afin de regrouper les clusters appartenant à un même RADAR dans le plan (t_n, g_n).

A. Pré-traitement

Les dimensions physiques des données en fréquence (MHz) et en durée d'impulsion (ns) sont incohérentes. Une renormalisation est nécessaire pour calculer les distances entre les caractéristiques. Plusieurs méthodes ont été testées comme la normalisation par quantiles ou un pré-clustering via l'algorithme des GMM (Modèle de Mélange Gaussien) [13]. La méthode des quantiles consiste à utiliser les intervalles interquantiles pour mettre à l'échelle les données tandis que celle des GMM estime les variances intra-cluster par rapport aux caractéristiques considérées. Nous avons finalement choisi d'utiliser la méthode des quantiles car elle préserve au mieux la distribution des données et est très facilement implémentable.

B. Clustering dans le plan (f_n, d_n)

Les signaux reçus contiennent les impulsions mélangées d'un nombre inconnu d'émetteurs. Il est donc nécessaire de séparer ces impulsions et de les regrouper en clusters. Tout d'abord, nous appliquons un algorithme de clustering pour séparer les impulsions uniquement à partir de deux caractéristiques très discriminantes et fiables : la fréquence et la durée d'impulsion. Des algorithmes de clustering classiques tels que les K-Means [14] ou les GMM [13] sont fréquemment utilisés. Ces méthodes sont limitées car elles font des hypothèses sur le nombre de clusters à identifier ou sur leurs formes.

Nous avons choisi d'utiliser HDBSCAN [15] qui est particulièrement adapté à notre problématique. HDBSCAN est un algorithme de clustering non supervisé, basé sur une version hiérarchique de l'algorithme DBSCAN [16] capable de détecter des clusters de densités et de formes différentes. Le fonctionnement de l'algorithme est présenté plus en détail dans un précédent article [17]. L'idée principale est de regrouper des points qui vivent dans une même zone dense. Les résultats du clustering dépendent fortement de l'hyperparamètre déterminant la taille minimale d'un groupe pouvant être considéré comme un cluster. HDBSCAN a été paramétré de façon à surestimer le nombre de clusters retournés afin de ne pas mélanger les impulsions de plusieurs émetteurs dans un même cluster. Il est par exemple possible d'avoir des émetteurs avec seulement quelques impulsions ou émettant très peu dans le temps. Des comparaisons ont été effectuées afin de trouver un compromis pour fixer ce seuil.

Les résultats du clustering effectué par HDBSCAN dans le plan (f_n, d_n) sont affichés sur la Figure 2. L'algorithme identifie 9 clusters ainsi qu'une classe d'outliers (-1). Les clusters ont des tailles hétérogènes allant de 180 à plus de 2000 impulsions. Le plan (f_n, d_n) de droite montre que les résultats du clustering sont cohérents et que les impulsions des émetteurs 0 et 1 ont correctement été regroupées dans des clusters uniques. À l'inverse, HDBSCAN a scindé les impulsions des RADARs émettant autour au dessus de 948 MHz en plusieurs clusters. Le plan (g_n, t_n) de gauche met en évidence la distribution de ces clusters le long des lobes.

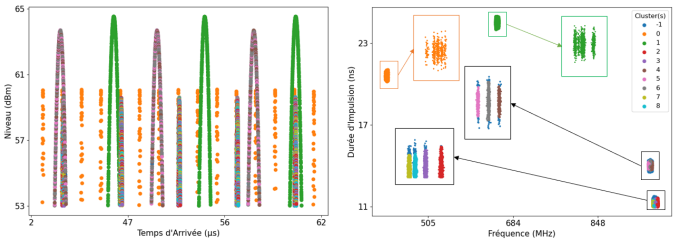


Fig. 2. Résultats du clustering d'HDBSCAN effectué à partir de la fréquence et la durée d'impulsion. Chaque couleur représentent les clusters ainsi qu'une classe d'outliers (-1).

C. Agglomération des clusters

Comme précédemment montré sur la Figure 1, il est possible qu'un RADAR soit représenté par plusieurs clusters; il est donc nécessaire de regrouper ces clusters. À partir des clusters d'HDBSCAN, un clustering agglomératif hiérarchique dans le plan (t_n, g_n) a été construit [18] en utilisant des distances de transport optimal comme illustré dans l'algorithme 1. Notre méthodologie est construite à partir d'une distance [19], [20] permettant de mesurer la similarité et la dissimilarité entre les clusters. La suite de l'article montrera que des distances de transport optimal sont particulièrement bien adaptées à notre problématique.

Algorithm 1 Clustering Agglomératif Hiérarchique combiné aux distances de transport optimal

- Représentation des clusters par une mesure à partir du temps d'arrivée et du niveau : τ.
- Calcul de la distance entre chaque cluster en utilisant le transport optimal : $d(\tau_i, \tau_j)$.
- Aggrégation des deux clusters les plus proches
- Mise à jour des distances.
- Réitération des étapes précédentes jusqu'à l'obtention d'un cluster unique.

Dans la suite de l'article, les distances de transport optimal seront calculées entre des distributions discrètes [21]. Considérons le cas de deux distributions de probabilité discrètes $\nu = \sum_{n=1}^{N} a_n \delta_{x_n}$ et $\mu = \sum_{m=1}^{M} b_m \delta_{y_m}$, avec $\mathbf{a} = (a_1, \dots a_N) \in \mathbf{R}_+^N$, $\sum_{n=1}^{N} a_n = 1$, et $\mathbf{b} = (b_1, \dots b_M) \in \mathbf{R}_+^M$, $\sum_{m=1}^{M} b_m = 1$. Un plan de transport $\mathbf{P}$ est défini par ses coefficients P_{nm}, qui modélisent la masse prise en x_n transportée en y_m. Ce transport à un coût par unité de masse $C_{nm} = c(x_n, y_m)$, où $c(\cdot, \cdot)$ est fonction de coût. Le coût total $C(\mathbf{P})$ d'un plan de transport est donc

$$C(\mathbf{P}) = \sum_{n=1}^{N} \sum_{m=1}^{M} C_{nm} P_{nm} \,. \tag{1}$$

La cohérence du plan de transport $\mathbf{P}$ avec les distributions de départ et d'arrivée est garantie par les conditions $\mathbf{P}\mathbf{1}_M = \mathbf{a}, \mathbf{P}^T\mathbf{1}_N = \mathbf{b}^T$. Le plan de transport optimal $\mathbf{P}^\star$ est obtenu par la résolution du problème d'optimisation linéaire suivant :

$$\mathbf{P}^\star = \underset{\mathbf{P} \in \mathbf{R}_+^{N \times M}}{\operatorname{argmin}} \; C(\mathbf{P}) \text{ tel que } \mathbf{P}\mathbf{1}_M = \mathbf{a}, \mathbf{P}^T\mathbf{1}_N = \mathbf{b}^T \,, \tag{2}$$

et la distance de transport optimal entre ν et μ est donnée par $d(\nu, \mu) = C(\mathbf{P}^\star)$. La détermination et la résolution du plan de transport optimal sont détaillés dans la toolbox POT [21], accessible sous python.

Les impulsions des clusters sont représentées par une mesure de probabilité décrivant la distribution de leur temps d'arrivée et de leur niveau :

$$\tau = \frac{1}{P} \sum_{p=1}^{P} \delta_{g_n, t_n} \,, \tag{3}$$

avec P le nombre d'impulsions du cluster. Afin de diminuer la complexité de calcul des distances, la mesure de probabilité utilisée en pratique est obtenue à partir d'un histogramme des données. Nous faisons l'hypothèse que les clusters appartenant à un même émetteur sont actifs simultanément et répartis le long des lobes comme précédemment illustré sur la Figure 2. Dans la stratégie d'agrégation, pour quantifier cette similarité ou cette dissemblance, nous avons utilisé une distance de transport optimal qui est défini comme le coût pour déplacer les points d'un endroit à un autre. Les clusters obtenus dans le plan appartenant au même émetteur auront entre leurs distributions une distance minime. Les deux clusters ayant la plus petite distance de transport optimal dans le plan (t_n, g_n) sont agrégés :

$$(i, j)^\star = \operatorname{argmin} d\left(\tau_i, \tau_j\right) . \tag{4}$$

Après la fusion, la distance entre les clusters fusionnés et les autres clusters est mise à jour. Puis, le processus est réitéré jusqu'à ce que les clusters soient entièrement agrégés. La figure 3 présente une itération du processus de fusion : le coût de transport entre les clusters 4 et 6 (orange et vert) sera très faible (les histogrammes de ces clusters étant très similaires) tandis que le coût entre les clusters 4 et 0 (orange et rouge) sera très élevé.

Le dendrogramme présente de façon simplifiée les agrégations des clusters réalisées à chaque étape du clustering agglomératif hiérarchique combiné aux distances de transport optimal. On peut voir sur la Figure 4 que les regroupements des clusters 2, 3, 7 et 8, représentés dans l'encadré vert, ont des coûts de transport faibles car tous ces clusters appartiennent au même RADAR. En revanche, le regroupement du cluster 0 avec les clusters 2, 3, 7, 8, représenté dans l'encadré rouge, aura un coût de transport très élevé, ce qui indique que le cluster 0 n'appartient pas au même émetteur que les clusters 2, 3, 7 et 8. Plusieurs métriques non supervisées (Silhouette score [22], Davies-Bouldin score [23], Calinski-Harabasz Score [24]) ainsi que la connaissance de l'environnement RADAR sont utilisées pour créer un critère d'arrêt afin de stopper les fusions. La ligne rouge indiquée sur la Figure 4 permet d'identifier les regroupements finaux : ici quatre ensembles d'impulsions sont retenus (cluster 0, cluster 1, regroupement des clusters 4,5,6 et regroupement des clusters 2, 3, 7, 8).

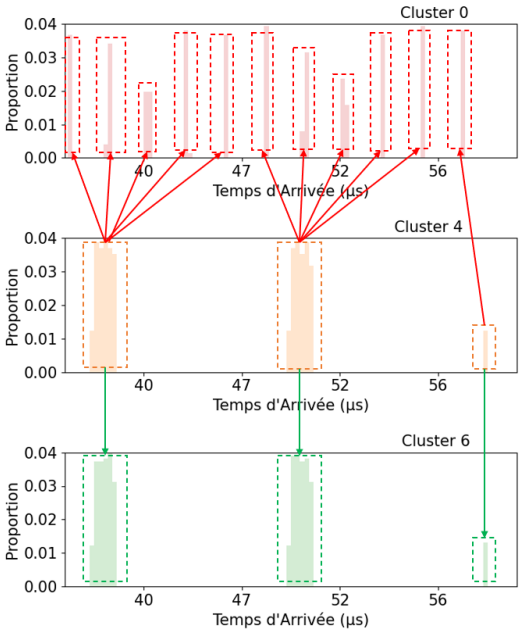


Fig. 3. Distribution des temps d'arrivée des clusters 0, 4 et 6 sous forme d'histogramme.

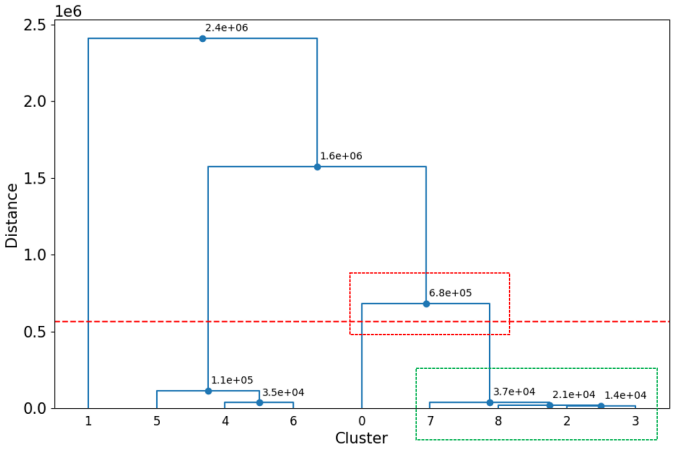


Fig. 4. Dendrogramme représentant les aggrégations à chaque étape du clustering agglomératif utilisant des distances de transport optimal.

Enfin, la Figure 5 illustre le résultat de l'agglomération des clusters. Ici le clustering a parfaitement fonctionné et identifie correctement les 4 émetteurs présents dans le signal. En particulier, les lobes présents autour de 47 μs ont parfaitement été reconstruits et le transport optimal a permis de regrouper correctement les différents clusters répartis sur ces lobes.

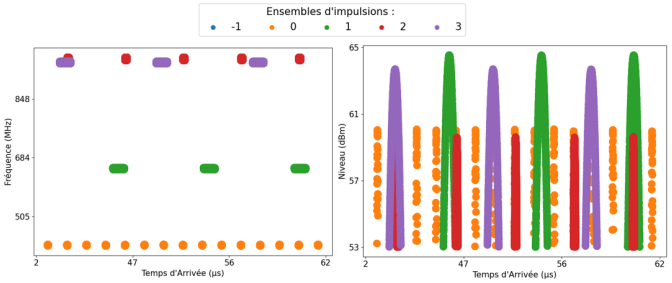


Fig. 5. Regroupements finaux. Les couleurs représentent les ensembles d'impulsions obtenus ainsi qu'une classe d'outliers (-1).

Afin d'évaluer la qualité des regroupements, nous avons utilisé 3 mesures :

- Homogénéité [25] : métrique permettant d'évaluer si toutes les impulsions d'un cluster appartiennent bien au même émetteur.
- Complétude [25] : métrique évaluant le niveau d'étalement des impulsions à travers les clusters.
- Taux d'outliers : mesure évaluant la proportion d'impulsions perdues durant le processus.

Les valeurs d'homogénéité et de complétude sont égales à 1, ce qui signifie que les impulsions sont correctement réparties dans les différents ensembles d'impulsions, et qu'il n'y a pas de mélange. Le taux d'outliers indique que 1.6% des impulsions sont perdues durant le processus de désentrelacement; ces impulsions ont été exclues par HDBSCAN à l'étape 1.

IV. Classification des émetteurs RADARs

Lorsque la phase de désentrelacement est terminée, la classification peut démarrer en utilisant les ensembles d'impulsions précédemment désentrelacés. La méthodologie est décrite dans l'algorithme 2. Elle est basée sur le développement d'une distance entre la description des émetteurs RADARs d'une base de données de référence et les ensembles d'impulsions désentrelacés. Dans notre cas, les ensembles d'impulsions et les classes d'émetteurs RADARs sont représentés sous forme de distributions de probabilités. La classification est ensuite effectuée en identifiant la classe d'émetteur RADAR la plus proche de celle de l'ensemble d'impulsions au sens d'une distance bien choisie. Lorsque l'émetteur RADAR possède des caractéristiques simples, comme par exemple une seule fréquence, des distances classiques telles que la distance euclidienne entre la moyenne des caractéristiques des impulsions reçues et les caractéristiques de l'émetteur de la base RADAR peuvent être utilisées. Néanmoins, dès lors que l'émetteur possède des caractéristiques plus complexes, il n'est pas possible de décrire la distribution des caractéristiques par de simples moyennes. Plusieurs méthodes existent afin de calculer une distance entre des distributions de probabilités comme la distance de variation totale ou la divergence de Kullback-Leibler. Dans notre cas, nos données sont représentées par des distributions discrètes ayant généralement des supports disjoints ce qui ne nous permet pas d'avoir recours à ces distances classiques. Les parties suivantes mettent en évidence l'intérêt de l'utilisation des distances de transport optimal pour répondre à notre problématique ainsi que leur robustesse au bruit.

Algorithm 2 Classification avec les distances de Transport Optimal.

- Construction d'une distribution de probabilité à partir des ensembles d'impulsions : ν
- Construction d'un mesure discrète à partir des classes d'émetteurs RADARs appartenant à une base de données de référence : μ_j
- Calcul du coût de transport entre la distribution de probabilité de l'ensemble d'impulsion et chaque classe d'émetteur : $d_j = d(\nu, \mu_j)$
- Assignation de la classe d'émetteur ayant le plus petit coût de transport entre sa distribution de probabilité et la distribution de l'ensemble d'impulsion : $j^\star$

A. Représentation des Classes d'émetteurs

La base de données RADAR utilisée pour faire la classification comprend les références de plus de 60 classes d'émetteurs distinctes. Certaines classes ont des caractéristiques très similaires tandis que d'autres sont très facilement distinguables. Nous avons choisi de travailler uniquement à partir de la fréquence et de la durée d'impulsion car ce sont des caractéristiques très discriminantes et fiables. À partir de cette base de données, nous avons construit une mesure décrivant chaque classe d'émetteur :

$$\mu_j = \sum_{n=1}^{N} \alpha_n \delta_{f_n, d_n} \,, \tag{5}$$

avec N le nombre de couples de fréquences et de durées d'impulsion sur lesquelles le RADAR émet, α la proportion d'impulsions dans chaque couple, et δ, la masse de Dirac (avec $\sum_{n=1}^{N} \alpha_n = 1$).

La Figure 6 illustre un exemple de classes d'émetteurs simulés. Chaque tige représente une fréquence utilisée par un émetteur et sa hauteur la proportion d'apparition de cette fréquence. Certains RADARs émettent sur des bandes de fréquences très différentes comme les émetteurs D et I tandis que d'autres ont des caractéristiques très proches comme les émetteurs E et F. Les RADARs peuvent être mono fréquence comme l'émetteur K qui émet également sur une fréquence commune à celle de l'émetteur E; cette similarité fréquentielle ne permet pas de les différencier. A l'inverse, tout comme l'émetteur A, les RADARs peuvent être multifréquences.

La Figure 7 montre les précédents émetteurs dans le plan (f_n, d_n). Les émetteurs K et E sont clairement séparés dans ce plan, ce qui nous indique que l'ajout de caractéristiques supplémentaires améliore la séparabilité des émetteurs.

La matrice de coût représentée sur la Figure 8 est construite à partir de la fréquence et de la durée d'impulsion. Elle présente de façon simplifiée les coûts de transport

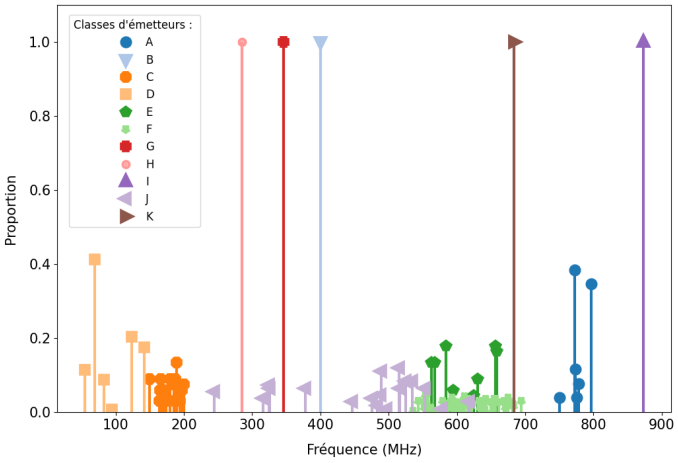


Fig. 6. Représentation de classes d'émetteurs simulées. Les émetteurs sont représentés en une dimension dans le plan fréquentiel. Chaque couleur représente une classe.

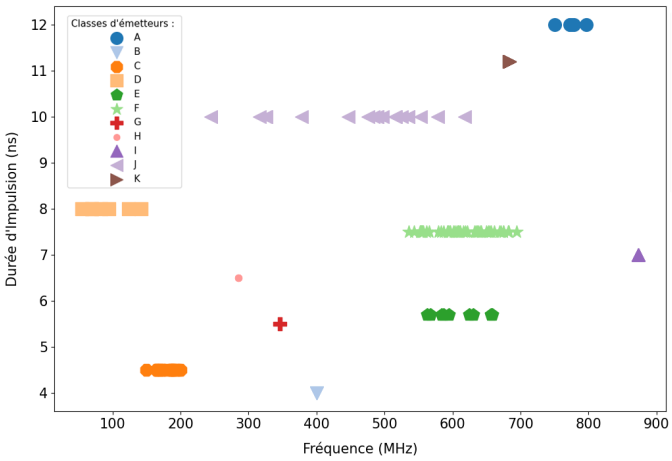


Fig. 7. Représentation des classes d'émetteurs simulées précédentes. Les émetteurs sont représentés en deux dimensions dans le plan F-DI. Chaque couleur représente une classe.

entre les émetteurs simulés précédents. Les couleurs caractérisent le niveau de proximité entre les émetteurs. Comme précédemment expliqué, certaines classes peuvent avoir des caractéristiques similaires. À titre d'exemple, les classes E et F sont séparées par une petite distance à l'inverse des classes A et C pour lesquelles elle est plus grande. Une attention particulière est portée sur les classes ayant des petites distances; on s'attend à ce qu'elles soient fréquemment confondues car elles ont des caractéristiques observables similaires.

La Figure 9 représente le plan de transport de l'émetteur F vers les émetteurs A, E, G, J et K. Le coût de transport des points de l'émetteur F vers l'émetteur E est faible car tous les points sont déplacés vers un emplacement très proche : les deux émetteurs émettent sur des bandes de fréquences très similaires et ont une durée d'impulsion proche. À l'inverse, le coût de transport de l'émetteur F vers l'émetteur G est 20 fois plus élevé car tous les points sont déplacés vers une bande de

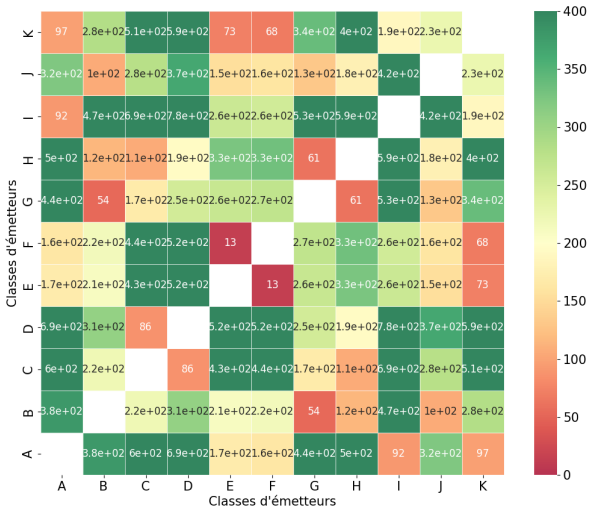


Fig. 8. Matrice de coûts entre les classes d'émetteurs simulées construite à partir de la fréquence et de la durée d'impulsion. Une couleur verte indique un coût de transport faible tandis que le rouge représente un coût élevé.

fréquence unique et très différente; les points doivent effectuer des déplacements plus importants. La distance de transport optimal tient compte de la proportion d'impulsions pour un couple donné de fréquence et de durée d'impulsion.

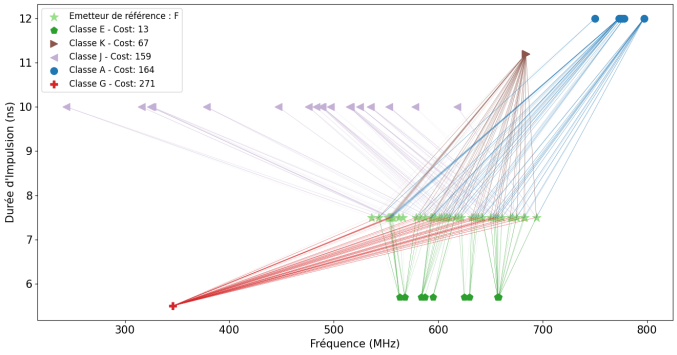


Fig. 9. Plans de transport des points de l'émetteur F vers les émetteurs A, E, G, J, K avec le transport optimal en deux dimensions.

B. Modèle de classification

Les ensembles d'impulsions sont modélisés sous forme de distributions de probabilité de la façon suivante :

$$\nu = \frac{1}{M} \sum_{m=1}^{M} \delta_{f_m, d_m} \,, \tag{6}$$

avec M le nombre d'impulsions de l'ensemble. Les ensembles d'impulsions sont également regroupés en intervalles de fréquences et de durée d'impulsion. L'algorithme identifie la vraie classe d'émetteur $j^\star$:

$$j^\star = \operatorname{argmin} d\left(\mu_j, \nu\right). \tag{7}$$

La classification est faite en attribuant à l'ensemble d'impulsion la classe d'émetteur RADAR la plus proche en termes de distance au sens du transport optimal.

C. Résultats

Le résultat du classifieur appliqué à l'ensemble d'impulsions 1 est illustré sur la Figure 10. Les impulsions et les trois classes d'émetteurs les plus proches sont superposées sur le graphique de gauche. Les points de la classe la plus proche se superposent parfaitement à ceux des données. Le classifieur identifie correctement l'émetteur présent dans cet ensemble; ce résultat est confirmé par le fait que la distance vers la deuxième classe la plus proche est 20 fois plus élevée. Les plans de transports entre la distribution des données et les trois classes les plus proches [21] sont affichés sur le graphique de droite. La deuxième classe représente un RADAR transmettant sur six fréquences différentes tandis que les données sont caractérisées par seulement 5 fréquences : les points des données sont envoyés sur les différentes fréquences de classe 2 en respectant les proportions; c'est pourquoi les impulsions autour d'une fréquence donnée ne sont pas toutes envoyées au même point. Enfin, bien que la troisième classe semble plus proche dans le plan, ces fréquences sont très différentes de celles de l'ensemble d'impulsion; les points doivent donc effectuer de plus grands déplacements, ce qui augmente considérablement le coût de transport.

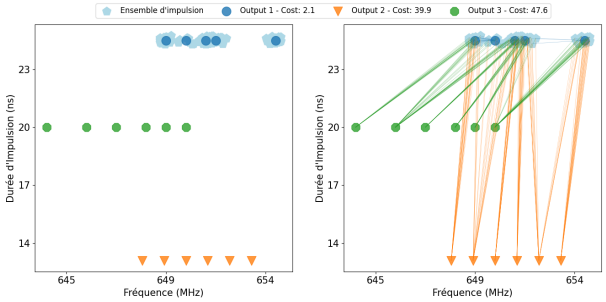


Fig. 10. Résultats de la classification pour l'ensemble d'impulsion 1.

La Figure 11 montre de façon analogue le résultat de notre méthodologie de classification appliquée à l'ensemble d'impulsions 3. Le graphique de gauche superpose les impulsions et les trois classes d'émetteurs les plus proches. Les points bleus se superposent également très bien à ceux des données. Le classifieur identifie correctement l'émetteur présent. Cependant, l'écart de distance entre l'ensemble d'impulsion et les sorties 1 et 2 est plus faible que celui de l'exemple précédent car les classes d'émetteurs 1 et 2 émettent sur des durées d'impulsion proches.

V. Cas d'application

Cette section présente un nouveau signal simulé à partir duquel nous avons appliqué le processus de reconnaissance RADAR. Ce signal contient environ 8000 impulsions et est présenté sur la Figure 12. La représentation fréquentielle met en évidence la simultanéité temporelle de plusieurs émetteurs autour de 27 μs. Certains émetteurs transmettent de façon continue et régulière comme celui à 1020 MHz à l'inverse du RADAR présent après 27 μs qui lui semble n'apparaître qu'une fois sur notre simulation et être multifréquences. Le

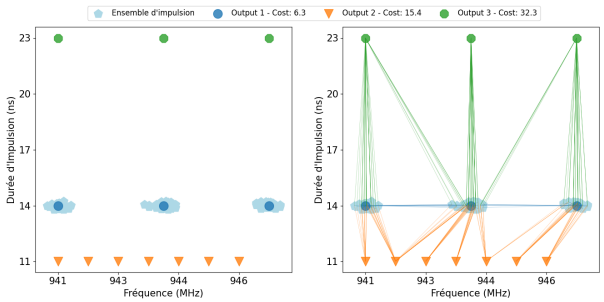


Fig. 11. Résultats de la classification pour l'ensemble d'impulsion 3.

plan (d_n, t_n) est caractérisé par un large étalement de la durée d'impulsion compliquant l'analyse et ne permettant pas de distinguer les émetteurs. De plus, la superposition des lobes sur le plan (g_n, t_n) rend la séparation des émetteurs difficile.

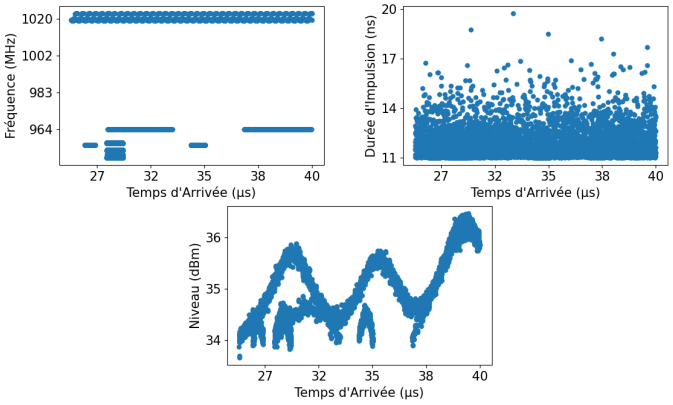


Fig. 12. Signal simulé regroupant les impulsions d'un nombre inconnu d'émetteurs RADAR.

A. Désentrelacement des impulsions

La première étape du désentrelacement consiste à séparer les impulsions à partir de la fréquence et de la durée d'impulsion grâce à l'algorithme de clustering HDBSCAN. La Figure 13 affiche les résultats retournés par HDBSCAN. L'algorithme identifie 7 clusters ainsi qu'une classe d'outliers (-1). Les impulsions des émetteurs 0 et 1 sont correctement séparés dans 2 clusters distincts. En revanche, les impulsions de l'émetteur à 11 ns sont scindées en plusieurs clusters. Il est donc nécessaire de regrouper ces clusters. Le plan (f_n, d_n) illustre la distribution de ces clusters le long du lobe autour après 27 μs.

La Figure 14 affiche les regroupements finaux des clusters effectués à partir du dendrogramme. Le modèle décisionnel identifie quatre ensembles d'impulsions distincts. Le plan (f_n, d_n) montre la présence de deux émetteurs différents transmettant à 11 ns. Notre méthode d'agglomération a été capable de distinguer ces deux émetteurs et stopper les fusions pour ne pas regrouper leurs impulsions.

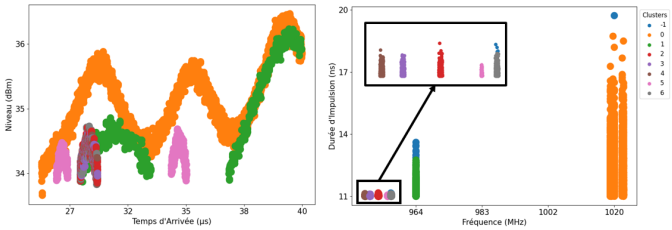


Fig. 13. Résultats du clustering HDBSCAN effectué dans le plan F-DI. Chaque couleur représente un cluster ainsi qu'une classe d'outliers (-1).

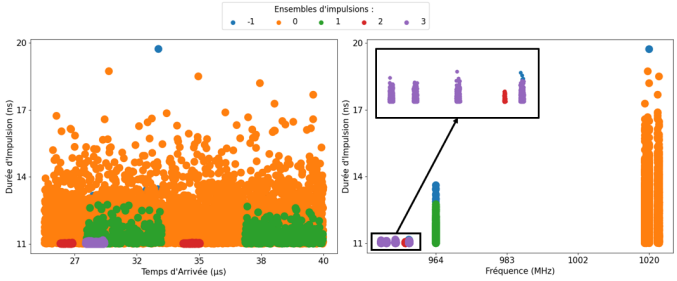


Fig. 14. Regroupements finaux. Les couleurs représentent les ensembles d'impulsions obtenus ainsi qu'une classe d'outliers (-1).

B. Classification de l'ensemble d'impulsion 3

Les résultats de la classification appliquée à l'ensemble d'impulsion 3 sont illustrés par la Figure 15. Le plan (f_n, d_n) de gauche superpose les données et les trois premières classes d'émetteurs identifiées par l'algorithme. Les données de l'output 1 correspondent bien au niveau fréquentiel. En revanche, on aperçoit un décalage sur la durée d'impulsion qui s'explique par une erreur de mesure provenant des capteurs. Dans le cas de notre ensemble d'impulsions, cette erreur est négligeable, d'autant plus que les données sont fréquentiellement variées. La très faible valeur du coût de transport vers la premère classe (< 1) confirme les résultats de l'identification; sur le plan de transport de droite, les données de l'ensemble d'impulsion effectuent très peu de mouvements pour coïncider avec celles de la première classe tandis que ce coût est 19 fois plus élevé pour la deuxième classe la plus proche.

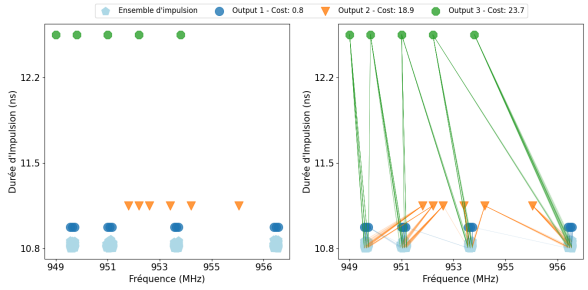


Fig. 15. Résultats de la classification pour l'ensemble d'impulsions 3.

VI. Conclusion et Perspectives

A. Conclusion

Dans cet article nous avons développé un processus de reconnaissance RADAR en deux étapes basé sur le développement et l'utilisation de distances de transport optimal à partir des paramètres primaires des RADARs. L'algorithme HDBSCAN a été appliqué sur de la fréquence et la durée d'impulsion pour séparer les impulsions en différents clusters. Puis, les clusters appartenant à un même émetteur ont été regroupés grâce à l'intégration des distances de transport optimal dans un clustering agglomératif hiérarchique construit à partir du temps d'arrivée et du niveau. Ensuite, en considérant qu'un ensemble d'impulsions correspondait à un seul émetteur, la méthode proposée a permis de classifier l'émetteur grâce à la théorie du transport optimal en deux dimensions à partir de la fréquence et de la durée d'impulsion. Les résultats obtenus sur les données simulées sont très encourageants et permettent d'identifier avec confiance la classe de l'émetteur tout en considérant un grand nombre de classes.

B. Perspectives

Dans la continuité de la méthodologie proposée, nous travaillons actuellement sur l'amélioration de plusieurs axes : tout d'abord, lors de la phase de désentrelacement, nous supposons que les clusters retournés par HDBSCAN contiennent les observations d'un seul émetteur mais il est possible que les RADARs puissent avoir des caractéristiques semblables et être confondus en fréquence et en durée d'impulsion. C'est le cas dans les ports ou les aéroports où plusieurs modèles similaires sont utilisés. Nous travaillons actuellement sur l'amélioration de la séparabilité de ces émetteurs en incorporant d'autres caractéristiques dans le clustering.

Concernant la classification, plusieurs perspectives sont en cours d'investigation pour mieux discriminer les RADARs : premièrement, l'ajout d'une troisième dimension dans le calcul des distances de transport optimal, tout particulièrement la période de répétition des impulsions (PRI), qui est défini comme la différence de temps d'arrivée entre des impulsions successives. Cette caractéristique n'est pas directement utilisable car elle nécessite un prétraitement pour être exploitée : les impulsions manquantes impliquent des distorsions importantes de la densité de probabilité des PRI. Ensuite, comme mentionné dans ce travail et illustré sur la Figure 15, les erreurs provenant de la durée d'impulsion réduisent les performances de la méthode. La robustesse de cette méthode vis-à-vis de telles erreurs doit être améliorée afin de prendre en compte des taux de mitage variables ou encore des erreurs de mesure importantes sur certains paramètres. Enfin, pour proposer une stratégie de classification complète, nous travaillons sur le développement d'une méthode capable de détecter des émetteurs non présents dans la base de données.

Remerciements

ATOS a soutenu ce travail en fournissant les données et son expertise RADAR.

REFERENCES

[1] D. Wilkinson and A. Watson, "Use of metric techniques in ESM data processing," in *IEE Proceedings F (Communications, Radar and Signal Processing)*, vol. 132, no. 4. IET, 1985, pp. 229–232.

[2] H. Mardia, "New techniques for the deinterleaving of repetitive sequences," in *IEE Proceedings F (Radar and Signal Processing)*, vol. 136, no. 4. IET, 1989, pp. 149–154.

[3] D. Milojević and B. Popović, "Improved algorithm for the deinterleaving of radar pulses," in *IEE Proceedings F (Radar and Signal Processing)*, vol. 139, no. 1. IET, 1992, pp. 98–104.

[4] Z. Zhou, G. Huang, H. Chen, and J. Gao, "Automatic radar waveform recognition based on deep convolutional denoising auto-encoders," *Circuits, Systems, and Signal Processing*, vol. 37, no. 9, pp. 4034–4048, 2018.

[5] A. Kawalec and R. Owczarek, "Radar emitter recognition using intrapulse data," in *15th International Conference on Microwaves, Radar and Wireless Communications (IEEE Cat. No. 04EX824)*, vol. 2. IEEE, 2004, pp. 435–438.

[6] J. Lunden and V. Koivunen, "Automatic radar waveform recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 1, pp. 124–136, 2007.

[7] Z.-M. Liu and S. Y. Philip, "Classification, denoising, and deinterleaving of pulse streams with recurrent neural networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 4, pp. 1624–1639, 2018.

[8] Z. Geng, H. Yan, J. Zhang, and D. Zhu, "Deep-learning for radar: A survey," *IEEE Access*, vol. 9, pp. 141 800–141 818, 2021.

[9] L. Ding, S. Wang, F. Wang, and W. Zhang, "Specific emitter identification via convolutional neural networks," *IEEE Communications Letters*, vol. 22, no. 12, pp. 2591–2594, 2018.

[10] M. A. Nuhoglu, Y. K. Alp, and F. C. Akyon, "Deep learning for radar signal detection in electronic warfare systems," in *2020 IEEE Radar Conference (RadarConf20)*. IEEE, 2020, pp. 1–6.

[11] J. Liu, J. P. Lee, L. Li, Z.-Q. Luo, and K. M. Wong, "Online clustering algorithms for radar emitter classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1185–1196, 2005.

[12] S. Apfeld and A. Charlish, "Recognition of unknown radar emitters with machine learning," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 4433–4447, 2021.

[13] G. J. McLachlan and K. E. Basford, *Mixture models: Inference and applications to clustering*. M. Dekker New York, 1988, vol. 38.

[14] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[15] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2013, pp. 160–172.

[16] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[17] M. Mottier, G. Chardon, and F. Pascal, "Deinterleaving and clustering unknown RADAR pulses," in *2021 IEEE Radar Conference (RadarConf21)*. IEEE, 2021, pp. 1–6.

[18] S. Chakraborty, D. Paul, and S. Das, "Hierarchical clustering with optimal transport," *Statistics & Probability Letters*, p. 108781, 2020.

[19] C. Villani, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[20] N. Bonneel, M. Van De Panne, S. Paris, and W. Heidrich, "Displacement interpolation using lagrangian mass transport," in *Proceedings of the 2011 SIGGRAPH Asia conference*, 2011, pp. 1–12.

[21] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer, "POT: Python optimal transport," *Journal of Machine Learning Research*, vol. 22, no. 78, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-451.html

[22] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[23] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

[24] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[25] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 410–420.

Détection de navires embarquable à bord de satellites

T. Goudemant[1], B. Francesconi[1], H. Farhat[1], L. Daniel[1], O. Thiery[2], E. Kervennic[1], S. Mzoughi[1], A. Girard[1]

[1]*Institut de Recherche Technologique Saint-Exupéry - France*
[2]*Geo4i - France*
Contact: thomas.goudemant@irt-saintexupery.com

***Abstract*—La surveillance maritime répond à d'importants enjeux à la fois écologiques, économiques et sécuritaires. Dans ces domaines, les systèmes d'alerte doivent être réactifs et autonomes. C'est dans ce cadre que cet article explore la faisabilité d'une solution logicielle de détection de navires embarquable sur une électronique de satellite d'observation haute-résolution en orbite-basse. L'article décrit plusieurs implémentations sur FPGA et GPU de cette solution basée sur le détecteur YOLOv3 entrainé sur une base de données inédite d'images annotées par un photo-interprète.**

***Index Terms*—Détection optique de navires, Imagerie satellite, IA embarquée, Surveillance autonome, Pêche illégale**

I. Introduction

A. Télédétection spatiale

L'observation de la Terre depuis l'espace permet de nombreuses applications aussi diverses que de la mesure du champ de gravité [6], la détection précoce de tempête [13], l'évaluation de la déforestation [16] ou de la pauvreté [8]. Notre article se focalise sur la surveillance maritime qui répond à des enjeux à la fois sécuritaires, écologiques, économiques :

- la stratégie militaire
- le contrôle du trafic maritime, dont la minimisation des risques, évitement de bateaux, détection de manoeuvres anormales [19]
- le contrôle des activités en mer, légales et illégales telles que le dégazage sauvage d'hydrocarbures [5], le transport de stupéfiants ou armes, les actes de pirateries, et la pêche illégale.

Concernant la pêche illégale, non déclarée et non réglementée, nommée IUU (*"Illegal, Unreported and Unregulated fishing"*), elle représenterait environ 19% des volumes de pêche et 10 milliards d'euros chaque année dans le monde [3]; cette pratique sauvage impacte la biodiversité et les 3 milliards de personnes qui dépendent de la pêche [4]. Normalement, les navires sont équipés du système radio d'identification AIS (*"Automatic Identification System"*), mais ce système peut être manuellement désactivé voire modifié afin de diffuser de fausses déclarations d'identité ou d'activité ; ces déclarations sont difficilement vérifiables car les moyens terrestres et aériens ne peuvent couvrir l'ensemble des surfaces maritimes. La surveillance par satellites permet ainsi de palier cette déficience et donne en plus de riches informations sur l'environnement. Télédétecter ces activités illégales de pêche ou transbordement n'est possible qu'avec des satellites de haute résolution spatiale, comme indiqué dans la table I. La résolution spatiale (GSD en anglais) désigne la distance (en mètres) au sol entre deux pixels d'une image satellite. Cette image peut avoir été acquise par un radar, SAR [21] en anglais, ou une caméra optique ; dans cet article, le mot image désigne les données acquises par une caméra optique. En plus de sa résolution spatiale, il est important d'en connaître sa résolution spectrale pour savoir combien et quelles couleurs seront captées par la caméra. Dans notre étude, les images proviennent de caméras percevant le Rouge, Vert et Bleu, RGB en anglais, et une quatrième bande spectrale proche-infrarouge que nous n'avons pas utilisée.

TABLE I
TÂCHES DE TYPE DRI (DÉTECTION / RECONNAISSANCE / IDENTIFICATION) RÉALISABLES PAR PHOTO-INTERPRÉTATION SELON LA GAMME DE RÉSOLUTION SPATIALE, POUR DES IMAGES OPTIQUES RGB

Résolution	Fonction	Remarques
2-15m	Détection (sauf petits bateaux)	Les navires de commerce et les grands navires de combat sont détectés sans pouvoir spécialement donner leur fonction
0.7-2 m	Reconnaissance	Les navires de commerce et les grands navires de combat sont détectés et il est possible de donner leur fonction. Les petits navires sont détectés sans pouvoir donner leur fonction.
0.7 m	Reconnaissance (Identification)	Les petits navires sont détectés et il est possible de donner leur fonction. L'identification (nom/immatriculation) n'est en général pas possible sauf cas particulier (navires remarquables connus)

B. Objectifs généraux des travaux et périmètre des résultats présentés

Les travaux présentés dans la suite de cet article sont menés dans le cadre du projet CIAR (Chaine Image Autonome et Réactive) de la branche Smart Technologies de l'IRT Saint-Exupery, dont l'objectif principal est le développement d'algorithmes de traitements d'images par IA embarquée sur cibles matérielles. Cet objectif est décliné en trois axes techniques de travail mis en oeuvre sur différents cas d'applications :

- Axe base de données d'images + annotations : s'approprier des bases de données existantes ou en

constituer de nouvelles répondant aux besoins des cas d'applications choisis (degré de confiance/qualité, représentativité, nombre d'images, etc) et aux besoins futurs des partenaires du projet.
- Axe algorithmique : concevoir et tester des algorithme d'IA pouvant être exécutés sur cibles matérielles à capacités limitée (calcul, puissance, mémoire...), adaptés aux images brutes en sortie capteur et robustes à la variabilité opérationnelle
- Axe portage sur cible : Maitriser les outils et le savoir-faire (ex : techniques de quantification des réseaux) pour le déploiement d'IA sur cible, optimiser les performances matérielles (débit, latence, consommation, empreinte sur cible...)

La suite de l'article présente les résultats préliminaires obtenus sur le cas d'application spécifique qu'est "la détection et la reconnaissance automatiques de navires dans des images satellites optiques RGB à l'aide de réseaux de neurones embarqués sur cibles matérielles".

C. Plan de l'article

L'article est structuré autour des trois grandes étapes du développement d'une Intelligence Artificielle embarquée :

- Section II : Description de la base de données d'images utilisée pour l'entraînement et l'évaluation des performances algorithmiques.
- Section III : Description de l'algorithme choisi, YOLOv3, de son entraînement et de ses performances obtenues avant déploiement.
- Section IV : Description des cibles électroniques utilisées, du déploiement sur cibles et performances embarquées.

II. Base de données

Les algorithmes d'apprentissage, dont font partie les réseaux de neurones, nécessitent de grandes quantités d'exemples pour apprendre à réaliser une tâche donnée (ex : détection d'objet) dans un contexte donné (ex : télédétection de navires). Un exemple est généralement constitué d'une donnée à traiter (ex : image de mer contenant des navires) associée à une annotation contenant l'information que doit retourner l'algorithme (ex : positions des navires dans l'image et leurs types). La performance et la robustesse des algorithmes reposent donc en grande partie sur la qualité et la représentativité de la base de données d'images annotées utilisée pour l'apprentissage. La construction des bases de données est donc une étape critique dans le développement de solutions basées sur ce type d'algorithmes, nécessitant, selon l'application, une forte expertise dans le domaine visé. Dans le cas de la détection ou reconnaissance de navires par imagerie spatiale, l'annotation précise des images nécessite l'expertise d'un photo-interprète.

Quelques bases de données d'images haute-résolution utilisables pour de la détection ou reconnaissance de navires dans des images satellites optiques sont accessibles publiquement [14]. Malheureusement, aucune de ces bases de données ne répondait à l'ensemble des exigences suivantes :

- Localiser précisément les navires, à minima boite englobante orientées autour des navires
- Fournir les informations concernant la résolution des images, le type de capteur et les traitements appliquées
- Besoin de pouvoir contrôler la taille des images soumises aux algorithmes pour optimiser leurs performances sur cible embarquée.

Par conséquent, la construction d'une base de données de grande qualité a été confiée à la société GEO4I, partenaire du projet CIAR et spécialiste en fourniture et analyse d'images satellites. Ainsi, nous avons généré de manière incrémentale une base de données d'images sources (RGB, et proche infrarouge) de grandes dimensions (> 10 à 100s Mpixels) et à très haute résolution (30 à 50cm), annotées à un niveau de détails inédits par les analystes GEO4I grâce a des outils propriétaires adaptés à ce type d'images. Le tableau II synthétise les caractéristiques principales de cette base de données. Les images commandées sont issues du catalogue Maxar Imagery Products, corrigées au sol et donc non représentatives des images en sortie capteur à bord du satellite. Néanmoins, pour pouvoir simuler dans le futur les spécificités des images bord, les niveaux de corrections demandés ont été réduits au minimum.

Actuellement un tiers des navires est annoté avec le niveau de classification le plus détaillé (50 classes). Les autres navires sont annotés avec 10 classes. Une mise à jour de la base de données en 2022 mettra à niveau les annotations de ces navires. L'intérêt d'augmenter le niveau de spécificité des classes est double : d'une part cela permet de mieux ségréguer des navires aux caractéristiques communes et donc de faciliter la classification (par exemple il a priori est plus facile et naturel de reconnaître un cargo qu'un 'navire de commerce', classe abstraite regroupant des navires très différents) ; d'autre part l'intérêt opérationnel est accru car en identifiant la fonction ou le type précis du navire on est capable de dire si sa position ou son activité en cours semble légitime (surveillance) ou encore, dans le cas d'un navire militaire, cela permet d'avoir une idée de sa dangerosité. La figure 1 illustre quelques-une des classes de navires identifiées dans les images sources.

TABLE II
Caractéristiques la base de donnée source

Superficie totale	nb. de zones	Résolution des images	nb. total de navires annotés	nb. de classes
$\sim 2200 km^2$	47	0.3 - 0.5 m	$\sim$15900	$\sim$50

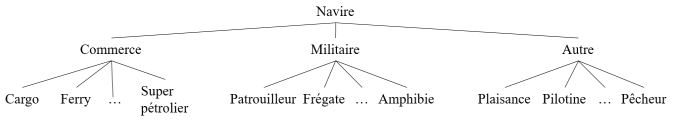


Fig. 1. Exemples de classes utilisées pour l'annotation des navires. La base de données contient environ 50 classes distinctes de navires

A partir de cette base de données source, GEO4I a généré des bases de données d'entraînement/validation/test nécessaires pour les algorithmes (voir Figure 2). L'outil utilisé par GEO4I est un SaaS (Software as a Service) collaboratif dédié à l'annotation d'images satellites ou d'images géoréférencées. Il a été développé initialement pour les besoins de la société afin de pouvoir créer et générer des bases de données d'apprentissage sur mesure. Il permet le détourage manuel d'objets, l'affectation de classes multiples (en effet un super pétrolier est également un navire et aussi un navire de commerce) et la génération automatique de bases de données d'apprentissage de façon paramétrable en termes de classes, de dimensions des images d'apprentissage ainsi que du type de détourage des navires (boites pour de la détection ou masques pour des besoins de segmentation).

Les travaux présentés dans cet article s'appuient sur une version préliminaire de la base de données d'images sources contenant environ 8000 navires distincts à des résolutions allant de 1.2 à 2m. Dans les images, les navires ont des tailles allant de quelques pixels à environ 350 pixels. Les scènes annotées ont été découpées en vignettes de taille 416x416 pixels, compatible des ressources mémoire des cibles hardware. Un recouvrement de 50% entre vignettes voisines et d'autres techniques de data-augmentation appliquées par le modèle lors de l'entraînement (redimensionnement, rognage et rotation aléatoire des images ainsi que distorsion des couleurs) sont appliqués pour augmenter artificiellement le nombre de navires et la variabilité des images. Les 2349 images résultantes ont été partitionnées aléatoirement en trois sous-ensembles :

- un jeu pour l'entraînement du modèle
- un jeu de validation: ces données ne sont pas utilisées pour entraîner le modèle mais permettent de valider que le réseau arrive à généraliser sur des données nouvelles. Ce jeu permet d'arrêter l'entraînement avant que le modèle "sur-apprenne" sur le jeu d'entraînement
- un jeu de test qui est utilisé pour évaluer les performances du modèle après l'entraînement.

Les tailles relatives des trois datasets ainsi que le nombre de bateaux annotés sont indiqués dans le tableau III.

TABLE III
RÉPARTITION DU DATASET EN TROIS SOUS-ENSEMBLES

	pourcentage du dataset	nombre d'imagettes	nombre d'annotations
train set	60%	1410	20825
val. set	20%	469	6847
test set	20%	470	6717

Afin de se concentrer sur le portage sur cible et simplifier le problème de reconnaissance, plus difficile à cette résolution, ces travaux préliminaires quantifient uniquement les performances de détection. Afin d'être représentatif de la majorité des capteurs haute résolution existants, seules les bandes RGB ont été utilisées pour ces travaux. Cela facilite également la réutilisation de la majorité des réseaux de neurones, conçus la plupart du temps pour des images RGB grand public. L'objectif de futurs travaux sera d'exploiter tout le potentiel offert par cette base de données en visant des tâches de reconnaissance détaillée des navires.

III. MODÈLE DE DÉTECTION DE BATEAUX

A. Choix de l'algorithme

Comme dans la plupart des applications embarquées, les algorithmes et les cibles matérielles doivent respecter les contraintes liées à leur environnement, en particulier dans un contexte spatial.

Premièrement, ces applications sont caractérisées par le besoin d'une faible consommation énergétique ainsi que des capacités mémoires et de calculs réduites par rapport aux applications opérées au sol. De plus, dans le cadre de la détection de bateaux, des images hautes résolutions (≤ 50 cm) seront acquises à très haut débit (≥ 500 MPixels/s), imposant une contrainte embarquée forte sur le temps de traitement par pixel. Finalement, des contraintes liées au déploiement de l'algorithme sur cibles (GPU, FPGA et TPU) sont également à prendre en compte. L'architecture et les couches du réseau devront être compatibles avec les outils de déploiement, limitant le portage des réseaux les plus récents. De plus, la taille du réseau doit être facilement adaptable aux capacités de la cible.

L'entraînement du réseau s'effectuant sur un serveur, caractérisé par de grosses capacités de calculs, ce sont principalement les ressources à l'inférence qu'il faut optimiser. Afin de maximiser le ratio performance de détection/temps d'inférence, et en considérant les contraintes de déploiement lors du portage, le réseau YOLOv3 [18] a été sélectionné. Ce modèle étant constitué uniquement de convolutions, et non de couches avec une dimension d'entrée prédéfinie (pooling / couches denses / etc.), il accepte une taille d'image variable en entrée. Il présente ainsi l'avantage d'être adaptable aux capacités de la cible d'exécution. YOLOv3 a inspiré de récente architecture telles que EnhancedYOLOv3Tiny [15], YOLO-Ship [22], ShipYOLO [11] ou YOLOX [10], et cette dernière architecture semble maintenant supportée par l'outil de déploiement VITIS AI 2.0 [20].

B. Description du modèle YOLOv3

Cette architecture, représentée à la figure 3, est principalement composée:

- Du réseau de neurones à convolution (CNN) *Darknet-53* constitué de 53 couches de convolutions. Le composant principal de ces couches est le block résiduel (en mauve), introduit pour la première fois par [12]. Cette première partie du réseau agit comme un encodeur, extrayant les caractéristiques de l'image (*feature maps*) à différents niveaux de résolution.
- De couches de sur-échantillonage (en vert) qui agissent comme un décodeur en augmentant la résolution des *feature maps*. En sortie du sur-échantillonage, les informations générées sont mélangées (addition) avec certaines couches intermédiaires du CNN.

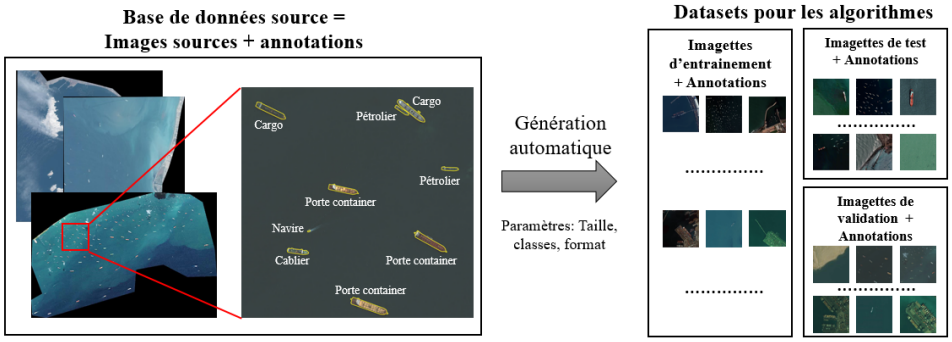


Fig. 2. Génération des datasets indépendants d'entraînement / validation / test pour l'apprentissage des réseaux de neurones et l'évaluation des performances. Imagery Products © 2021 Maxar Technologies.

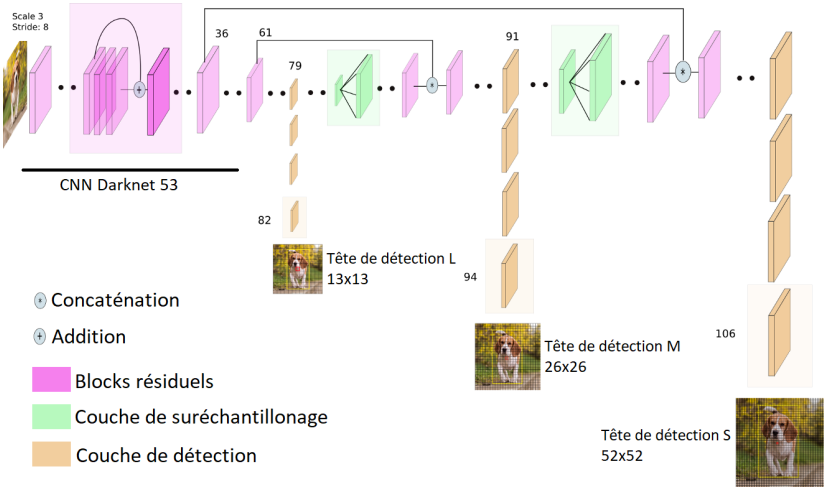


Fig. 3. Représentation de l'architecture de YOLOV3 comme présenté dans [7]. La dimension des images d'entrée est fixée à (416x416x3) ce qui influe sur le nombre de cellules des trois têtes de détection (13x13, 26x26 et 52x52 cellules)

- De trois têtes de détection (en jaune) se terminant par une grille de cellules. Chacune des ces cellules peut détecter et classifier jusqu'à trois objets. D'une grille à l'autre, le nombre de cellules et la dimension des objets détectés varient. Plus une grille de détection est située en fin du réseau, plus elle est générée à partir de couches haute-résolution, ce qui lui permet de détecter des plus petits objets.

Pour une image (3 canaux RGB) en entrée de dimension (W,H,3), la dimension des grilles en sortie de modèle est respectivement de, $(\frac{W}{32} \times \frac{H}{32})$, $(\frac{W}{16} \times \frac{H}{16})$ et $(\frac{W}{8} \times \frac{H}{8})$ cellules. Chacune de ces cellules produit 3 boites de prédictions (*bounding box*) représentées chacune par un vecteur BB de dimension $(5 + C)$ et défini par le vecteur (1) :

$$\text{BB} = (x, y, w, h, p_{obj}, p_0, .., p_{C-1}) \tag{1}$$

avec:

C = nombre de classes prédites par le réseau
xy = coordonnées du centre de la boite
wh = largeur et hauteur de la boite
p_{obj} = probabilité de présence d'un objet
p_i = probabilité d'appartenir à la classe i

Il est à noter que le réseau ne prédit pas directement les boites au bon format et qu'une mise en forme est nécessaire. Ce traitement, défini dans [18], consiste principalement en:

- la translation des coordonnées du centre de la boite en fonction de la position de la cellule sur la grille
- la modifications de l'*a priori* sur la dimension des boites (ancres, cf. section III-C2).

Une fois cette mise en forme appliquée, les objets de la classe i présents dans l'image sont détectés en filtrant les BB dont le score est supérieur à un seuil, l'*Object threshold*:

$$score = p_{obj} \times p_i \tag{2}$$

Cependant, un même objet est régulièrement détecté par plusieurs cellules voisines (cf. figure 4). Afin de ne conserver qu'une seule boite, un algorithme appliqué en post-traitement, le NMS (*Non Maxima Suppression*), va détecter les boites correspondant au même objet et les supprimer. Un seuil, le *NMS threshold*, permet de définir à partir de quel recouvrement l'algorithme considère que deux boîtes correspondent au même navire. Ce seuil est fixé à 50% pour nos simulations.

Fig. 4. Illustration de la détection d'un même objet par plusieurs cellules voisines. Après application de l'algorithme NMS, seule la boite en blanc est conservée. Imagery Products © 2018 Maxar Technologies.

C. Entraînement du réseau de détection

1) Fonction de coût: Comme défini par [17], afin de détecter les objets annotés $(\hat{x}, \hat{y}, \hat{w}, \hat{h}, \hat{p_{obj}}, \hat{p_0}, .., \hat{p_{C-1}})$, le réseau est entraîné à minimiser une fonction de coût (*loss function*).

Cette coût se divise en plusieurs composantes pondérées par différents coefficients λ:

- coût de localisation (λ_{coord}): elle est liée à l'erreur sur les coordonnées (x_i, y_i, w_i, h_i) des boites prédites
- coût de non-détection (λ_{obj}): elle est liée à l'erreur de non détection d'un objet
- coût de fausse détection (λ_{noobj}): elle est liée à l'erreur de prédire un objet inexistant
- coût de classification (λ_{class}): elle est liée à l'erreur sur la classification d'un objet

2) Choix des ancres: Les ancres représentent un a priori sur la dimension des bateaux à détecter. Au nombre de 9, elles sont associées chacune à une des 3 grilles et à une des 3 prédictions des cellules. Afin de correspondre au mieux à la distribution des bateaux du jeu de données, les dimensions (w,h) des navires sont divisées en 9 classes à l'aide de l'algorithme de clustering K-means. Les ancres sont ensuite déterminées en sélectionnant les dimensions moyennes de ces 9 classes.

Durant l'inférence, chacune des prédictions des cellules génèrent un couple (δ_w, δ_h) qui correspond à la modification à apporter à l'ancre (p_w, p_h) pour obtenir la largeur et hauteur de la prédiction selon l'équation (3):

$$w = p_w \times e^{\delta_w} \quad \text{et} \quad h = p_h \times e^{\delta_h} \tag{3}$$

3) Procédure d'entraînement: Pour entraîner ce modèle à notre cas d'utilisation, nous avons choisi une implémentation de YOLOv3 en Keras disponible sur [9]. Nous avons modifié son architecture afin qu'il ne détecte que la classe bateau, le nombre de classes *C* est donc égal à un. Le réseau a été entraîné avec 3 GPU Nvidia GTX 1080 Ti. Les hyperparamètres utilisés pour l'entraînement sont listés ci-dessous. Mis à part les ancres et le redimensionnement des images qui ont été optimisés, le choix des ces valeurs correspond à l'état de l'art.

- Max *epochs*: 300
- Taille du *batch*: 24 soit 8 par GPU
- Redimensionnement aléatoire des images entre 384x384 et 448x448
- Taux d'apprentissage initial: 1e-4
- Initialisation des poids: réseau Darknet-53 pré-entrainé sur ImageNet
- Paramètres de la fonction d'erreur:

$$\lambda_{coord} = \lambda_{noobj} = \lambda_{class} = 1, \ \lambda_{obj} = 5 \tag{4}$$

- Ancres: [5,7,7,13,12,17,14,9,16,29,28,16,41,38,87,57,111,136]

D. Performances

Avant de présenter les performances du modèle, nous définissons différents concepts utiles pour la suite :

- IoU: de l'anglais *Intersection over Union*, ratio entre l'intersection et l'union des surfaces des boites prédite et attendue. L'IoU vaut 1 lorsque la prédiction est parfaite et 0 lorsque les boites n'intersectent pas.
- VP, vrai positif: prédiction correcte. Comme présenté à la section III-B, une prédiction est correcte si son score (2) est supérieur à l'*Object threshold* (fixé à 30%) et que l'IoU avec la vérité terrain est supérieure à un seuil (fixé à 50%).
- FP, faux positif: détection ne correspondant pas à un bateau, ou relative à un bateau déjà détecté.
- FN, faux négatif: un navire non détecté (score ou IoU avec la vérité terrain insuffisant).

À la deuxième colonne de la figure 6, plusieurs résultats d'inférence sur le jeu de test sont représentés. Le modèle est capable de détecter la majorité des vérités terrain, et ce, pour différents états de mer et morphologies de navires. Les performances du réseau pour les 3 jeux de données sont reprises dans le tableau IV et sont définies ainsi :

- Précision: proportion de prédictions correctes par rapport à l'ensemble des prédictions.
- Rappel: proportion de prédictions correctes par rapport à l'ensemble des navires présents dans les images.
- F1-Score: la moyenne harmonique de la précision et du rappel.
- AP50: une métrique classique qui correspond à l'aire sous la courbe précision-rappel pour un IoU$\geq 50\%$.

TABLE IV
PERFORMANCES DU MODÈLE EN FONCTION DU JEU DE DONNÉES

dataset	Précision	Rappel	F1-Score	AP50
Train	0,405	0,779	0,533	0,654
Valid	0,395	0,756	0,519	0,618
Test	0,416	0,784	0,543	0,676

Nous constatons que le réseau se comporte légèrement mieux sur le jeu de test que le jeu d'entraînement. Bien que le modèle soit censé se comporter mieux sur le jeu d'entraînement, l'évolution des performances lors de l'entraînement est un processus bruité et il est possible que

les performances sur le jeu de test dépassent celles du jeu d'entraînement.

Nous observons également que la précision du modèle est largement inférieure au rappel. Cela signifie que l'algorithme a tendance à générer plus de faux positifs que de faux négatifs. En analysant les images de la figure 6, la majorité des faux positifs proviennent de cellules voisines détectant plusieurs fois le même navire et non de zones de l'image sans navires. Il est cependant possible, en modifiant le *NMS threshold*, d'améliorer les précision du modèle au détriment du rappel et de l'AP50 (cf. tableau V). Ce compromis entre précision et rappel étant simplement un paramètre du post-traitement, il peut être modifié au cours de la mission et ne nécessite pas de ré-entraînement du modèle. Dans notre cas, nous avons choisi arbitrairement d'optimiser la métrique AP50, et avons donc conservé une valeur de 50% pour le *NMS threshold*.

TABLE V
MÉTRIQUES SUR LE JEU DE TEST EN FONCTION DU *NMS threshold*

NMS Thr.	Précision	Rappel	F1-Score	AP50
10%	0,600	0,674	0,635	0,603
30%	0,520	0,724	0,605	0,640
50%	0,416	0,784	0,543	0,676
70%	0,275	0,824	0,412	0,661

Finalement, la distribution, pour le jeu de test, des vrais positifs, des faux négatifs, des faux positifs ainsi que du nombre total de navires annotés en fonction de la dimension (en pixel) du navire est résumée par la figure 5. Un bon algorithme minimise, pour toutes les dimensions de navires, les FP (courbe rouge) et les FN (courbe jaune) et le nombre de VP (courbe verte) tend vers le nombre total de bateaux (courbe bleue). Dans cette figure, nous constatons que le modèle se comporte mieux pour les grands navires que les petits. En effet, la proportion de faux positifs et de faux négatifs est plus importante pour les petits navires ($\leq$ 15 pixels). Au-delà d'une dimension de 20 pixels, le nombre de bateaux non-identifiés chute en dessous de 10% et la proportion de faux positifs est inférieure aux vrais positifs. En effet, les performances du modèle s'améliorent avec la résolution et donc avec la taille des navires.

IV. DÉPLOIEMENT SUR ÉLECTRONIQUE EMBARQUÉE

A. Présentation des différentes cibles

Pour le déploiement de l'algorithme, nous avons sélectionné trois cibles différentes dans le but de se rapprocher des différentes contraintes opérationnelles (consommation et capacités de mémoire et de calcul réduites) et de couvrir différentes solutions hardware (FPGA et GPU). Les caractéristiques de ces cibles sont reprises dans le tableau VI.

1) NVIDIA Jetson Nano: destinée au portage d'applications d'IA embarquées à faible coûts et à consommation réduite, elle permet de valider un portage sur GPU avec peu de ressources.

2) NVIDIA Jetson NX: de la même famille que la Jetson Nano, elle offre des performances supérieures à faible consommation. Un critère dans le choix de cette cible est qu'il existe

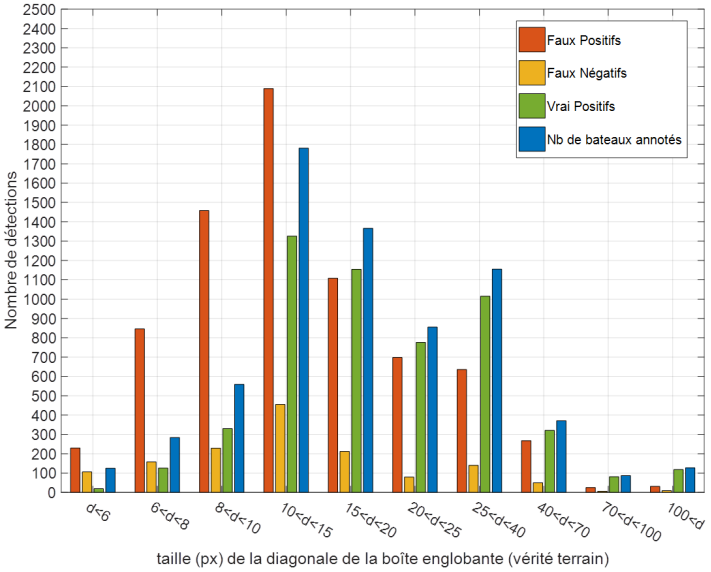


Fig. 5. Répartition du nombre de FP, FN, VP et de bateaux sur le jeu de test en fonction de la diagonale de la boite englobante

une version [1] compatible avec des applications embarquées (drone, avion, etc.) mais non spatiales. Une autre cible de la même famille, la Jetson TM TX2i, a également été certifiée pour l'environnement spatial [2].

3) Zynq UltraScale+ MPSoC ZCU104: cette carte d'évaluation est composée d'un MPSOC (*Multi-processor System on Chip*) ZU7EV. La partie logique programmable est pré-configurée avec une architecture matérielle optimisée pour l'exécution de réseaux de neurones, fournie par le constructeur Xilinx. Elle inclut des accélérateurs matériels (DPU : *Deep Learning Processor Unit*) permettant d'accélérer l'inférence des principales couches de réseaux de neurones à convolutions.

TABLE VI
CARACTÉRISTIQUES DES CIBLES

	NVIDIA Jetson Nano	NVIDIA Jetson NX	Zynq UltraScale+ MPSoc ZCU104
Perfo.	472 GFLOPs	21 TOPs	n.a.
Alim.	5V	entre [10V,20V]	12V
Conso.	$\leq 10W$	$\leq 20W$	$\leq 20W$
Microproc.	ARM Cortex-A57 4 coeurs à 1,43 GHZ	ARM NVIDIA Carmel 64bit 6 cœurs	Quadcore Arm Cortex-A53, Dualcore Arm Cortex-R5
GPU	NVIDIA Maxwell 128 coeurs CUDA	NVIDIA Volta 384 cœurs	Mali-400 MP2 GPU
Accél.	n.a.	NVIDIA Deep Learning Accelerator	Logique Prog. 504K cellules logiques, 1728 DSP
Mémoire RAM	4 GB LPDDR4	8GB LPDDR4	2GB DDR4
Quantif.	FP16	FP16/INT8	INT8 (VITIS AI Quantizer)
Autre	Codec vidéo H.264/H.265	Codec vidéo H.264/H.265	Codec vidéo H.264/H.265

B. Procédure de portage

Le déploiement d'un algorithme nécessite tout d'abord de convertir notre modèle tf.keras en un format ne contenant plus de poids entraînables. Cette opération, dénommée *"graph freezing"*, est disponible dans la librairie Tensorflow et nous permet d'obtenir un modèle au format *pb*. Une fois cette opération effectuée, le portage sur une cible est dépendant des outils proposés par le constructeur.

1) Déploiement sur les Jetsons: Le portage d'un réseau chez le constructeur NVIDIA consiste à convertir le modèle *pb* vers le format ONNX. Le passage vers le format ONNX fige les dimensions de l'image d'entrée et le nombre de cellules dans les grilles en sortie. Ce modèle est ensuite optimisé et compilé en flottant 16bits à l'aide de la librairie TensorRT 8.0.1, appelée à travers son API en Python. Pour la Jetson NX, il est également possible de compiler le modèle en entier 8bits (INT8). La conversion des valeurs flottantes en entiers est réalisée par TensorRT et nécessite une calibration à l'aide des images d'entraînement. Une fois le portage effectué, le modèle s'exécute sur GPU à travers un code Python. Il est à noter que les fonctions de post-traitement et le NMS présentés à la section III-B sont exécutés par ce script sur le CPU.

Un modèle plus complet, intégrant le post-traitement ainsi que le NMS dans le GPU, a été développé. Une fois déployé, ce dernier présente de meilleures performances en temps d'inférence, étant donné la parallélisation du post-traitement de chaque cellule. Ces résultats sont présentés dans la section IV-C.

2) Déploiement sur la ZCU104: Le portage sur la ZCU104 nécessite la librairie VITIS-AI 1.3 de chez Xilinx qui compile le modèle en un jeu d'instructions dédié permettant d'exécuter efficacement de nombreuses architectures de réseaux de neurones à partir de modèles entraînés avec les librairies TensorFlow, PyTorch et Caffe. Dans un premier temps, le déploiement consiste à quantifier en entier 8bits le modèle *pb* à l'aide d'une calibration sur les images d'entraînements. Une fois quantifié, le modèle est compilé pour la cible. La compilation du modèle fige les dimensions de l'image d'entrée et le nombre de cellules dans les grilles en sortie. Une fois le portage effectué, le modèle s'exécute à l'aide d'un code en C++. Les fonctions de post-traitement et le NMS présentés à la section III-B sont exécutés par ce code sur le CPU.

C. Performances

Différentes prédictions, en fonction du portage, sont reprises dans la figure 6 et les performances dans le tableau VII. Du côté des Jetsons, nous constatons que le portage en flottant ne dégrade pas les performances. La calibration en INT8 est optimisée par NVIDIA et ne dégrade l'AP50 que de 2%. Côté ZCU104, la calibration dégrade également de 2% l'AP50.

Nous avons également estimé le temps d'inférence moyen par imagette en fonction de la cible, de la taille du batch (nombre d'images traitées en parallèle) et en faisant également varier la taille des images en entrée (416x416, 640x640 et 1024x1024 pixels). Ces résultats, repris dans le tableau VIII, nous permettent de tirer plusieurs conclusions.

TABLE VII
COMPARATIF DES PERFORMANCES DU MODÈLE SUR LE DATASET DE TEST AVANT ET APRÈS PORTAGE

Cible	Quant.	Précision	Rappel	F1-Score	AP50
modèle tf.keras	FP64	0,416	0,784	0,543	0,676
Jetson NX	FP16	0,415	0,784	0,543	0,675
Jetson NX	INT8	0,386	0,779	0,516	0,654
Jetson Nano	FP16	0,416	0,785	0,544	0,676
ZCU104	INT8	0,388	0,781	0,518	0,659

Pour les Jetsons:

- Le débit n'évolue pas significativement avec la taille du batch et des images. Une justification probable serait que les traitements utilisent la totalité des ressources du GPU et qu'il n'est pas possible de paralléliser d'avantage.
- La majorité du temps de calcul provient du post-traitement (mise en forme des sorties + NMS, cf. section III-B) qui est effectué de manière séquentielle par le CPU. La version du code (*"NX+NMS"*) incluant ce post-traitement dans le GPU permet d'obtenir des performances en temps similaire à l'inférence seule.
- La quantification en entier 8bits ne réduit pas significativement le temps d'inférence
- Les performances en temps de calcul de la Jetson NX sont environ 8 fois supérieures à celle de la Jetson Nano.

Pour la ZCU104:

- Le temps d'inférence est constant quelles que soient les dimensions de l'image: il semblerait que la ZCU n'est pas utilisée à sa capacité maximale et que le modèle peut se redimensionner et être parallélisé en fonction des dimensions de l'image en entrée. Nous constatons de très bon débit pour de grandes taille d'images.
- Similairement aux autres cibles, le post-traitement nécessite un temps de calcul important réduisant le débit total.

Finalement, dans le tableau IX, nous avons également repris l'occupation mémoire ainsi que la consommation du modèle après portage sur différentes cibles. Il est à noter que sur les Jetsons, la mémoire RAM est partagée entre le CPU et le GPU. Son occupation est dès lors directement proportionnelle à la taille du modèle. Pour la ZCU, le modèle est déployé dans le FPGA et la RAM allouée n'est pas un bon indicateur de l'occupation du modèle. La puissance consommée par la carte n'a pas pu être obtenue pour la ZCU104.

Nous constatons que:

- Sur les différentes cibles, la mémoire allouée évolue avec la taille des images et du batch. Cependant, pour les Jetsons, il n'est pas possible de connaître la répartition entre inférence/post-traitement de cette consommation mémoire.
- Pour la Jetson NX, la quantification en entiers permet de réduire drastiquement la consommation de puissance

TABLE VIII
COMPARATIF DU TEMPS D'EXÉCUTION DU MODÈLE EN FONCTION DES PARAMÈTRES DE PORTAGE

			Inférence		Infér. + Post-Trait.	
Cible	taille imag.	Batch Size	T par image	Débit Mpx/s	T par image	Débit Mpx/s
Jetson NX FP16	416	BS1	24ms	7,13	**178ms**	0,97
		BS4	20ms	8,48	172ms	1,01
		BS8	20ms	8,74	168ms	1,03
	640	BS4	46ms	8,88	314ms	1,31
	1024		118ms	8,86	1073ms	0,98
+INT8	416	BS1	22ms	7,87	176ms	0,98
+NMS			n.a.	n.a.	**27ms**	**6,34**
Jetson Nano	416	BS1	188ms	0,92	400ms	0,43
		BS4	195ms	0,89	396ms	0,44
		BS8	190ms	0,91	392ms	0,44
	640	BS4	414ms	0,99	806ms	0,51
	1024		929ms	1,13	2302ms	0,46
ZCU104	416	BS1	83ms	2,08	96ms	1,81
	640		83ms	4,92	206ms	1,99
	1024		83ms	12,62	537ms	1,95

TABLE IX
COMPARATIF DE LA CONSOMMATION DES CIBLES EN FONCTION DES PARAMÈTRES DE PORTAGE

Cible	taille imag.	Batch Size	Conso. RAM Repos (GB)	Δ Conso. RAM (GB)	Conso. Puiss. Repos (W)	Δ Conso. Puiss. (W)
Jetson NX FP16	416	BS1	∼3.1	0,91	∼4.50	1,9
		BS4		0,90		2,4
		BS8		0,99		3,4
	640	BS4		1,39		4,3
	1024			1,56		5,0
+INT8	416	BS1		0,90		0,4
+NMS				0,90		4,2
Jetson Nano	416	BS1	∼1.48	0,90	∼2.7	1,2
		BS4		0,99		1,1
		BS8		1,12		1,5
	640	BS4		0,95		1,2
	1024			1,64		1,4
ZCU 104	416	BS1	∼0,37	0,41	n.a.	n.a.
	640			0,53		n.a.
	1024			0,71		n.a.

V. CONCLUSION ET PERSPECTIVES

Cet article parcourt les différentes étapes du développement d'un détecteur optique de navires. À notre connaissance, ce détecteur à base de réseau neuronal semble être le premier embarquable dans un satellite, ce que nous avons démontré en le déployant sur différentes cibles électroniques (FPGA et GPU) faibles consommations et similaires à des cibles embarquables sur satellite. Dans un premier temps, comme pour tout algorithme d'apprentissage supervisé, la performance et la robustesse du modèle reposent en grande partie sur la qualité, la représentativité et la quantité d'échantillons de la base de données. Pour cette raison, nous avons confié à notre partenaire GEO4I la construction d'une base de données de grande qualité, multiclasses et de haute résolution. Ensuite, le modèle de détection a été sélectionné en tenant compte des contraintes liées au milieu spatial et à l'embarqué. C'est l'algorithme YOLOv3 qui a été retenu, offrant un bon ratio performances/consommation de ressources et étant compatible avec les différentes chaînes de déploiement. En fonction des besoins opérationnels, le compromis entre le taux de fausses alarmes et de non détections peut également être ajusté à en vol, et ce, sans ré-entraînement. Finalement, l'évolution des performances de l'algorithme ainsi que la consommation en ressources hardware ont été comparées après portage sur cibles. Nous avons pu constater que la quantification du modèle peut entraîner de légères pertes. Au niveau du débit de l'algorithme, le post-traitement non parallélisé, et plus particulièrement le NMS, contribue à la majeure partie du temps de traitement. Une implémentation parallélisée sur GPU est proposée, ce qui améliore très significativement l'inférence.

Ces travaux ont déjà permis de mettre en place une démonstration "sur table" : une webcam connectée à nos cartes électroniques embarquant l'algorithme décrit dans cet article permet la détection de navires dans une image satellite imprimée sur un large poster.

En guise de perspective, des études sont actuellement menées afin d'élargir la détection de navires à la tâche de reconnaissance et d'identification. À l'aide de notre base de données multiclasses, nous entraînons notre modèle à classifier ces navires par classe (civil, militaire, autres) et sous-classe (porte-avions, conteneur, pétrolier, etc.). Suite à la mise à jour des chaînes de déploiement, nous envisageons également de porter de nouvelles architectures plus performantes, telles que YOLOX, sur les cibles présentées et sur d'autres, les *TPU* (*Tensor Processing Unit*), spécifiquement développées pour accélérer les réseaux de neurones.

DÉCLARATION DE CONFLIT D'INTÉRÊT

Les auteurs déclarent n'avoir aucun conflit d'intérêt.

REMERCIEMENTS

Ces travaux ont été menés dans le cadre du projet CIAR (*Chaîne Image Autonome et Réactive*) de l'Institut de Recherche Technologique Saint-Exupéry. Les auteurs remercient les partenaires industriels et académiques du projet : Thales Alenia Space, Activeeon, Avisto, ELSYS Design, MyDataModels, Geo4i, Inria et le LEAT/CNRS.

REFERENCES

[1] Fiche technique de a179 - lightning. https://www.recabuk.com/datasheets/systems/aitech/A179.pdf. Accessed: 2022-05-05.

[2] Fiche technique de s-a1760 - venus. https://www.emcomo.de/fileadmin/user_upload/EMCOMO/PDF/S-A1760-Preliminary-Datasheet.pdf. Accessed: 2022-05-05.

[3] European Commission's factsheet about illegal fishing, jun 2021. [Online; accessed 13. May 2022].

[4] Illegal fishing and human rights abuse in China's distant water fleet, May 2022. https://ejfoundation.org/news-media/global-impact-of-illegal-fishing-and-human-rights-abuse-in-chinas-vast-distant-water-fleet-revealed-2[Online; accessed 13. May 2022].

[5] Filippo Maria Bianchi, Martine M. Espeseth, and Njål Borch. Large-Scale Detection and Categorization of Oil Spills from SAR Images with Deep Learning. Remote Sens., 12(14):2260, July 2020.

[6] Jan Martin Brockmann, Till Schubert, and Wolf-Dieter Schuh. An Improved Model of the Earth's Static Gravity Field Solely Derived from Reprocessed GOCE Data. Surv. Geophys., 42(2):277–316, March 2021.

[7] Yuan Dai, Weiming Liu, Haiyu Li, and Lan Liu. Efficient foreign object detection between psds and metro doors via deep neural networks. IEEE Access, 8:46723–46734, 2020.

[8] Ryan Engstrom, Jonathan Samuel Hersh, and David Locke Newhouse. Poverty from Space: Using High-Resolution Satellite Imagery for Estimating Economic Well-Being, December 2017.

[9] experiencor/keras yolo3. Dépot de l'implémentation de yolo-v3 sous keras. https://github.com/experiencor/keras-yolo3, 2020.

[10] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: exceeding YOLO series in 2021. CoRR, abs/2107.08430, 2021.

[11] Xu Han, Lining Zhao, Yue Ning, and Jingfeng Hu. ShipYOLO: An Enhanced Model for Ship Detection. J. Adv. Transp., 2021:1060182, June 2021.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[13] R. Hinz, J. I. Bravo, M. Kerr, C. Marcos, A. Latorre, and F. Membibre. EO ALERT: Machine Learning Based On Board Satellite Processing for Very Low Latency Convective Storm Nowcasting. Zenodo, October 2020.

[14] Bo Li, Xiaoyang Xie, Xingxing Wei, and Wenting Tang. Ship detection and classification from optical remote sensing images: A survey. Chin. J. Aeronaut., 34(3):145–163, Mar 2021.

[15] Hao Li, Lianbing Deng, Cheng Yang, Jianbo Liu, and Zhaoquan Gu. Enhanced YOLO v3 Tiny Network for Real-Time Ship Detection From Visual Image. IEEE Access, 9:16692–16706, January 2021.

[16] Ana María Pacheco-Pascagaza, Yaqing Gou, Valentin Louis, John F. Roberts, Pedro Rodríguez-Veiga, Polyanna da Conceição Bispo, Fernando D. B. Espírito-Santo, Ciaran Robb, Caroline Upton, Gustavo Galindo, Edersson Cabrera, Indira Paola Pachón Cendales, Miguel Angel Castillo Santiago, Oswaldo Carrillo Negrete, Carmen Meneses, Marco Iñiguez, and Heiko Balzter. Near Real-Time Change Detection System Using Sentinel-2 and Machine Learning: A Test for Mexican and Colombian Forests. Remote Sens., 14(3):707, February 2022.

[17] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. CoRR, abs/1506.02640, 2015.

[18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. CoRR, abs/1804.02767, 2018.

[19] Sandeep Kumar Singh, Shradha Fowdur, Jakob Gawlikowski, and Daniel Medina. Leveraging Evidential Deep Learning Uncertainties with Graph-based Clustering to Detect Anomalies. ResearchGate, July 2021.

[20] Xilinx. Vitis-AI 2.0, May 2022. https://github.com/Xilinx/Vitis-AI/tree/master/models/AI-Model-Zoo/model-list/pt_yolox_TT100K_640_640_73G_2.0[Online; accessed 13. May 2022].

[21] Tianwen Zhang, Xiaoling Zhang, Jianwei Li, Xiaowo Xu, Baoyou Wang, Xu Zhan, Yanqin Xu, Xiao Ke, Tianjiao Zeng, Hao Su, Israr Ahmad, Dece Pan, Chang Liu, Yue Zhou, Jun Shi, and Shunjun Wei. SAR Ship Detection Dataset (SSDD): Official Release and Comprehensive Data Analysis. Remote Sens., 13(18):3690, September 2021.

[22] SuYu Zhou and Jun Yin. YOLO-Ship: A Visible Light Ship Detection Method. In 2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE), pages 113–118. IEEE, January 2022.

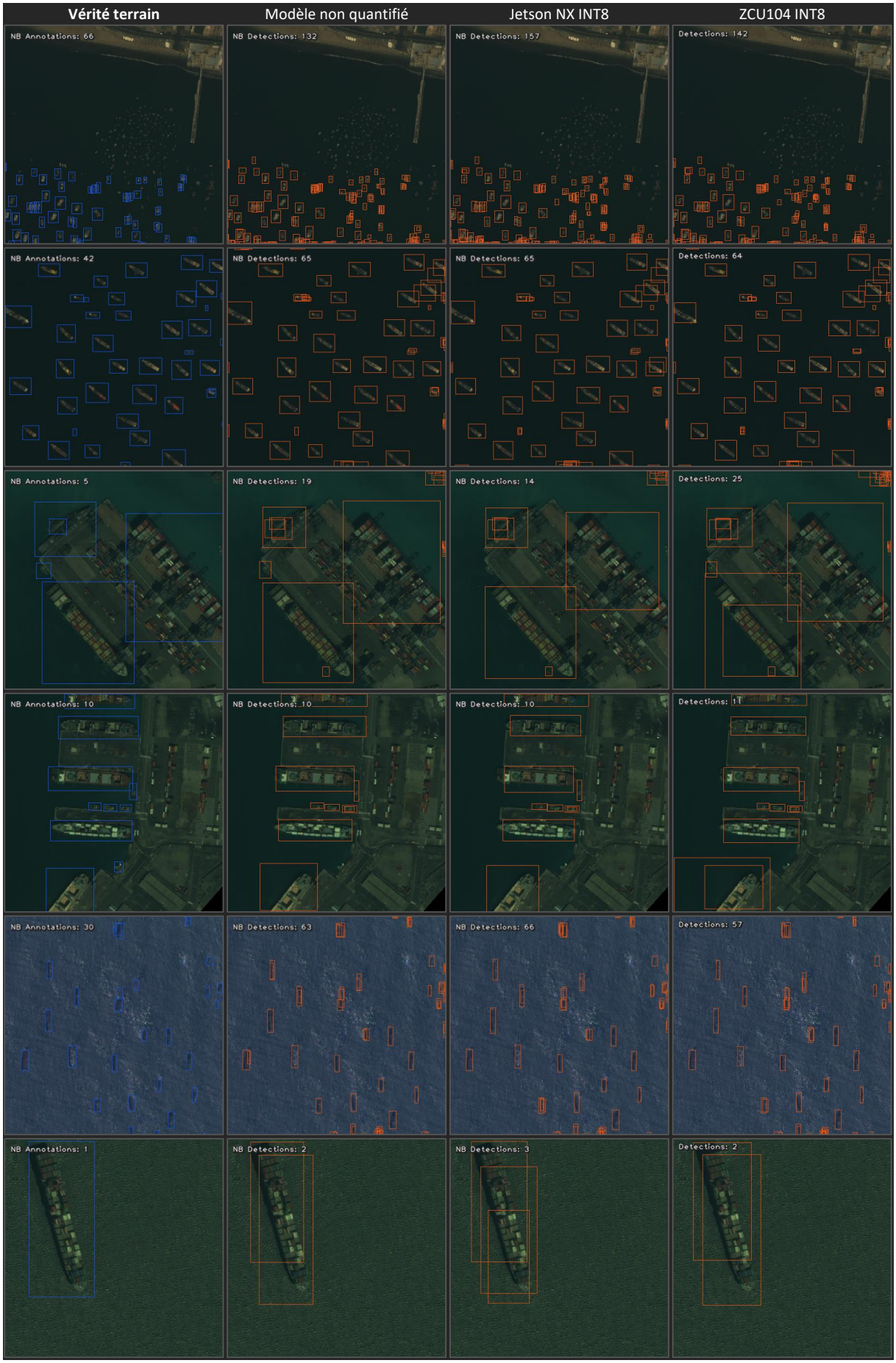


Fig. 6. Exemples de prédictions du modèle *YOLO-v3* sur six imagettes appartenant au jeu de test. Images de gauche à droite : vérité terrain, prédiction du modèle non quantifié (en flottant 16 ou 64bits), prédiction quantifiée sur la Jetson NX en entiers 8bits, prédiction sur la ZCU en entiers 8bits. La réalité terrain est représentée en bleu et les prédictions en rouge. Imagery Products © 2018 Maxar Technologies.

Projectile trajectory estimation: an LSTM approach

Alicia ROUX
Guidance, Navigation and Control (GNC) department
French-German Research Institute of Saint-Louis
Saint-Louis, France
alicia.roux@isl.eu

Sébastien CHANGEY
Guidance, Navigation and Control (GNC) department
French-German Research Institute of Saint-Louis
Saint-Louis, France
sebastien.changey@isl.eu

Jonathan WEBER
Université de Haute-Alsace, IRIMAS (UR 7499)
Mulhouse, France
jonathan.weber@uha.fr

Jean-Philippe LAUFFENBURGER
Université de Haute-Alsace, IRIMAS (UR 7499)
Mulhouse, France
jean-philippe.lauffenburger@uha.fr

***Abstract*—This paper presents a deep learning approach to estimate a generic mortar trajectory in a GNSS-denied environment. For this purpose, Long-Short-Term-Memories (LSTMs) are trained on projectile fire simulations. Network input data are the embedded IMU (Inertial Measurement Unit), the reference magnetic field, flight parameters specific to the considered ammunition (initial velocity, fin angle, barrel elevation) and a time vector. This paper focuses on the influence of input data normalization and navigation frame rotation during the training step, leading to rescaling a 3D-value over similar variation ranges with no information loss. LSTM estimates are compared to a classical Dead Reckoning navigation algorithm. Results clearly show the AI contribution, especially for projectile position and velocity estimation.**

***Index Terms*—Projectile navigation, Inertial Measurement Unit, Artificial intelligence, Long-Short-Term-Memory**

I. Introduction

Projectile trajectory estimation is a complex task due to dynamic constraints imposed on the system and low-cost sensors used. For this purpose, projectile navigation is based on embedded IMU (Inertial Measurement Unit) and GNSS (Global Navigation Satellite System) signals. The IMU and GNSS measurements are combined with navigation algorithms such as Kalman Filters [1] to estimate a trajectory. Due to GNSS signals vulnerability (hostile conditions, disturbed or unavailable signals) [2], users aims to exclude these measurements for trajectory estimation [3]–[6].

Moreover, new methods based on AI (Artificial intelligence) are increasingly used for defense applications such as surveillance, reconnaissance, tracking or navigation [7], [8]. Therefore, this paper presents an AI-based algorithm to estimate a projectile trajectory in a GNSS-denied environment using only the embedded IMU and pre-flight parameters specific to the ammunition considered.

Considering that a trajectory is a time serie, AI can provide interesting approaches for its estimation. Time series prediction could be based on Recurrent Neural Networks (RNN) [9]–[11]. RNNs are particularly well suited for time series prediction as they memorize past data to predict future data. However, the simplest form of RNNs, the Vanilla RNN, exhibits vanishing/exploding gradient problems during the training step, so another form of RNNs can be considered: the Long Short-Term Memory (LSTM) [9]–[11]. A LSTM is an extension of the Vanilla RNN including a memory cell in addition to the hidden states, in order to capture both long-term and short-term time dependencies [11].

This paper presents an AI-based solution for projectile navigation in a GNSS-denied environment. LSTMs are trained to estimate projectile position, velocity and Euler angles.
In summary, the main contributions of this work are:

- to detail an LSTM-based approach to estimate a mortar trajectory in the local navigation frame, from IMU measurements, the reference magnetic field, flight parameters (initial velocity, fin angle, barrel elevation) and a time vector.
- to present BALCO (BALlistic COde) [12] used to generate the dataset. This simulator provides true-to-life trajectories of several projectile types according to specific flight parameters. Note that this work focuses only on generic mortar trajectories.
- to investigate different normalization form of the LSTM input data in order to evaluate their contribution on the estimation accuracy. For this purpose, several LSTMs are trained with different input data normalization.
- to study the impact of the local navigation frame rotation on the estimation accuracy. Rotating the local navigation frame during the training step allows a quantity to have similar variation ranges along the three axes.
- to evaluate LSTM estimation accuracy compared to a classical Dead Reckoning navigation algorithm [1], performed on the whole test dataset.

The outline of the paper is as follows. A first part *(II)* presents a brief introduction to projectile navigation and LSTM operating principle. A second part *(III)* focuses on the projectile trajectory dataset. The third part *(IV)* details LSTMs trained to estimate a projectile trajectory and finally, the last part *(V)* presents estimation results.

II. Related work

This section introduces conventional algorithms used for projectile navigation, AI-based navigation approaches, and a brief overview of the LSTM principle.

A. Model-based projectile trajectory estimation

Projectile navigation exploits GNSS (Global navigation satellite system) and IMU (Inertial Measurement Units) measurements, i.e. accelerometers, gyrometers or magnetometers embedded in the projectile. These data are then fused with Kalman Filters [1] such as the Adaptive Extended Kalman Filter [13], the Invariant Extended Kalman Filter [3], the Unscented Kalman Filter [14] or the mixed Extended Unscented Filter [15]. These filters are based on a Dead Reckoning algorithm and then corrected by observations. A Dead Reckoning algorithm [1] aims to integrate gyrometer ω and accelerometer a readings in the sensor frame $\boldsymbol{s}$ to estimate at each discrete time k:

$$R_k = R_{k-1}[\omega_k \Delta_t]_\times \tag{1}$$

$$v_k = v_{k-1} + (R_{k-1}a_k + g)\,\Delta_t \tag{2}$$

$$p_k = p_{k-1} + v_{k-1}\Delta_t + \tfrac{1}{2}(R_{k-1}a_k + g)\,\Delta_t^2 \tag{3}$$

with $R_k \in SO(3)$ the rotation matrix from the sensor frame $\boldsymbol{s}$ to the local navigation frame $\boldsymbol{n}$ determined by the projectile Euler angles, $g \in \mathbb{R}^3$ the constant gravity vector, $p_k \in \mathbb{R}^3$ and $v_k \in \mathbb{R}^3$ respectively the projectile position and velocity, and $[.]_\times$ an $SO(3)$ operator defined as:

$$\forall x \in \mathbb{R}^3, \quad [x]_\times = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{4}$$

Moreover, for high-speed spinning projectiles such as a 155mm shell, the embedded sensors have to resist to high rotation rates and accelerations and some standard sensors such as gyrometers saturate and become useless as explained in [16]. Therefore, several algorithms specialized in high-speed spinning projectile attitude estimation exploit the magnetometer measurements as in [4]–[6].

B. AI-based trajectory estimation

AI methods are increasingly used in the military field such as surveillance and target recognition, military training or cybersecurity [8], [17], [18]. Nevertheless, AI-based projectile trajectory estimation is not commonly used, despite recurrent networks (RNNs) are perfectly adapted to such applications.

Recurrent Neural Networks: Recurrent networks are commonly used for time series prediction as estimations are computed from past characteristics memorized in feedback loops [9]–[11]. The simplest form of RNNs, the Vanilla RNN, has no memory cell and is therefore inadequate to model long-term time dependencies. In addition, this network exhibits vanishing/exploding gradient problems during the training step [9], [11]. To overcome this issue, memory cells have been added to the Vanilla RNN, forming the Long Short-Term Memory (LSTM).

LSTM unit: A LSTM is composed by several units to deal with short and long-term memory. Figure 1 presents a LSTM unit with x_t the input at timestamp t, h_{t-1} the hidden state (previous LSTM unit output) and c_t the memory state, which aims to memorize long-term dependencies.

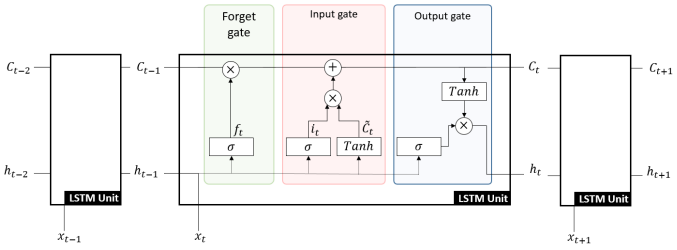


Fig. 1. LSTM cell operating principle composed by three gates with x_t the input at the current time, h_{t-1} the hidden state at the previous time, and c_t the memory cell state.

As shown in Figure 1, a LSTM unit is composed by three gates:
– the *forget gate* filters, through a Sigmoid function σ, data contained in the concatenation of x_t and h_{t-1}. Data are forgotten for values close to 0 and are memorized for values close to 1. The *forget gate* model is:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{5}$$

– the *input gate* extracts relevant information from $[h_{t-1}, x_t]$ by applying a Sigmoid σ and a Tanh function. The *input gate* is represented by:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_f) \tag{6}$$

$$\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{7}$$

The memory cell C_t is updated from the *forget gate* f_t and the *input gate* i_t, $\tilde{C}_t$, to memorize pertinent data:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{8}$$

– the *output gate* defines the next hidden state h_t containing information about previous inputs. The hidden state h_t is updated with the memory cell C_t normalized by a Tanh function and $[h_{t-1}, x_t]$ normalized by a Sigmoid function:

$$h_t = \sigma(W_h.[h_{t-1}, x_t] + b_h) \times tanh(C_t) \tag{9}$$

with $W_{(.)}$ and $b_{(.)}$, the different gate weights and biases.

Currently, only a few works exploit recurrent networks in the military field. There are commonly used for navigation, such as aircraft flight path prediction [19], [20], vehicle trajectory estimation [21], maritime route prediction [22], human motion prediction [23], [24] or target tracking [25]. It is however interesting to mention [26] focusing on projectile trajectory estimation based on LSTMs trained from incomplete and noisy radar measurements.

III. Projectile trajectory dataset

Results presented in this paper exploit a projectile fire dataset generated by BALCO (BALlistic COde) [12], i.e. a high fidelity projectile trajectory simulator.

As shown in Figure 2, two reference frames are considered:
– the *local navigation frame* ***n*** (black frame in Figure 2) in which projectile trajectories are expressed, is tangent to the Earth and assumed fixed during the projectile flight.
– the *sensor frame* ***s*** (green frame in Figure 2), rigidly fixed to the projectile and misaligned with the projectile gravity center, the frame where the inertial measurements are performed.

The dataset used in this work includes 5 000 mortar fire simulations and each simulation includes:

- *IMU measurements in the sensor frame* ***s****:* gyrometer $\omega \in \mathbb{R}^3$, accelerometer $a \in \mathbb{R}^3$ and magnetometer $h \in \mathbb{R}^3$ readings. Two kinds of inertial measurements are available:
  – *IMU measurements* performed in the sensor frame including a sensor error model: a misalignment model between each sensor axis and the projectile gravity center, a sensitivity factor, a bias and a noise (assumed Gaussian with zero mean) relative to each sensor axis.
  – *IMU DYN measurements*, performed in the sensor frame and issued from *IMU measurements* to which a transfer function is added to each sensor. This sensor model allows to denotes the response of the three sensors over the operating range.
- *reference magnetic field* $h_n \in \mathbb{R}^3$*:* in the local navigation frame ***n***, assumed constant during the projectile flight.
- *flight parameters:* the fin angle $\delta_f \in [0, 3]°$, the initial velocity at the end of the propulsion phase $v_0 \in [220, 320]$ m/s, and the barrel elevation angle $\alpha \in [40, 60]°$.
- *time vector* $k\Delta_t$ with $\Delta_t = 1e^{-3}s$ the IMU sampling period.
- *reference trajectory:* projectile position p, velocity v and Euler angles Ψ in the local navigation frame ***n*** at the IMU frequency.

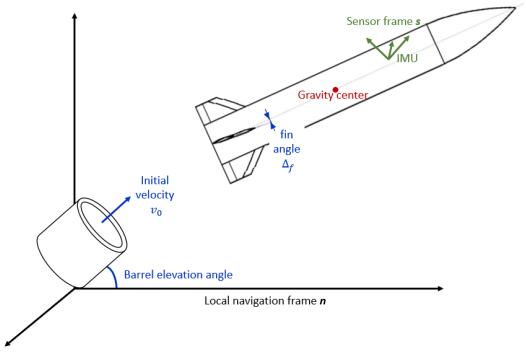


Fig. 2. Navigation frames (black - Local navigation frame ***n***, red - projectile gravity center, green - sensor frame ***s***) and flight parameters (fin angle δ_f, initial velocity v_0, barrel elevation angle α).

IV. Problem formulation

This part introduces LSTMs trained to estimate projectile trajectories from the dataset presented in section *(III)*. Four network types derived in eight versions are trained. Moreover, this part presents normalization methods, the local navigation frame rotation and network training characteristics.

A. Overview

As mentioned in the introduction *(I)*, the LSTM goal is to estimate a projectile trajectory from pre-flight data and the embedded IMU. As shown in Figure 3, LSTM predictions at time t are obtained from an input sequence of length τ and where the 16 features are:

- inertial measurements $\mathcal{M} \in \mathbb{R}^{12}$: *IMU measurements*, i.e. accelerometer a, gyrometer ω and magnetometer h readings in the sensor frame ***s*** and the reference magnetic field h_n in the local navigation frame ***n***,
- flight parameters $\mathcal{P} = (\delta_f, v_0, \alpha) \in \mathbb{R}^3$,
- the time vector $\mathcal{T} \in \mathbb{R}^1$.

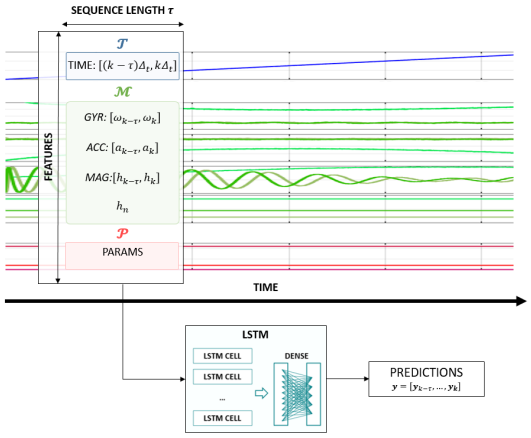


Fig. 3. LSTM input data: inertial measurements $\mathcal{M}$, flight parameters $\mathcal{P}$, time vector $\mathcal{T}$.

Four LSTMs types are trained and differ depending on the output features learned. $LSTM_{ALL}$ trained to estimate 9 output features which are the projectile position p, velocity v and Euler angles Ψ in the local navigation frame ***n***, $LSTM_{POS}$, $LSTM_{VEL}$, $LSTM_{ANG}$ trained to estimate 3 output features which are respectively the projectile position p, the projectile velocity v and the projectile Euler angles Ψ in the local navigation frame ***n***.

LSTMs are declined in 8 versions presented in Table I, to study the influence of the Min/Max $MM(.)$ and the Standard Deviation $STD(.)$ normalization as well as the influence of the local navigation frame rotation on estimation accuracy.

B. Input data normalization

Network input data normalization is a preprocessing data approach to rescale input data to similar variation ranges while preserving the same distribution and ratios as the original data.

TABLE I
VERSION SPECIFICATIONS: INFLUENCE OF THE NETWORK INPUT DATA NORMALIZATION AND THE LOCAL NAVIGATION FRAME ROTATION.

Name	NORMALIZATION	ROTATION
V_1	No	No
V_2	$MM(\mathcal{T}), MM(\mathcal{M}), MM(\mathcal{P})$	No
V_3	$MM(\mathcal{T}, \mathcal{M}, \mathcal{P})$	No
V_4	$STD(\mathcal{T}), STD(\mathcal{M}), STD(\mathcal{P})$	No
V_5	$STD(\mathcal{T}, \mathcal{M}, \mathcal{P})$	No
V_6	No	Yes
V_7	$MM(\mathcal{T}, \mathcal{M}, \mathcal{P})$	Yes
V_8	$STD(\mathcal{T}, \mathcal{M}, \mathcal{P})$	Yes

Min/Max normalization: Versions V_2, V_3 and V_7 use the Min/Max normalization $MM(.)$ defined as follows:

$$x_{MM} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{10}$$

with x_{max} and x_{min} respectively the maximum and minimum of x. This normalization ranges values in the interval $[0, 1]$.

Standard Deviation normalization: Versions V_4, V_5 and V_8 use the Standard Deviation normalization $STD(.)$ defined as follows:

$$x_{STD} = \frac{x - \mu}{\sigma} \tag{11}$$

with x the quantity to normalize, μ its mean and σ its standard deviation. Thus x_{STD} is a quantity with a zero-mean and a standard deviation of one. This normalization is especially used for input data with different units.

It is important to note that the normalization factors x_{max}, x_{min}, x_{min}, μ and σ are computed before networks training, and are evaluated on the training dataset.

C. Local navigation frame rotation

As presented in Table I, versions V_6, V_7 and V_8 use the local navigation frame rotation. This method, illustrated in Figure 4, aims to rotate the local navigation frame $\boldsymbol{n}$ by a fixed angle γ (local rotated navigation frame $\boldsymbol{n_\gamma}$) such as:

$$x_\gamma = R_\gamma x \tag{12}$$

with $x \in \mathbb{R}^3$ defined in $\boldsymbol{n}$, $x_\gamma \in \mathbb{R}^3$ expressed in $\boldsymbol{n_\gamma}$ and $R_\gamma \in SO(3)$ the transition matrix from the local navigation frame $\boldsymbol{n}$ to the local rotated navigation frame $\boldsymbol{n_\gamma}$. The rotation is applied along the three local navigation frame axes and the angle γ is fixed manually to ensure that the three components of a quantity expressed in the local rotated navigation frame $\boldsymbol{n_\gamma}$ are similar along the three axes. In other words, the local navigation frame rotation is used for the same purpose as normalization, to express a quantity over similar variation ranges.

For example, projectile position variation ranges along the x-axis and z-axis are around several kilometers while along the y-axis, the projectile position varies by a few meters. Express the position in the local rotated navigation frame $\boldsymbol{n_\gamma}$, the position along the three axes are around one kilometer.

All quantities expressed in the local navigation frame $\boldsymbol{n}$ are rotated, i.e. projectile position, velocity and Euler angles.

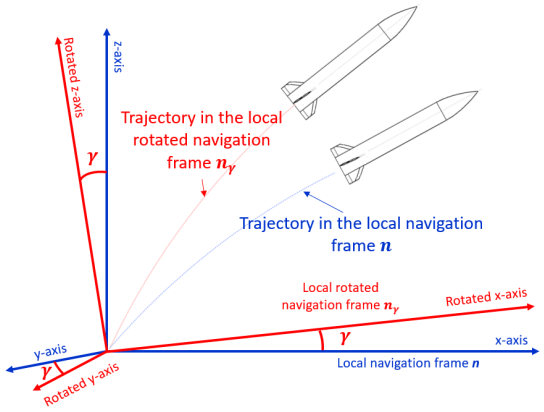


Fig. 4. Local navigation frame **n** and local rotated navigation frame $\boldsymbol{n_\gamma}$.

In other words, during the training step, labels are expressed in the local rotated navigation frame $\boldsymbol{n_\gamma}$ and LSTMs predict trajectories in $\boldsymbol{n_\gamma}$. During testing, LSTMs estimate projectile trajectories in the local rotated navigation frame $\boldsymbol{n_\gamma}$ and then, estimations are moved back to the initial local navigation frame $\boldsymbol{n}$.

D. LSTM training details

Networks $LSTM_{ALL,\ POS,\ VEL,\ ANG,\ V_{1-8}}$ are trained on a training dataset composed by 100 simulations, a validation dataset composed by 10 simulations and a test dataset composed by 20 simulations generated by BALCO presented in part *(III)*. This reduced dataset is defined to evaluate the impact of the normalization and the local navigation frame rotation on estimation accuracy. Moreover, one simulation includes an average of 35 000 time steps.

The batch size is 64 and the window size (SEQ_LEN) is set to 20 timestamp to capture enough long-term dependencies without depending on measurement noise. LSTMs are composed of two layers of 64 and 128 hidden units.

The loss between LSTM estimates and the reference trajectory is evaluated with the Mean Squared Error (MSE) defined as:

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(\widehat{x}_k - x_{ref_k})^2 \tag{13}$$

with $\widehat{x}$ the LSTM estimate and x_{ref} the reference value. Network weights and biases are updated by Adam optimization algorithm [27].

V. RESULTS AND ANALYSIS

This section reports the estimated trajectories of a mortar according to the different networks mentioned in section *(IV)*. LSTM estimates are compared to a classical navigation algorithm: a Dead Reckoning (1)-(3).
A first part *(V-A)* focuses on one mortar trajectory result, then, the LSTM performances are analyzed on the whole test dataset *(V-B)*. Finally, the last part *(V-C)* is centered on the LSTM performance on a larger dataset and on the influence of the IMU error model.

A. Focus on one mortar fire simulation

Figures 5-7 present the estimated position, velocity, and Euler angles and the associated errors for one mortar shot in the test dataset. For readability reasons, three estimation methods are first compared: the Dead Reckoning algorithm (1)-(3), $LSTM_{ALL,V_1}$ *(IV)*, and $LSTM_{ALL,V_6}$ (local navigation frame rotation).

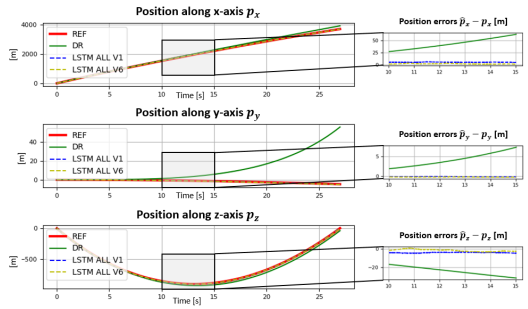


Fig. 5. Estimated projectile position and associated errors [m].

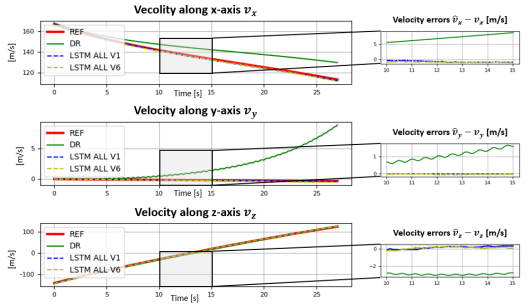


Fig. 6. Estimated projectile velocity and associated errors [m/s].

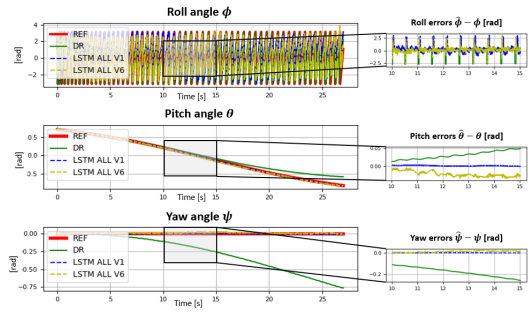


Fig. 7. Estimated projectile Euler angles and associated errors [rad].

As shown in Figures 5 and 6, positions and velocities estimated by LSTMs are significantly more accurate than Dead Reckoning. For the orientation (Figure 7), LSTMs are only precise to estimate the pitch θ and yaw angle ψ. Errors in LSTM roll angle ϕ estimation are due to mortar rotation rate. LSTMs fail to fully capture all roll angle variations. Moreover, according to Figures 5 and 6, rotating the local navigation frame improves projectile position and velocity estimation but slightly degrades pitch angle θ estimation (Figure 7).

B. Analysis on the whole test dataset: 20 mortar shots

In order to validate the previous observations, networks presented in section *(IV)*, $LSTM_{ALL,\ POS,\ VEL,\ ANG,\ V_{1-8}}$, are evaluated on the whole test dataset according to two criteria based on the Root Mean Square Error (RMSE):

$$RMSE_x = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\widehat{x}_k - x_{k,ref}\right)^2} \tag{14}$$

with $\widehat{x}$ the estimated quantity, x_{ref} the reference and N the number of samples.

Success Rate $\mathcal{C}_1$: The first evaluation criterion is the success rate, i. e. number of simulations in the test dataset where a LSTM's RMSE is strictly smaller than the Dead Reckoning's RMSE:

$$\mathcal{C}_1 = \sum_{k=1}^{N_{sim}} RMSE_{LSTM} < RMSE_{DR} \tag{15}$$

with N_{sim} the number of simulations in the test dataset.

Error Rate $\mathcal{C}_2$: The second evaluation criterion is the error rate evaluated such as:

$$\mathcal{C}_2 = \frac{1}{N_{sim}}\sum_{k=1}^{N_{sim}} RMSE_{sim_k} \tag{16}$$

with N_{sim} the number of simulations in the test dataset.

The success rate $\mathcal{C}_1$ of $LSTM_{ALL,POS,VEL,ANG,V_1-V_8}$ are presented in Figure 8-10.(a) and the error rates of LSTMs $\mathcal{C}_{2_{LSTM}}$ and Dead Reckoning $\mathcal{C}_{2_{DR}}$, are presented in Figure 8-10.(b), for position, velocity and Euler angles estimation.

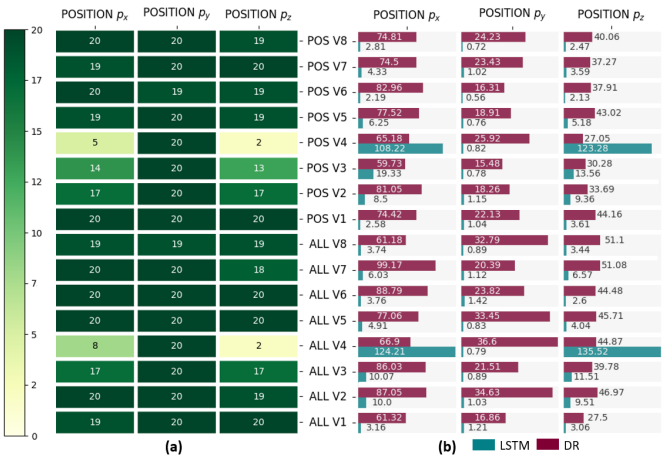


Fig. 8. Position estimation analysis: $\mathcal{C}_1$ criterion *(a)* and $\mathcal{C}_2$ criterion $[m]$ *(b)*.

Position analysis results: According to Figure 8, LSTMs strongly outperform Dead Reckoning for position estimation. Moreover, $LSTM_{POS,V_1-V_8}$, specialized in position estimation exclusively, slightly exceed $LSTM_{ALL,V_1-V_8}$. Normalizations applied to input data affect position estimates differently. Firstly, V_3 and V_4 versions exhibit lower success and error rates than other normalizations. Thus, normalizations

$MM(\mathcal{T},\mathcal{M},\mathcal{P})$ and $STD(\mathcal{T}), STD(\mathcal{M}), STD(\mathcal{P})$ are not appropriate to this application. Secondly, the accuracy of networks with normalization (Min/Max $V_{2,3,7}$ and STD $V_{4,5,8}$) is worse than networks with no normalization $V_{1,6}$ as normalization implies a loss of information. Finally, the Min/Max normalization per feature V_2 is better than a Min/Max normalization for all features V_3, in contrast to the STD normalization (V_5 better than V_4).
Rotating the local navigation frame V_{6-8} improves the position estimation accuracy especially along the z-axis.

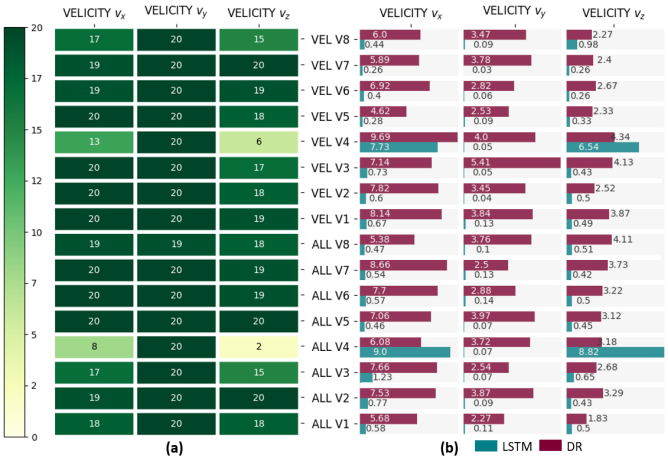


Fig. 9. Velocity estimation analysis: $\mathcal{C}_1$ criterion *(a)* and $\mathcal{C}_2$ criterion $[m/s]$ *(b)*.

Veclocity analysis results: According to Figure 9, similar observations as previously can be formulated. LSTMs clearly outperform Dead Reckoning for velocity estimation. Specialized networks $LSTM_{VEL}$ are a bit better than $LSTM_{ALL}$. The STD normalization for all features V_5 exhibits the best results among the different normalization options investigated, especially for velocity along the z-axis. Moreover, rotating the local navigation frame V_6 significantly improves the projectile velocity estimation along all the three axes.

Fig. 10. Euler angles estimation analysis: $\mathcal{C}_1$ criterion *(a)* and $\mathcal{C}_2$ criterion $[rad]$ *(b)*.

Euler angles analysis results: Euler angles estimation is more mitigated according to figure 10. Focusing on the success rate $\mathcal{C}_1$, LSTMs deteriorate the roll ϕ and pitch θ angles estimates compared to Dead Reckoning, but accurately estimate the yaw angle ψ. In addition, $LSTM_{ANG}$ is less accurate than $LSTM_{ALL}$ for roll ϕ and pitch θ estimation according to criterion $\mathcal{C}_1$, contrary to the yaw angle ψ. As previously, the $STD(\mathcal{T},\mathcal{M},\mathcal{P})$ normalization of $LSTM_{ALL}$ exhibits the best performances for the three Euler angles estimation as well as the local navigation frame rotation.

In summary, end-to-end estimation using LSTM is particularly appropriate for projectile position and velocity estimation. According to the reported results, Figure 8-10, specialized networks do not significantly improve estimation accuracy. Moreover, these results show that $STD(\mathcal{T},\mathcal{M},\mathcal{P})$ normalization is more appropriate to estimate a projectile trajectory. Finally, rotating the local navigation frame is an efficient method to optimize projectile position and velocity estimation.

C. IMU measurements: effects on position estimation

The dataset presented in section *(III)* contains two kinds of inertial readings; *IMU measurements*, used so far, and *IMU DYN measurements*, where sensors are characterized by a dynamic model. This part focuses on the effect of the IMU error model on projectile position estimation. To this end, two LSTMs are trained with the same specifications as $LSTM_{ALL\ V1}$ (no normalization, no rotation):

- $LSTM_{IMU}$ trained with *IMU measurements* with no normalization and no rotation in order to estimate the projectile position, velocity and orientation.
- $LSTM_{IMU\ DYN}$ trained with *IMU DYN measurements* with the same characteristics as $LSTM_{IMU}$.

Both networks are trained on 4 000 mortar fire simulations, validated on 400 and tested on 400.

– $\boldsymbol{LSTM_{IMU}}$ **:** Figure 11 presents the RMSE (14) returned by $LSTM_{IMU}$ as a function of the Dead Reckoning RMSE for projectile position estimation on the 400 simulations in the test dataset.

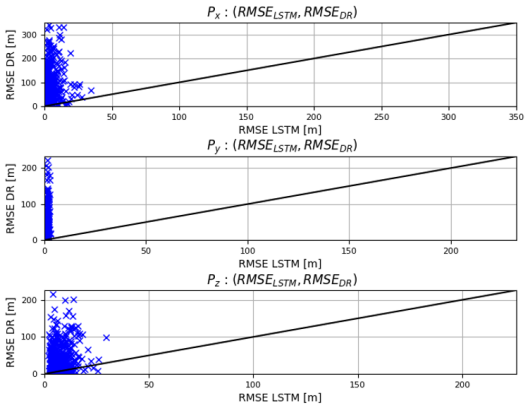


Fig. 11. $LSTM_{IMU}$: $(RMSE_{LSTM}, RMSE_{DR})$ [m] for 400 mortar fire simulations.

According to Figure 11, $LSTM_{IMU}$ significantly outperforms Dead Reckoning for position estimation along the three axes, as most of the markers are located in the upper part. $LSTM_{IMU}$ accurately estimates the projectile position, especially along the y-axis.

Figure 12 presents position errors along at the impact point (position errors (p_x, p_y) at the final time of a shot) for all 400 simulations in the test dataset.

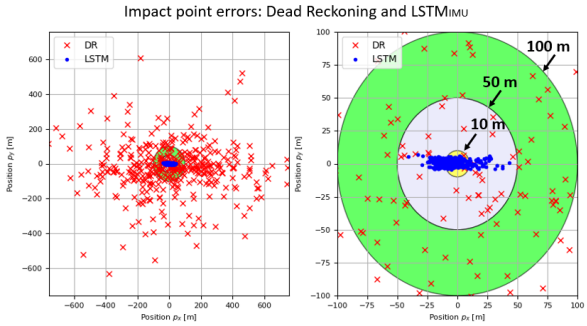


Fig. 12. Errors at impact point obtained by $LSTM_{IMU}$ (blue dot) and the Dead Reckoning (red dot).

As shown in figure 12, most $LSTM_{IMU}$ errors at the impact point are located inside a 50-meter error disk (gray disk), while Dead Reckoning errors are much larger. More precisely, 249 of the 400 shots have errors at the impact point less than 10m with the LSTM, 149 shots with errors between 10 and 50 m and 2 shots with errors between 50 and 100 m. Concerning the Dead Reckoning, 293 of the 400 simulations have errors greater than 100 m. Thus, on a large dataset, with various flight parameters, $LSTM_{IMU}$ succeeds in accurately estimate the projectile position.

– $\boldsymbol{LSTM_{IMU\ DYN}}$ **:** Figures 13 and 14 respectively show $(RMSE_{LSTM_{IMU\ DYN}}, RMSE_{DR})$ for projectile position estimation and errors at the impact point.

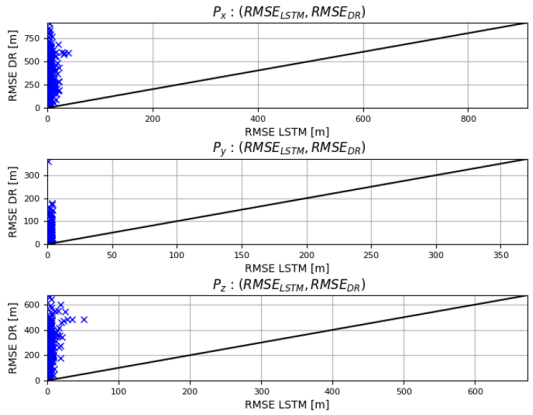


Fig. 13. $LSTM_{IMU\ DYN}$: $(RMSE_{LSTM}, RMSE_{DR})$ [m] for 400 mortar fire simulations.

Dead Reckoning completely diverges for position estimation from IMU DYN data. Conversely, LSTM accurately estimates

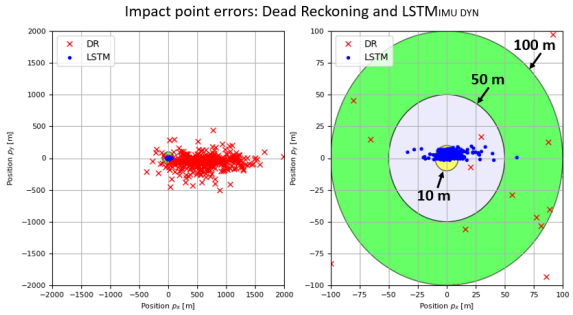


Fig. 14. Errors at impact point obtained by $LSTM_{IMU\ DYN}$ (blue dot) and the Dead Reckoning (red dot).

the projectile position: from Figure 13, most of the markers are located in the upper part, and from Figure 14, most errors at impact point are less than 50m. Concerning the $LSTM_{IMU\ DYN}$, 266 of the 400 shots have errors at the impact point less than 5m while with the Dead Reckoning, 378 of the 400 shots have errors greater than 100m. Therefore, despite sensor dynamics, the LSTM is still able to estimate the projectile position.

In summary, according to Figures 11 and 14, a LSTM is able to accurately estimate a projectile position despite dynamic inertial data and a various range of flight characteristics. In addition, generic guided mortar accuracy is around 10 to 70m and LSTM estimation results improve these accuracies.

Conclusion

This paper presents a deep learning approach to estimate a mortar trajectory in a GNSS-denied environment. LSTMs are trained only from inertial measurements, flight parameters and a time vector. Different normalizations are applied to input data as well as the local navigation frame rotation during the training step, in order to deal with the different variation ranges along the three axes.

According to the reported results, LSTMs are accurate to estimate projectile positions and velocities compared to a classical navigation algorithm. Moreover, LSTMs are still accurate to estimate projectile positions even with dynamic inertial measurements and outperform the accuracy of generic guided mortars.

Currently, this estimation method aims to be tested on real data. Moreover, although not presented here, this method is generalized to other kinds of projectiles: 155mm shells, 40mm projectile and Basic Finner.

Now, the idea is to develop Deep Kalman filters by integrating LSTM models into a Kalman filter. In other words, the idea is to use a LSTM to estimate one model of the Kalman filter such as the prediction model or the observation model. For example, in the case of an Extended Kalman Filter to estimate a projectile trajectory from IMU measurements and corrected by GNSS observations, LSTM estimates can be used as GNSS measurements, allowing to bypass the GNSS and avoid any decoying or jamming problem.

REFERENCES

[1] Paul Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition*. 2013.

[2] Gregory Duckworth and Edward Baranoski. Navigation in GNSS-denied environments: Signals of opportunity and beacons. In *Proceedings of the NATO Research and Technology Organization (RTO) Sensors and Technology Panel (SET) Symposium*, 2007.

[3] Alicia Roux, Sébastien Changey, Jonathan Weber, and Jean-Philippe Lauffenburger. Projectile trajectory estimation: performance analysis of an Extended Kalman Filter and an Imperfect Invariant Extended Kalman Filter. In *2021 9th International Conference on Systems and Control (ICSC)*, pages 274–281, 2021.

[4] Christophe Combettes, Sébastien Changey, Ronan Adam, and Emmanuel Pecheur. Attitude and velocity estimation of a projectile using low cost magnetometers and accelerometers. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 650–657, 2018.

[5] Sébastien Changey, Emmanuel Pecheur, Loic Bernard, and all. Real time estimation of projectile roll angle using magnetometers: In-flight experimental validation. In *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pages 371–376, 2012.

[6] Aurélien Fiot, Sébastien Changey, and Nicolas Petit. Attitude estimation for artillery shells using magnetometers and frequency detection of accelerometers. *Control Engineering Practice*, 122:105080, 2022.

[7] Adrian Carrio, Carlos Sampedro, Alejandro Rodriguez-Ramos, and Pascual Campoy. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, 2017, 2017.

[8] Peter Svenmarck, Linus Luotsinen, Mattias Nilsson, and Johan Schubert. Possibilities and challenges for artificial intelligence in military applications. 05 2018.

[9] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[10] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[11] Ralf Staudemeyer and Eric Rothstein Morris. Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

[12] Pierre Wey, Daniel Corriveau, Thomas Saitz, Wim de Ruijter, and Peter Strömbäck. BALCO 6/7-DoF Trajectory Model. 05 2016.

[13] Hui Zhao and Zhong Su. Real-time estimation of roll angle for trajectory correction projectile using radial magnetometers. *IET Radar, Sonar & Navigation*, 14(10):1559–1570, 2020.

[14] Liangliang An, Liangming Wang, Ning Liu, Jian Fu, and Yang Zhong. A Novel Method for Estimating Pitch and Yaw of Rotating Projectiles Based on Dynamic Constraints. *Sensors*, 19(23), 2019.

[15] Sébastien Changey and all. A mixed extended-unscented filter for attitude estimation with magnetometer sensor. In *2006 American Control Conference*, pages 6 pp.–, 2006.

[16] Nabil Jardak, Ronan Adam, and Sebastien Changey. A Gyroless Algorithm with Multi-Hypothesis Initialization for Projectile Navigation. *Sensors*, 21:7487, 11 2021.

[17] YuLong Zhang, ZiJie Dai, LongFei Zhang, ZhengYi Wang, Li Chen, and YuZhen Zhou. Application of artificial intelligence in military: From projects view. In *2020 6th International Conference on Big Data and Information Analytics (BigDIA)*, pages 113–116, 2020.

[18] Antonio Carlo. Artificial intelligence in the defence sector. In *International Conference on Modelling and Simulation for Autonomous Systems*, pages 269–278. Springer, 2020.

[19] Zhiyuan Shi, Min Xu, Quan Pan, Bing Yan, and Haimin Zhang. LSTM-based Flight Trajectory Prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[20] Nikolai Gaiduchenko, Pavel Gritsyk, and Yanka Malashko. Multi-step ballistic vehicle trajectory forecasting using deep learning models. In *2020 International Conference Engineering and Telecommunication (En&T)*, pages 1–6, 2020.

[21] SeongHyeon Park, Byeongdo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. *CoRR*, abs/1802.06338, 2018.

[22] Kristian Aalling Sørensen, Peder Heiselberg, and Henning Heiselberg. Probabilistic maritime trajectory prediction in complex scenarios using deep learning. *Sensors*, 22(5), 2022.

[23] Abdulrahman Al-Molegi, Mohammed Jabreel, and Baraq Ghaleb. Stf-rnn: Space-time features-based recurrent neural network for predicting people's next location. 12 2016.

[24] Emad Barsoum, John Kender, and Zicheng Liu. Hp-gan: Probabilistic 3d human motion prediction via gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[25] Sami Jouaber, Silvere Bonnabel, Santiago Velasco-Forero, and Marion Pilte. NNAKF: A neural network adapted Kalman filter for target tracking. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4075–4079, Toronto, France, June 2021. IEEE.

[26] Li-he Hou and Hua-jun Liu. *An End-to-End LSTM-MDN Network for Projectile Trajectory Prediction*, pages 114–125. 11 2019.

[27] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Promises and Limitations of Self-supervised Learning for Automatic Speech Processing

Lucas Maison
Laboratoire Informatique d'Avignon (LIA)
Avignon Université
lucas.maison at univ-avignon.fr

Marcely Zanon Boito
Laboratoire Informatique d'Avignon (LIA)
Avignon Université
marcely.zanon-boito at univ-avignon.fr

Yannick Estève
Laboratoire Informatique d'Avignon (LIA)
Avignon Université
yannick.esteve at univ-avignon.fr

***Abstract*—Self-supervised learning (SSL) has recently been successfully introduced as a training strategy for Transformer-based neural models. Thanks to this approach, these models are now able to construct speech representations by using only audio data, without any manual labels (i.e. no supervision). Once trained, they can be leveraged for training competitive end-to-end models for speech processing with smaller amounts of annotated data. Moreover, when the available annotated data is plenty, automatic speech recognition (ASR) and translation (AST) systems based on these SSL models are now the new state of the art. In this work, we are interested in their application in challenging settings that are relevant for security. We measure the robustness of a French-based SSL model to African accent, and we present some promising but limited results for speech translation without the use of transcriptions.**

***Index Terms*—automatic speech recognition, speech translation, self-supervised learning, speech processing, security**

I. Introduction

Speech recognition has been dominated by data-driven approaches for almost four decades. From the 80s until a few years ago, automatic speech recognition (ASR) systems were based on the use of three kinds of knowledge. The first one was captured by the acoustic models, usually based on a Hidden Markov Model (HMM) combined to a Gaussian Mixture Model (GMM) or more recently to a Deep Neural Network (DNN). The acoustic models were designed to compute the likelihood of the presence of a phoneme (the speech unit that discriminates a word in a language) according to the audio signal. The second knowledge was represented by a pronunciation dictionary, in order to map a sequence of phonemes to one or several words. The last knowledge was captured by a language model, in order to compute the probability to observe a sequence of words in a language.

At that time, acoustic and language models were mainly statistical models, and that is why we say that such approaches are data-driven. To get such models accurate and robust, a large amount of training data is needed, following the "there is no better data than more data" paradigm. During the last decade, DNNs for both acoustic and language models have replaced the GMMs, and neural end-to-end approaches eliminated the HMMs, but the need for data remained, at least, the same.

Among this data, the most *important* modality is having speech audio data with its manual transcription. This paired audio/text data is necessary to train ASR systems. It is also a very costly data that can be rare for many languages.

Self-supervised Learning (SSL) has been recently proposed as an interesting alternative for data representation learning. Proven useful learned representations can be found both in vision [1], [2] and in NLP [3], [4]. The attractiveness of SSL in general, and SSL from speech in particular, is that it can leverage huge amounts of unannotated data, which is cheaper than the audio/text data used by classical systems. This leveraging can be done by resolving pseudo-tasks, which do not require human annotation, as pre-training a feature extractor, which is then used to extract useful speech representations for the real (downstream) tasks. The two most commonly used approaches for SSL from speech are *Autoregressive Predictive Coding* (APC) and *Contrastive Predictive Coding* (CPC). The former's pseudo-task is considering the sequential structure of speech, and predicting information about a future frame [5], [6], whereas the latter's consists of distinguishing a future speech frame from distractor samples [7]–[9] which is an easier learning objective compared to APC. These representations have been proven to improve the performance in several speech tasks [10], while being less sensitive to domain and/or language mismatch [11] and being transferable to other languages [12].

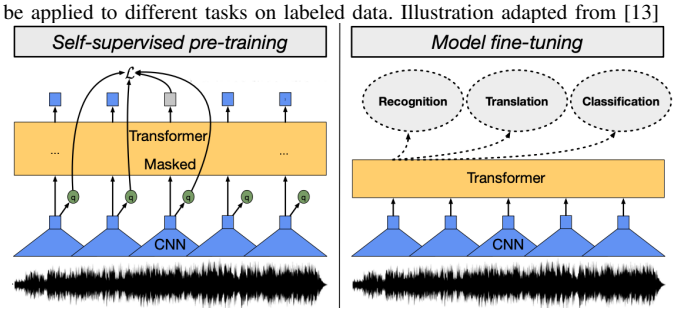
SSL opens new perspectives to build and deploy ASR for low-resource languages, or low-resource domains. Such an approach speeds up the creation of a new ASR system, and reduces its cost, since a significantly smaller amount of annotated data is necessary to get competitive results in comparison to the previous state of the art.

Speech recognition can be involved in many tasks for security purposes. It is also the case for speech translation, on which SSL is also useful, especially for neural end-to-end architectures.

This paper discusses current limitations of the wav2vec 2.0 models, focusing on two applications relevant for security: automatic speech recognition (ASR) and automatic speech translation (AST). It is organized as follows: Section II presents a high-level summarization of the technology behind these SSL models. Section III discusses their application to ASR, especially to process accented French. Section IV discusses their application to AST in an extreme low resource scenario. Section V presents our final remarks.

II. Self-supervised Models for Speech: the wav2vec 2.0 architecture

Fig. 1. Illustration of the wav2vec 2.0 framework (left), which jointly learns contextualized speech representations and an inventory of discretized speech units during pre-training on unlabeled data. The fine-tuning step (right) can be applied to different tasks on labeled data. Illustration adapted from [13]



The *wav2vec 2.0* proposed by [14] is an extension of [8], [9], [15]. Depicted in Figure 1, it consists of a multi-layer convolutional feature encoder $g_{enc} : \mathcal{X} \rightarrow \mathcal{Z}$, which transforms raw input audio x into latent speech representations $z = \{z_1, z_2, ..., z_T\}$ for T time-steps. These latent features are then fed into a Transformer $g : \mathcal{Z} \rightarrow \mathcal{C}$ for building contextualized representations $c = \{c_1, c_2, ..., c_T\}$ that capture the information of the whole sequence.

The wav2vec 2.0 also performs discretization on the output of the feature encoder z_t to q_t by using a quantization module $\mathcal{Z} \rightarrow \mathcal{Q}$. The model's Transformer network learns contextualized representations directly from continuous speech representations (z) via time-step masking and a contrastive task (CPC) which identifies the true quantized latent audio representation in a set of distractors for each masked time step. This consequently allows [14] to train wav2vec 2.0 in an end-to-end fashion, in which all its components are trained jointly toward minimizing an objective (Equations 1, 2, 3).

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d \tag{1}$$

$$\mathcal{L}_m = -\log \frac{\exp(sim(c_t, q_t)/\kappa)}{\sum_{\tilde{q} \sim Q_t} \exp(sim(c_t, \tilde{q})/\kappa)} \tag{2}$$

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v} \tag{3}$$

In Equation 1, the training objective is defined as the sum of two components:

- **Contrastive Loss** $\mathcal{L}_m$ which is defined in Equation 2. Particularly, given c_t centered over the masked time step t, the model is trained to contrast the true quantized latent speech representation q_t from K quantized latent distractors $\tilde{q} \in Q_t$ uniformly sampled from other masked time steps of the same utterance. $sim(a, b) = a^\top b / ||a|| ||b||$ is the *cosine similarity* between context representations c_t and quantized latent speech representations q_t.
- **Diversity Loss** $\mathcal{L}_d$ which is defined in Equation 3. It helps to increase the use of the quantized codebook representations, encouraging the model to equally use all the V entries in each of G codebooks by maximizing the entropy of the averaged softmax distribution over the codebook entries for each codebook $\bar{p}_g$ across a batch of utterances. In Equation 1, $\mathcal{L}_d$ is scaled by α, which is a tunable hyperparameter.

Masking: time-step masking mentioned earlier is done by randomly sampling without replacement a certain proportion p of all time steps to be starting indices and then mask the subsequent M consecutive time steps for every sampled index. Note that spans may overlap, and inputs to the quantization module are not masked.

Fine-tuning: the wav2vec 2.0 framework also allows fine-tuning the pre-trained model directly on ASR (or AST, or speech classification) labeled data by stacking a linear projection layer initialized randomly on top of the context network. Fine-tuned models are optimized by minimizing a Connectionist Temporal Classification (CTC) loss. It has been shown [14] that fine-tuning even on only 10 minutes of labeled training data (48 recordings of 12.5 seconds on average) helps achieve a respective Word Error Rate (WER) of 4.8% and 8.2% on the test-clean and test-other sets of the Librispeech corpus (read speech).

III. ASR application: from standard to accented speech

In recent years, huge progress has been achieved in the domain of automatic speech recognition (ASR). Neural models now reach human performance on the English ASR task: best systems reach a Word Error Rate (WER) of 5.8% and 11% on Switchboard and CallHome datasets respectively, whereas performance of human annotators is estimated to be around 5.9% and 11.9% [16].

However, it has been shown that the performance of these models can decrease drastically when they are used in new or non-ideal conditions. For example, a noisy environment or the accent of a speaker can both impact the quality of the transcription [17], [18]. It is important to study the robustness of systems to such conditions, because these are likely to be encountered in real settings. Furthermore, support of accented speech is mandatory if one aspire to build an inclusive, general-purpose ASR system. It could also be of interest for security or intelligence agencies wanting to support a large spectrum of accents. In this section, we focus on the case of accented speech in French, presenting results for ASR models trained on standard and accented French speech.

A. Models and datasets

Focusing on the French language, the *LeBenchmark* initiative [19], [20] is a proeminent work in the area of SSL benchmarking. It provides evaluation recipes for four downstream tasks (ASR, AST, automatic emotion recognition, spoken language understanding) alongside with wav2vec 2.0 models

Fig. 2. ASR results (WER, the lower the better) over the two test sets for models fine-tuned on CommonVoice.

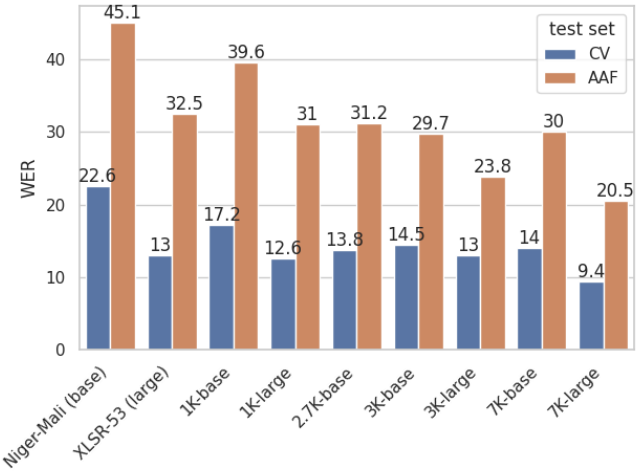
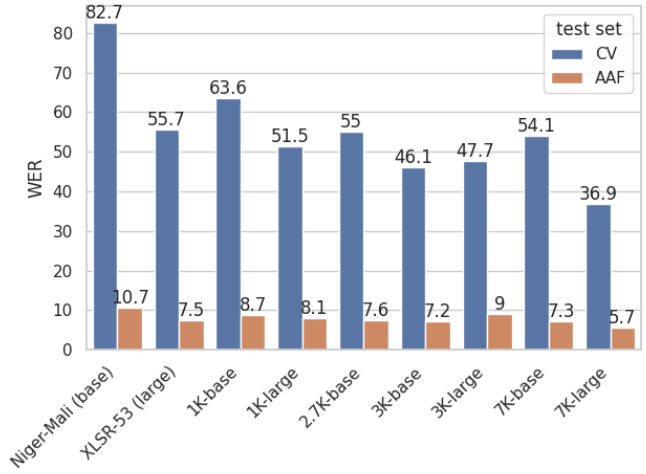


Fig. 3. ASR results (WER, the lower the better) over the two test sets for models fine-tuned on African Accented French.



of various sizes, and pre-trained using different amounts of speech audio.

For our ASR models, we use the following models from the *LeBenchmark*: LB-1K-base/large, LB-2.7K-base, LB-3K-base/large, and LB-7K-base/large, which were pre-trained on respectively 1,096, 2,773, 2,933 and 7,739 hours of French audio [20]. The "base" refers to the standard model architecture from [14] that has 95 millions parameters, while the "large" refers to their larger architecture that presents greater capacity (317 millions parameters).

In addition to these French models, two multilingual models were tested. The first one, Niger-Mali [21], is a base model pre-trained on 641 h of speech in 5 languages, including 111 h of accented French. The second one, XLSR-53 [22], is a large model pre-trained on 56k hours of speech in 53 languages, including 1,429 h of French.

Each pre-trained wav2vec 2.0 model acts as a speech encoder, which is optimized for the ASR task together with an additional feed-forward network. This head network consists of three linear layers with 768 or 1,024 neurons for a *base* or *large* model, respectively. Each linear layer is followed by batch normalization and a Leaky ReLU [23] activation function. We use dropout with $p = 0.15$ between each linear layer. At last, a final linear layer projects the output into token space, and log-softmax is applied to obtain probabilities of each token. We use individual characters as tokens. Note that we do not apply any language model besides our end-to-end model.

We employ the `SpeechBrain` [24] toolkit for all our experiments. All models are fine-tuned during 50 epochs using the CTC loss, and Adam [25] and Adadelta [26] optimizers are used to update the weights, one for the wav2vec 2.0 model and one for the additional top layers.

We use two different datasets in this study. The first one is the French subset of CommonVoice (CV) 3.0 [27] that comprises 56 h of recordings. It represents our reference dataset of unaccented speech. The second one is the African Accented French (AAF) dataset [28]. It is composed of 13 h of speech [1]. Speakers are from Cameroon, Chad, Congo, Gabon and Niger, and they speak French with a strong African accent. It should be noted that this latter dataset was used as part of the pre-training data for the following *LeBenchmark* models: LB-3K-base/large, LB-7K-base/large.

B. Fine-tuning ASR models: from non-accented to accented speech

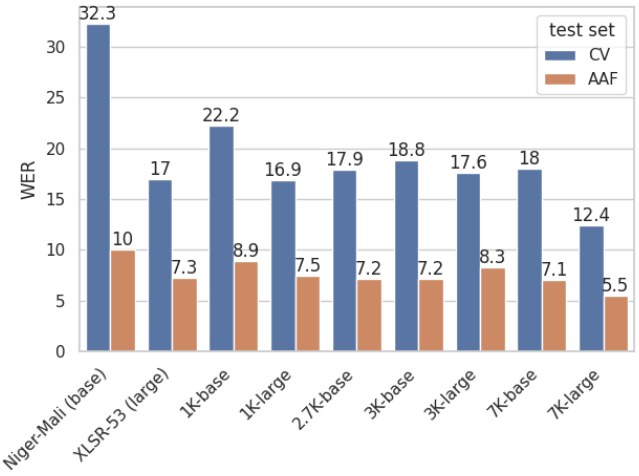
To assess the robustness of the pre-trained wav2vec 2.0 models with respect to accent variability, we fine-tune each model on the train split of CV. Then, we evaluate the resulting models on both the test split of CV and AAF. Results are shown on Figure 2. The trend we observe is that, the more speech data we use for pre-training, the best the model performs, meaning that it better specialized its speech representations.

We also notice that, thanks to their increased capacity, "large" models perform much better than "base" ones. Moreover, multilingual models perform rather badly when compared with French models trained with similar amounts of speech. The best model (LB-7K-large) obtains a WER of 9.37% on CV, which is comparable to the best scores reported in [20] on the same dataset (CV). However, this same model scores 20.47% on AAF. Moreover, all the tested models follow a similar trend, with a WER that doubles on AAF compared to CV. This means that these models are likely to make twice as many transcription errors when used by non-native speakers rather than native speakers.

In order to improve the robustness of the models on accented speech, we restart the experiment and fine-tune each pre-

[1]The original dataset is larger, but we excluded portions containing annotation errors.

Fig. 4. ASR results (WER, the lower the better) over the two test sets for models fine-tuned on a mixed dataset (CV+AAF).



trained model on the train split of AAF, and evaluate the resulting models on the same test sets as before. We can see on Figure 3 that doing so greatly reduce the WER on AAF (-75% on average), with our best model now scoring 5.72%. However, this large improvement comes at the cost of a similar performance degradation on CV (+282% on average). This demonstrates that fine-tuning directly on accented speech is beneficial if we desire to transcribe a particular accent, but should not be done if the goal is to build an all-purpose system.

Finally, we created a mixed dataset of accented and native speech by taking the full AAF training set and an equal amount of speech from the CV training set. We use this new dataset to fine-tune the models. Results of evaluation are shown on Figure 4. We can see that fine-tuning on this mixed dataset allows the models to reach good performance on both accented and non-accented speech. Our best model scores 12.38% and 5.47% on CV and AAF respectively. Compared to the models fine-tuned on CV only, these models reach much lower WER on AAF (-75% on average), while only suffering mild performance degradation on CV (+32% on average).

C. Discussion of results

It may seem surprising that the "Niger-Mali" model does not obtain a good performance despite being pre-trained on a large amount of accented French. We believe that the most important factor contributing to low WER is the amount of French audio in the pre-training dataset. The Niger-Mali model is the one with the lowest quantity of French seen during pre-training (111 h), thus explaining its poor score.

The presence of accented French in the pre-training dataset may still play an important role: we can see on Figures 2, 3, and 4 that LB-3K-base is achieving slightly better results on AAF compared to LB-2.7K-base, the main difference between these two models being the presence of accented speech in the pre-training dataset of the former.

The second multilingual model has been pre-trained on much more data (56k hours), but only 1,429 h of French. Its scores no better than the other *large* models pre-trained on much smaller (but French only) datasets. This seems to indicate that multilingual models, despite obtaining good performances on a variety of languages, are not suited for recognizing accented speech.

In summary, in this section we illustrated existing limitations of ASR models for transcribing accented speech in French. We experimented with two multilingual wav2vec 2.0 models, and seven French models, comparing the performance of obtained ASR systems in standard and accented French speech. We find that models fine-tuned on native speech only are not robusts to accent variation, but that incorporating accented data in the fine-tuning dataset greatly improve robustness.

IV. AST Application: SSL models for low-resource end-to-end AST

Traditionally, the speech translation task is defined in a cascaded fashion: the speech is first transcribed by an ASR model, and then a text-to-text machine translation (MT) module produces the final translation in the target language. The limitations of this approach for AST includes the error propagation between the ASR and MT modules, the omission of speech cues that could disambiguate the information given to the MT module, and the need for both a considerable amount of transcribed *and* translated data.

Going beyond the practical time and money constrains for producing this data in non-mainstream languages in order to train and deploy cascaded systems, and the cost of training the systems themselves, it is also important to be aware that not all languages present a standard written form. Indeed, most of the world's languages are not actively written, even the ones with an official writing system [29]: these are called *oral languages*.

This is one reason behind the recent motivation of the speech community to investigate *end-to-end* approaches for AST [30], [31]. We define end-to-end AST as a single optimized model that receives as input speech and produces as output textual translations. Optionally, these models can be jointly optimized for producing transcriptions as well, when these are available during training. This joint training was shown to increase translation performance [32], [33].

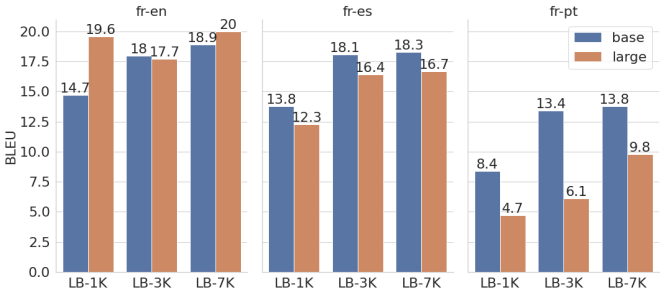
In this section we shed light on some limitations of SSL-based end-to-end AST models for processing oral-languages, for which the amount of available data is limited. This is relevant in the context of security because a government or an organization might aim to deploy AST models for minority or dialect languages in areas of particular interest. In these cases, the amount of available data is usually limited. Ideal AST systems for security should thus be able to work in *low-resource settings.*

This section is organized as follows. We first validate our AST architecture by producing results for three language pairs in the `mTEDx` dataset [34] (Section IV-A).

TABLE I
STASTISTICS FOR THE MTEDX FR-{EN,ES,PT} DATASET.

train						valid		test	
en		es		pt					
# spk	duration	# spk	duration	# spk	duration	# spk	duration	# spk	duration
250	45:04	196	32:30	112	20:01	12	1:38	10	1:33

Fig. 5. AST results (BLEU scores, the higher the better) over the test set for the three language pairs, and using the base and large wav2vec 2.0 models.



These languages have decreasing amounts of available parallel data: French-English (48 h), French-Spanish (35 h), French-Portuguese (23 h).

Having defined these *mid-to-low-resource baselines*, in Section IV-B, we explore the case of the Tamasheq dataset presented in this year's IWSLT campaign [35]. The challenge is producing translation, without available transcription, having only 17 hours of speech in Tamasheq aligned to French translations. In this case, we illustrate how general purpose SSL models fail to produce exploitable representations. Lastly, in Section IV-C we summarize our findings on the use of SSL models for low-resource end-to-end AST.

A. AST in mid-to-low-resource settings

The presented end-to-end AST models are similar to the end-to-end ASR model architecture presented in Section III. They are implemented on `SpeechBrain` [24], being made of a wav2vec 2.0 as a *foundation block*, followed by a linear projection, and a Transformer Decoder [36]. The weights for the wav2vec 2.0 speech encoder block are initialized from the pre-trained SSL models available in the *LeBenchmark* model collection [19], [20].[2] The model is trained on the negative log likelihood loss, and two different instances of the Adam optimizer manage the weight updates: one dedicated to the wav2vec 2.0 block, the other one to the following layers.

As aforementioned, we train mid-to-low-resource baselines using the `mTEDx` dataset. We use as source the French language (speech), and as target languages (text): English (en), Spanish (es) and Portuguese (pt). The resulting language pairs share validation and test sets, but they vary on the amount of available training data. This information (duration), together with the number of speakers (# spk), is presented in Table I.

[2]Available at https://huggingface.co/LeBenchmark

Figure 5 presents the AST results[3] using the three language pairs, and three different wav2vec 2.0 models: LB-1K, LB-3K and LB-7K. These models differ on the amount of training data used during SSL pre-training, with 1K corresponding to approximately 1,000 hours of speech. For each model, we experiment with both *base* and *large* architecture sizes.[4]

Looking at the results using **base** wav2vec 2.0 models (Figure 5, darker bars), we notice that AST models trained in all language pairs benefit from having SSL wav2vec 2.0 models trained using more data: using LB-7K-base and LB-3K-base as foundation blocks seem to be clearly superior compared to AST models that used LB-1K-base. We however, do not observe a very clear distinction between LB-7K-base and LB-3K-base, which might be due to the wav2vec 2.0 model reaching the limits of its own capacity [20].

Focusing on the **large** models (Figure 5, brighter bars), we notice that the trend is not the same for all languages. For English, it seems to still exist some benefit on employing these larger models, compared to their base counterparts: BLEU scores are higher or equivalent. For the other two languages, we notice that performance using large models is inferior to the one reached by their base counterparts.

For these languages, we have less training examples compared to English: for Spanish we have only 35 h, and for Portuguese only 23 h. We thus believe that this discrepancy in performance between base and large models for these languages might be related to the amount of available data, since for large wav2vec 2.0 models we have an additional 221.2 million trainable parameters.[5] The lack of available data might result on these extra parameters not being properly fine-tuned, thus resulting in the observed deterioration in performance. We believe that the fine-tuning of base models might be more *realistic* in settings of data scarcity, as the overhead caused by the extra parameters in large architectures seem to be excessive for models working with less than 50 hours of speech.

Finally, it is also important to highlight that the results obtained for our baselines in this work are considerably higher compared to results obtained with AST models trained *without* SSL models as a foundation block. In [34], and for the same dataset, they reach BLEU scores of 8.9, 10.6 and 7.9 for English, Spanish and Portuguese respectively.

Summarizing, in this section we presented end-to-end AST models in mid-to-low-resource settings. For the language pair with the most available speech data (fr-en) we observe benefits on having large pre-trained SSL models, and we reach acceptable BLEU scores compared to the literature [38], [39]. This finding however does not hold as we reduce the amount of trainable data: even 35 h in Spanish seems not to be enough to fully fine-tune a large wav2vec 2.0 architecture, and results for large models drag behind the results for base models.

[3]BLEU4 scores computed using `sacreBLEU` [37].

[4]There are approximately 221.2 million extra parameters in the large architecture.

[5]In this work we do not explore partially freezing the wav2vec 2.0 blocks.

The challenge of low-resource AST is illustrated with the clear performance drop between our three setups: *fr-en* reaches higher performance than *fr-es* (data reduction of 13 h), and the latter outperforms *fr-pt* models (data reduction of 12 h compared to Spanish, 25 h compared to English). This highlights the existence of a minimal amount of data needed in order to make the training of end-to-end AST architectures based on SSL models exploitable.

B. Use case: AST for Tamasheq

We now present our experiments for the Tamasheq-French dataset in the context of the IWSLT 2022 low-resource speech translation track. The dataset contains 17 h of speech in the Tamasheq language, which corresponds to 5,829 utterances translated to French [40]. Additional audio data was also made available through the *Niger-Mali audio collection*: 224 h in Tamasheq and 417 h in geographically close languages (French from Niger, Fulfulde, Hausa, and Zarma).[6] For all this data, the speech style is radio broadcasting, and the dataset presents no transcription.

We start by training AST models that use wav2vec 2.0 models simply as feature extractors: the output of these SSL architectures replaces commonly used mel filterbank (MFB). This was shown to result in a considerable performance boost using the mTEDx dataset in [19], and we choose this approach since our results from the previous session hint that end-to-end fine-tuning for the AST task requires more data than the available 17 h.

In these settings, we compare two general purpose wav2vec 2.0 models – the multilingual XLSR-53 [22] and the French LB-7K-large [20] – against two smaller base models trained in the target language: Tamasheq-only, trained on 243 h of Tamasheq, and Niger-Mali, trained on the totality of the Niger-Mali audio collection (641 h).

Our AST models that use wav2vec 2.0 models as feature extractors are very close to the recipe for low-resource ST from wav2vec 2.0 features described in [20]. We use the *fairseq s2t* toolkit [41] for training an end-to-end AST Transformer model [36] with 4 heads, dimensionality of 256, inner projection of 1,024, 6 encoder and 3 encoder layers. The Transformer is preceded by a 1D convolutional layer (k=5, stride=2) for down-projecting the wav2vec 2.0 large (1,024) or base (768) features into the Transformer input dimensionality. These models are trained for 500 epochs using the Adam optimizer with 10k warm-up steps. For decoding, we use beam search with a beam size of 5. We generate a 1k unigram vocabulary for the French text using *Sentencepiece* [42], with no pre-tokenization. Lastly, we include baseline results that replace wav2vec 2.0 features by 80-dimensional MFB features. In this setting, the CNN preceding the transformer encoder is identical from the one in [20].

AST results using the four wav2vec 2.0 models as feature extractors are presented in Table II. For each model other than Tamasheq-only, we investigate fine-tuning on a *task-agnostic* fashion for approximately 20,000 updates on all available Tamasheq speech (243 h). This fine-tuning is supposed to reduce performance issues related to domain shift, however, in our setting we do not notice a significant performance gap between fine-tuned models and their pre-trained counterparts.

TABLE II
BLEU4 RESULTS FOR TAMASHEQ-FRENCH AST.

wav2vec 2.0 model	Fine-tuning	valid	test
None (MFB)	-	2.22	1.80
LB-FR-7K	-	2.36	1.80
LB-FR-7K	Task-agnostic	2.48	1.92
XLSR-53	-	2.05	1.42
XLSR-53	Task-agnostic	1.99	1.91
Niger-Mali	-	2.81	**2.68**
Niger-Mali	Task-agnostic	2.94	**2.57**
Tamasheq-only	-	2.99	**2.42**

Regarding the overall very low AST performance,[7] it is notable that the results obtained by using general purpose wav2vec 2.0 models are not very different from the baseline results (MFB). This was also observed in the literature: it seems wav2vec 2.0 models tend to perform poorly as feature extractors in low-resource settings, compared to MFB [43]. Finally, although performance is poor regardless of the feature extractor, the wav2vec 2.0 models trained on target data seem to output superior features for Tamasheq-French AST. This is despite the fact they are trained with considerably smaller quantities of data compared to the large and general purpose models.

Based on this finding, the best AST results we reported on [21] used these *smaller* wav2vec 2.0 models on an end-to-end fashion. However, as expected by our mid-to-low-resource baseline results from last section, there are not enough training hours to successfully fine-tune an entire wav2vec 2.0-based AST architecture for Tamasheq-French.[8]

We circumvent this by exploring the representation from intermediate layers: previous work [44] has shown that the middle layers inside the Transformer Encoder inside the wav2vec 2.0 architecture contain a higher abstraction level with respect to the speech signal, being more useful for end-to-end ASR fine-tuning compared to the last layers. Inspired by that finding, we experimented pruning the last layers of the wav2vec 2.0 model, which reduced the amount of trainable parameters. Our final model, and the best result for the IWSLT 2022 low-resource task, was an end-to-end wav2vec 2.0-based AST model using the Tamasheq-only model. It comprised only 7 Transformer layers (out of 12) on its wav2vec 2.0 foundation block, and **it achieved a BLEU4 of 6.0**.

C. SSL models for low-resource end-to-end AST

Throughout this section, we illustrated how the performance of wav2vec 2.0-based AST models drops in settings of data scarcity. One important aspect of our results is the complete

[6]https://demo-lia.univ-avignon.fr/studios-tamani-kalangou/

[7]High-resource end-to-end speech translation BLEU4 scores range from 19 to 30 in the last editions of IWSLT [38], [39].

[8]The best BLEU score for an end-to-end wav2vec 2.0 AST model was of 2.34, by using the Tamasheq-only wav2vec 2.0 model.

lack of transcription we impose to our experimental setup: by doing this we reduce the final translation scores, but we are able to assess performance for situations where this information is not available (e.g. the processing of oral dialects).

The main finding of our AST experiments is the overall under-performance of off-the-shelf wav2vec 2.0 large models in low-resource settings, even after fine-tuning them on target data. We also notice that the wav2vec 2.0 base models we trained with considerable less data were more effective in these same settings. We believe this happens because these task-specific SSL models better inform downstream tasks such as AST, since there is no domain shift in the data representation. This hints that massive multi-purpose wav2vec 2.0 models might not be the adequate solution for low-resource speech-to-text approaches, and that instead, smaller and better informed SSL blocks, trained on target data and/or domain, should be favored.

V. Final Remarks

This paper presented some premises and limitations of wav2vec 2.0 models. These are promising because they allow us to build ASR and AST models with less data than the previous approaches: this is an important issue since in-domain manually labeled data is very rare. This offers a new mean of action in order to address new languages.

Regarding our ASR experiments, we have seen that in order to improve recognition of accented speech, it is necessary to include accented speech in the fine-tuning data. Fine-tuning on both accented and non-accented speech seems to be a promising method for building general-purpose systems. In this work we used an equal amount of accented and native speech in our mixed dataset. The variation of the amount of accented speech in the fine-tuning dataset, the inclusion of additional accents (Swiss French, Quebec French), and the use of data augmentation to increase artificially the amount of accented speech are left as future work.

Regarding our AST experiments, we illustrated how building speech translation models without the existence of transcriptions is a challenging topic, and that models based on off-the-shelf wav2vec 2.0 models fail performance-wise in low-resource settings. Future research will focus on increasing performance in challenging settings: techniques such as speech augmentation, the production of *dummy transcriptions* [21], and multilingual pre-training and adaptation are promising topics.

The LIA will continue to study these approaches with the goal of making them highly accurate for real-world data.

Acknowledgment

This work used HPC resources from GENCI-IDRIS (grants 2020-A0111012991, 2021-AD011013317 and 2021-AD011013331). It was also funded by the European Commission through the SELMA project under grant number 957017.

References

[1] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," *arXiv preprint arXiv:1906.00910*, 2019.

[2] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202

[5] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *arXiv preprint arXiv:1904.03240*, 2019.

[6] Y.-A. Chung and J. Glass, "Improved speech representations with multi-target autoregressive predictive coding," *arXiv preprint arXiv:2004.05274*, 2020.

[7] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[8] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," *arXiv preprint arXiv:1911.03912*, 2019.

[9] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[10] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3497–3501.

[11] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. v. d. Oord, "Learning robust and multilingual speech representations," *arXiv preprint arXiv:2001.11128*, 2020.

[12] M. Riviere, A. Joulin, P.-E. Mazaré, and E. Dupoux, "Unsupervised pretraining transfers well across languages," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7414–7418.

[13] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pino, A. Baevski, A. Conneau, and M. Auli, "Xls-r: Self-supervised cross-lingual speech representation learning at scale," *arXiv*, vol. abs/2111.09296, 2021.

[14] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[15] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *International Conference on Learning Representations (ICLR)*, 2020.

[16] W. Xiong, J. Droppo, X. Huang, F. Seide, M. L. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Toward human parity in conversational speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 12, pp. 2410–2423, 2017.

[17] Q.-S. Zhu, J. Zhang, Z.-Q. Zhang, M.-H. Wu, X. Fang, and L.-R. Dai, "A noise-robust self-supervised pre-training model based speech representation learning for automatic speech recognition," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3174–3178.

[18] T. Fukuda, R. Fernandez, A. Rosenberg, S. Thomas, B. Ramabhadran, A. Sorin, and G. Kurata, "Data Augmentation Improves Recognition of Foreign Accented Speech," in *Proc. Interspeech 2018*, 2018, pp. 2409–2413.

[19] S. Evain, H. Nguyen, H. Le, M. Z. Boito, S. Mdhaffar, S. Alisamir, Z. Tong, N. Tomashenko, M. Dinarelli, T. Parcollet, A. Allauzen, Y. Estève, B. Lecouteux, F. Portet, S. Rossato, F. Ringeval, D. Schwab, and L. Besacier, " *LeBenchmark*: A Reproducible Framework for Assessing Self-Supervised Representation Learning from Speech," in *Interspeech*, 2021, pp. 1439–1443.

[20] S. Evain, H. Nguyen, H. Le, M. Z. Boito, S. Mdhaffar, S. Alisamir, Z. Tong, N. Tomashenko, M. Dinarelli, T. Parcollet *et al.*, "Task agnostic and task specific self-supervised learning from speech with *LeBenchmark*," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[21] M. Z. Boito, J. Ortega, H. Riguidel, A. Laurent, L. Barrault, F. Bougares, F. Chaabani, H. Nguyen, F. Barbier, S. Gahbiche, and Y. Estève, "ON-TRAC consortium systems for the IWSLT 2022 dialect and low-resource speech translation tasks," in *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*. Dublin, Ireland (in-person and online): Association for Computational Linguistics, May 2022, pp. 308–318. [Online]. Available: https://aclanthology.org/2022.iwslt-1.28

[22] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint arXiv:2006.13979*, 2020.

[23] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[24] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "SpeechBrain: A general-purpose speech toolkit," 2021, arXiv:2106.04624.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[26] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *ArXiv*, vol. abs/1212.5701, 2012.

[27] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, 2020, pp. 4211–4215.

[28] "African accented french," https://www.openslr.org/57/, accessed : 22-06-2022.

[29] S. Bird, "Bootstrapping the language archive: New prospects for natural language processing in preserving linguistic heritage," *Linguistic Issues in Language Technology*, vol. 6, no. 4, 2011.

[30] A. Berard, O. Pietquin, C. Servan, and L. Besacier, "Listen and translate: A proof of concept for end-to-end speech-to-text translation," *CoRR*, vol. abs/1612.01744, 2016.

[31] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen, "Sequence-to-Sequence Models Can Directly Translate Foreign Speech," in *Proc. Interspeech 2017*, 2017, pp. 2625–2629.

[32] H. Le, J. Pino, C. Wang, J. Gu, D. Schwab, and L. Besacier, "Dual-decoder transformer for joint automatic speech recognition and multilingual speech translation," *arXiv preprint arXiv:2011.00747*, 2020.

[33] M. Sperber, H. Setiawan, C. Gollan, U. Nallasamy, and M. Paulik, "Consistent transcription and translation of speech," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 695–709, 2020.

[34] E. Salesky, M. Wiesner, J. Bremerman, R. Cattoni, M. Negri, M. Turchi, D. W. Oard, and M. Post, "Multilingual tedx corpus for speech recognition and translation," in *Proceedings of Interspeech*, 2021.

[35] A. Anastasopoulos, L. Barrault, L. Bentivogli, M. Z. Boito, O. Bojar, R. Cattoni, A. Currey, G. Dinu, K. Duh, M. Elbayad, Y. Estéve, M. Federico, C. Federmann, S. Gahbiche, H. Gong, R. Grundkiewicz, B. Haddow, B. Hsu, D. Javorský, V. Kloudová, S. M. Lakew, X. Ma, P. Mathur, P. McNamee, K. Murray, M. Nădejde, S. Nakamura, M. Negri, J. Niehues, X. Niu, J. Ortega, J. Pino, E. Salesky, J. Shi, S. Stüker, K. Sudoh, M. Turchi, Y. Virkar, A. Waibel, C. Wang, and S. Watanabe, "FINDINGS OF THE IWSLT 2022 EVALUATION CAMPAIGN," in *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*. Dublin, Ireland: Association for Computational Linguistics, 2022.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[37] M. Post, "A call for clarity in reporting BLEU scores," in *Conference on Machine Translation: Research Papers*. Belgium, Brussels: Association for Computational Linguistics, Oct. 2018, pp. 186–191. [Online]. Available: https://www.aclweb.org/anthology/W18-6319

[38] E. Ansari, A. Axelrod, N. Bach, O. Bojar, R. Cattoni, F. Dalvi, N. Durrani, M. Federico, C. Federmann, J. Gu *et al.*, "Findings of the iwslt 2020 evaluation campaign," in *Proceedings of the 17th International Conference on Spoken Language Translation*, 2020, pp. 1–34.

[39] A. Anastasopoulos, O. Bojar, J. Bremerman, R. Cattoni, M. Elbayad, M. Federico, X. Ma, S. Nakamura, M. Negri, J. Niehues, J. Pino, E. Salesky, S. Stüker, K. Sudoh, M. Turchi, A. Waibel, C. Wang, and M. Wiesner, "FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN," in *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*. Bangkok, Thailand (online): Association for Computational Linguistics, Aug. 2021, pp. 1–29. [Online]. Available: https://aclanthology.org/2021.iwslt-1.1

[40] M. Z. Boito, F. Bougares, F. Barbier, S. Gahbiche, L. Barrault, M. Rouvier, and Y. Estéve, "Speech resources in the tamasheq language," *Language Resources and Evaluation Conference (LREC)*, 2022.

[41] C. Wang, Y. Tang, X. Ma, A. Wu, D. Okhonko, and J. Pino, "fairseq s2t: Fast speech-to-text modeling with fairseq," *arXiv preprint arXiv:2010.05171*, 2020.

[42] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[43] D. Berrebbi, J. Shi, B. Yan, O. Lopez-Francisco, J. D. Amith, and S. Watanabe, "Combining spectral and self-supervised features for low resource speech recognition and translation," *arXiv preprint arXiv:2204.02470*, 2022.

[44] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," *arXiv preprint arXiv:2107.04734*, 2021.

Self-Supervised Pretraining on Satellite Imagery: A Case Study on Label-Efficient Vehicle Detection

1st Jules Bourcier
Preligens (ex-Earthcube)
Inria, Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK
jules.bourcier@preligens.com

2nd Thomas Floquet
Preligens (ex-Earthcube)
MINES Paris - PSL University
thomas.floquet@minesparis.psl.eu

3rd Gohar Dashyan
Preligens (ex-Earthcube)
gohar.dashyan@preligens.com

4th Tugdual Ceillier
Preligens (ex-Earthcube)

5th Karteek Alahari
Inria, Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK

6th Jocelyn Chanussot
Inria, Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK

***Abstract*—In defense-related remote sensing applications, such as vehicle detection on satellite imagery, supervised learning requires a huge number of labeled examples to reach operational performances. Such data are challenging to obtain as it requires military experts, and some observables are intrinsically rare. This limited labeling capability, as well as the large number of unlabeled images available due to the growing number of sensors, make object detection on remote sensing imagery highly relevant for self-supervised learning. We study in-domain self-supervised representation learning for object detection on very high resolution optical satellite imagery, that is yet poorly explored. For the first time to our knowledge, we study the problem of label efficiency on this task. We use the large land use classification dataset Functional Map of the World to pretrain representations with an extension of the Momentum Contrast framework. We then investigate this model's transferability on a real-world task of fine-grained vehicle detection and classification on Preligens proprietary data, which is designed to be representative of an operational use case of strategic site surveillance. We show that our in-domain self-supervised learning model is competitive with ImageNet pretraining, and outperforms it in the low-label regime.**

***Index Terms*—deep learning, computer vision, remote sensing, self-supervised learning, object detection, land use classification, label-efficient learning**

I. Introduction

Very high resolution (VHR) satellite imagery is one of the key data from which geospatial intelligence can be gathered. It is an essential tool to detect and identify a wide range of objects, on very large areas and on a very frequent basis. Recently, we have seen the multiplication of available sensors, which has led to a large increase in the volume of data available. This makes it very challenging for human analysts to exploit these data without resorting to automatic solutions. Deep learning techniques today have been highly effective to perform such tasks. However, training those models requires very large labeled datasets. Annotating objects of interest in VHR images can prove to be very costly, being both difficult and time-consuming, and requiring fine domain expertise. In specific contexts such as in geospatial intelligence, the targets can be intrinsically rare, difficult to localize and to identify accurately. This makes the acquisition of thousands of examples impractical, as is typically required for classic supervised deep learning methods to generalize. Consequently, a major challenge is the development of label-efficient approaches, i.e., models that learn with few annotated examples.

To reduce the number of training samples for

difficult vision tasks such as object detection, transfer learning of pretrained neural networks is used extensively. The idea is to reuse a network trained *upstream* on a large, diverse source dataset. ImageNet [19] has become the de facto standard for pretraining: due to its large-scale and genericity, ImageNet-pretrained models show to be adaptable beyond their source domain, including remote sensing imagery [17]. Nonetheless, the *domain gap* between ImageNet and remote sensing domains brings questions about the limitations of this transfer when there are very few samples on the task at hand, e.g., the detection of rare observables from satellite images. To fit the distributions of *downstream* tasks with maximum efficiency, one would ideally use generic *in-domain* representations, obtained by pretraining on large amounts of remote sensing data. This is infeasible in the remote sensing domain due to the difficulty of curating and labeling these data at the scale of ImageNet. However, imaging satellites provide an ever-growing amount of unlabeled data, which makes it highly relevant for learning visual representations in an unsupervised way.

Self-supervised learning (SSL) has recently emerged as an effective paradigm for learning representations on unlabeled data. It uses unlabeled data as a supervision signal, by solving a *pretext task* on these input data, in order to learn semantic representations. A model trained in a self-supervised fashion can then be transferred using the same methods as a network pretrained via a upstream supervised task. In the last two years, SSL has shown impressive results that closed the gap or even outperformed supervised learning for multiple benchmarks [3], [4], [8], [9]. Recently, SSL has been applied in the remote sensing domain to exploit readily-available unlabeled data, and was shown to reduce or even close the gap with transfer from ImageNet [1], [16], [26]. Nonetheless, the capacity of these methods to generalize from few labels has not been explored on the important problem of object detection in VHR satellite images.

In this paper, we explore in-domain self-supervised representation learning for the task of object detection on VHR optical satellite imagery. We use the large land use classification dataset Functional Map of the World (fMoW) [6] to pretrain representations using the unsupervised framework of MoCo [9]. We then investigate the transferability on a difficult real-world task of fine-grained vehicle detection on proprietary data, which is designed to be representative of an operational use case of strategic site surveillance. Our contributions are:

- We apply a method based on MoCo with temporal positives [1] to learn self-supervised representations of remote sensing images, that we improve using (i) additional augmentations for rotational invariance; (ii) a fixed loss function that removes the false temporal negatives in the learning process.
- We investigate the benefit of in-domain self-supervised pretraining as a function of the annotation effort, using different budgets of annotated instances for detecting vehicles.
- We show that our method is better than or at least competitive with supervised ImageNet pretraining, despite using no upstream labels and 3× less upstream data.

Furthermore, our in-domain SSL model is more label-efficient than ImageNet: when using very limited annotations budgets ($\simeq$20 images totalling $\simeq$12k observables), we outperform ImageNet pretraining by 4 points AP on vehicle detection and 0.5 point mAP on joint detection and classification.

II. Related work

A. Self-supervised representation learning

SSL methods use unlabeled data to learn representations that are transferable to downstream tasks (e.g. image classification or object detection) for which annotated data samples are insufficient. In recent years, these methods have been successfully applied to computer vision with impressive results that closed the gap or even outperformed supervised representation learning on ImageNet, on multiple benchmarks including classification, segmentation, and object detection [3], [4], [8], [9]. They commonly rely on a pretraining phase, where a neural network, a *representation encoder*, is trained to solve a pretext task, for which generating labels does not require any effort or human involvement. Solving the pretext task is done only for the true

purpose of learning good data representations that allow for efficient training on a downstream task of genuine interest.

B. Contrastive learning

Contrastive learning has recently become the most competitive unsupervised representation learning framework, with approaches such as MoCo [9], SimCLR [4], and SwAV [3]. Contrastive methods work by attracting embeddings of pairs of samples known to be semantically similar (*positive pairs*) while simultaneously repelling pairs of unlike samples (*negative pairs*). The most common way to define similarity is to use the *instance discrimination* pretext task [7], [24], in which positives are generated as random data augmentations on the same image, and negatives are simply generated from different images. Thanks to this pretext task, the encoder learns similar representations for several views of the same object instance in an image and distant representations for different instances. *Momentum Contrast* (MoCo) [9] is a strong contrastive method that implements a dynamic dictionary with a queue and a moving-averaged encoder, which enables building a large and consistent dictionary on-the-fly (see section III-A for more details). In this paper, we adopt a recent *geography-aware* rework of this approach made by [1].

C. Representation learning in remote sensing

Building on that success in computer vision, SSL has recently been applied to remote sensing and was shown to reduce or even close the gap with transfer from ImageNet. [12] first made use of contrastive learning for remote sensing representation learning and [13] also apply a spatial augmentation criteria on top of MoCo [9]. These works exploit relevant prior knowledge about the remote sensing domain: the assumption that images that are geographically close should be semantically more similar than distant images. Another way of making the learning procedure *geography-aware* is to exploit the image time series that one can get from a given geographic area thanks to the frequent revisit of satellites. This approach is adopted by [1], that use spatially aligned images over time to construct *temporal positive pairs*. With their *geography-aware* representations learned on the fMoW dataset [6], they improve significantly on classification, segmentation and object detection downstream tasks. However, they do not study label efficiency. In this paper, we apply this method, called MoCoTP, to an operational use-case. We study the label efficiency and bring small extensions to this model, that further improve its performance. In the same vein, [16] present a pipeline for self-supervised pretraining on uncurated remote sensing data and propose a method that learns representations that are simultaneously variant and invariant to temporal changes. They significantly outperform ImageNet pretraining on classification from few labels on medium resolution images (10m). One can also exploit the multispectral and multisensor nature of remote sensing. [20] split multispectral images into two different subsets of channels and use them as augmented (positive) views. [21] extend this to multiple sensors, taking subsets of the combination of all bands. Regarding the domain of pretraining data in remote sensing, [17] show that the relatedness of the pretraining distribution to the downstream task can improve performances in low-labeled settings. However, they only study supervised pretraining and classification tasks, and show that transfer performance is very dependent on the labeling and data curation quality in the pretraining dataset. Therefore this leaves unresolved the problem of obtaining generic representations with less dependence on labels and this is where SSL can help.

III. Method

In this section, we detail our approach to explore the applicability of SSL to vehicle detection and classification on optical satellite imagery. The overall procedure is described in Fig. 1. We first pretrain a ResNet-50 backbone on the fMoW dataset [6] in an unsupervised way with a recent SSL method, MoCoTP [1]. See section III-A for details on the approach and section IV-A2 for implementation details.

We use the weights of this pretrained backbone on two downstream tasks: (i) fMoW image recog-

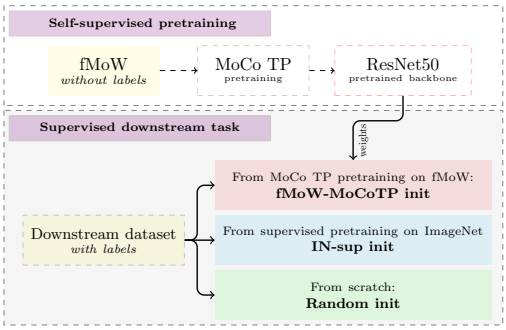


Fig. 1. Schematic outline of our method. The top block represents the pretraining phase. The bottom block represents the pretrained weights that are injected into the downstream task model.

nition task: we inject the weights in a classifier, and perform *linear probing* and *finetuning*. See section IV-A for more details. (ii) Vehicle detection and classification on Preligens proprietary data: we inject the weights in a RetinaNet detector [15], and finetune the entire model. See section IV-B2 for implementation details.

A. Self-supervised learning with MoCo and Temporal Positives

We employ the MoCo [9] framework for contrastive SSL. The base method we use is from the improved variant MoCo-V2 [5]. MoCo learns to match an input *query* q to a *key* k^+ (representing the encoded views of the same sample) among a set of negative keys k^-, using the instance discrimination pretext task [24]. It uses a deep encoder (e.g. a ResNet [10]) to map input image queries and keys to a vector representation space. Negative keys are extracted with a moving average network (momentum encoder) to maintain consistent representations during training, and are drawn from a memory queue. We refer readers to [9] for details on this. MoCo uses the popular choice of InfoNCE [18] for the contrastive loss:

$$\mathcal{L}(q, k^+) = -\log \frac{e^{(q \cdot k^+/\tau)}}{e^{(q \cdot k^+/\tau)} + \sum_{k^-} e^{(q \cdot k^-/\tau)}} \quad (1)$$

where τ is a temperature scaling parameter.

On top of MoCo, we adopt the extension to temporal views proposed in [1], MoCo with Temporal Positives (MoCoTP). It extends the instance discrimination pretext task to use spatially aligned images from different times as positives. Maximizing similarity between temporal views can provide richer semantic information that extracts persistent scene features over time [1]. The same random augmentations as in MoCo-V2 are also applied on the temporal samples.

Improvements to MoCoTP. Compared to MoCoTP, we adopt the following modifications from [2], to improve the framework of [1]: (i) In addition to the geometric and color perturbations of MoCo-V2, we apply random vertical flips and rotations by multiples of 90°. Since the data augmentation scheme plays a leading role in contrastive learning [22], we aim to learn representations more suited to overhead images thanks to rotational invariance. (ii) [1] introduces temporal positives as a drop-in replacement for q and k^+ in (1). However, this can introduce *false negatives*. Indeed, at each iteration of training, it may happen that the set of negatives k^- contains temporal views for samples of the current mini-batch of queries. Such false negatives will cause an incorrect repulsion between the embeddings of similar samples. To avoid the false negatives to interfere with the learning objective, we simply mask out the logits $q \cdot k^-$ in the InfoNCE loss in (1) for every k^- that happens to be a temporal view of q.

After being trained on the pretext task for a given number of iterations, the query encoder is extracted and can be transferred to downstream tasks.

B. Transfer to dowstream task

Following the MoCoTP pretraining on fMoW, we transfer the obtained weights on two downstream tasks: fMoW image recognition task and a real-world use-case, vehicle detection and classification on Preligens proprietary data. We refer to the downstream training initialized with SSL weights learned on the fMoW dataset as **fMoW-MoCoTP init**. For each downstream task, we compare fMoW-MoCoTP init with two baselines:

- **IN-sup init**: the backbone has been pretrained on ImageNet in a supervised way;

- **Random init**: the backbone is initialized randomly (i.e., no pretraining).

IV. Experimental setup

A. Pretraining and evaluation on fMoW

1) Dataset: For the sake of learning semantic representations in remote sensing, we adopt the fMoW dataset [6], following [1]. fMoW is a public dataset of VHR imagery from Maxar Earth observation satellites, is large-scale with 363,571 training images, and covers 207 countries. It provides images from same locations over time. We apply the self-supervised MoCoTP method for pretraining on fMoW training set using these available temporal views. fMoW also includes ground-truths labels for functional land use classification with 62 diverse categories with a long-tailed distribution. We do not use those labels for self-supervised pretraining, but use them downstream to evaluate representations learned directly for the classification of the images seen during pretraining. Following [6] and [1], we use the fMoW-RGB products for our experiments, which provides 3-bands imagery at 0.5m ground resolution. Preprocessing is applied identically to [6] to resize input images to 224×224 pixels.

Ensuing the pretraining stage, the model is evaluated on the land use classification task with the two protocols of linear probing (training a linear classifier on top of frozen features from the pretrained encoder) or finetuning (updating all parameters of the network). To study the label efficiency of the learned representations, supervised evaluation is performed on 1%, 10%, and 100% of the labeled training data. For 1% and 10% labels, we subsample by preserving the base class distribution. The evaluation metric is the F1-score averaged over classes. Testing is performed on the fMoW validation set for comparison with [1], which consists of 53,041 images.

2) Implementation details: In all our experiments, we use MoCoTP with ResNet-50 for the query and key encoders. We utilize the same hyperparameters as in [2]. For self-supervised pretraining, we use a learning rate of 3e-2 with a cosine schedule, batch size of 256, dictionary queue size of 65536, temperature scaling of 0.2, SGD optimizer with a momentum of 0.9, weight decay of 1e-4. Augmentations are the same as in MoCo-V2, plus a vertical flip with 0.5 probability and a 90° rotation with 0.75 probability. Pretraining lasts 200 epochs. For linear probing, we use a learning rate of 1, no weight decay, and only random resized cropping for the augmentations. For finetuning, we use a learning rate of 3e-4 for ResNet weights and 1 for the final classification layer, weight decay of 1e-4, and the same augmentations used for pretraining. We compare self-supervised pretraining against Random init and IN-sup init, under the different label regimes. Models are trained with cross-entropy loss and evaluated on epoch with the highest top-1 accuracy on the validation set.

B. Transfer to vehicle instance detection

1) Dataset: We describe here the Preligens proprietary datasets used for the transfer to object detection. Statistics of our datasets are reported in Table I.

We call 'S' our base dataset. It consists of 204 Maxar WorldView-3 satellite images at 0.3m resolution, and approximately 150k vehicles. To study the label efficiency, we subsample this base dataset S into smaller datasets, 'XS' and 'XXS', targeting respectively 50% and 10% of the observables present in S. We ensure that XXS is included in XS, so that our datasets follow a "Matriochka" structure, which simulates the incremental nature of annotation efforts. The sampling strategy is such that the class distribution is well preserved. To perform variance experiments on our results, we run the sampling thrice and get three different variants of the XS and XXS datasets. We proceed similarly for the S training set by selecting other satellite images that match the class distribution of the initial S dataset. We keep the same validation and testing sets throughout the experiments, and make sure that the geographical sites of the images are distinct between train and test splits. The training and validation raster images are divided into tiles of 512x512 pixels with an overlap of 128 pixels. We take all positive tiles (i.e., tiles containing at least one instance) and some negative tiles, with a positive/negative ratio of 5 and 3 for training and

validation sets respectively: this setting allows to focus training efforts on positive tiles while keeping a fair amount of negative tiles. The ground truth labels are non-oriented bounding boxes of target observables with their class label. Our classification problem is composed of 8 vehicle categories: *Civilian*, *Military*, *Armored*, *Launcher*, *Ground Support Equipment* (*GSE*), *Electronics*, *Heavy Equipment* (*HE*), and *Lifting Equipment* (*LE*). The classes are naturally heavily imbalanced: the first three classes are very dominant, covering $\sim$96.5% of the vehicles in our datasets, while the last five classes are very under-represented.

2) Implementation details:

The detection model. The object detection model used is a RetinaNet [15], with a ResNet-50-FPN backbone [14] with pyramid levels P_2 to P_6. The backbone is initialized with the pretrained weights, while the detector specific modules are initialized randomly. We finetune the RetinaNet model end-to-end, and use the focal loss objective for the classification [15].

Hyperparameters. We select a learning rate of 1e-4, an Adam optimizer and a batch size of 8, and no weight decay. As data augmentations, we use 90° rotations and flips, as well as CLAHE. We train until convergence and reduce the learning rate by 2 on plateau of the validation loss. Then, the epoch selected for the evaluation of the model is the one achieving the best F1-score calculated on the validation set, with a fixed detection score threshold of 0.15.

Evaluation. The F1-score is computed from the precision-recall curves obtained by varying the detection threshold from 0.15 to 0.9. In the following, we refer to the results on the task of vehicle detection (regardless of the class) as *level-1* results, and on the task of joint detection and classification as *level-2* results. The IoU threshold matching the predictions with the ground truths is set to 0.0, which is operationally relevant for observable counting purposes. In addition to the level-1 F1-score, we also compute the level-1 average precision (AP) and level-2 mean average precision (mAP) which are commonly used metrics to evaluate detection models.

V. Results

A. fMoW classification

Table II shows the results of linear probing and finetuning on the 62-class land use classification task of fMoW. With 100% labels, we see that our improved reproduction of MoCoTP increases performance by 4.36 pts compared to [1] in linear probing and 1.62 pts in finetuning. This shows that the use of the rotation augmentations and the correction of false negatives in the loss function is helpful, especially for linear probing, which closes the gap with finetuning completely. As a note, the sole additional augmentations also improve the baselines Random init and IN-sup init of [1] by 1.29 pts and 0.68 pts respectively. Moreover, MoCoTP shows impressive label efficiency: in the semi-supervised settings of 1% and 10% labels, we see that it gives respectively 96% and 87% of the performance of the network trained with 100% labels. Also, it surpasses IN-sup init by large margins, with e.g. +20.57 pts on 1% finetune. These results indicate that MoCoTP is very efficient at learning semantic features from the upstream dataset. Therefore, this is encouraging in order to transfer to a downstream operational task where labeled data are scarce.

B. Transfer to vehicle detection

1) Label efficiency: Table III and Fig. 2 show the results on vehicle detection. The F1-score with fMoW-MoCoTP init is always higher than with IN-sup init or Random init. fMoW-MoCoTP init achieves an F1-score of 65.1% with only 12k observables on dataset XXS. On dataset XS, with 50% less examples than on dataset S, fMoW-MoCoTP init is only 3.8 pts below the score obtained on dataset S. Moreover, the smaller the dataset, the larger the gap between fMoW-MoCoTP init and IN-sup init or Random init: on dataset S, fMoW-MoCoTP init's F1-score is 5.20 pts better than Random init on average, and also 0.40 pts better than IN-sup init, whereas on the XXS dataset, fMoW-MoCoTP init's F1-score is 39 pts better than Random init, and 3.7 pts better than IN-sup init. These results show that self-supervised in-domain pretraining can be competitive with supervised pretraining on ImageNet, and even give better results

TABLE I

DATA STATISTICS FOR VEHICLE TRAINING, VALIDATION AND TESTING SETS. FOR EACH OF THE XXS, XS AND S TRAINING SETS, THE REPORTED NUMBERS ARE THE MEAN NUMBER OF OBSERVABLES BETWEEN THE DIFFERENT SAMPLED SETS.

Dataset	Images	Pos. tiles	Neg. tiles	Vehicles	Civilian	Military	Armored	GSE	Launcher	Electronics	HE	LE
XXS	19	597	113	11,833	6,668	3,215	1,516	154	158	64	33	23
XS	108	3,152	599	58,189	32,937	14,719	8,466	571	732	385	237	139
S	204	6,438	1,231	115,617	66,504	29,332	16,148	820	1364	698	432	319
Val	63	2,526	4,178	53,204	31,339	11,919	8,464	607	412	270	130	101
Test	88	–	–	32,550	19,923	7,542	3,872	361	334	237	184	97

TABLE II

RESULTS ON FMOW CLASSIFICATION (F1-SCORE IN %). FOR 1% AND 10% LABELS, THE VALUES ARE 'MEAN (SD)' ACROSS 3 REPLICATES WITH VARYING TRAINING SAMPLES. * DENOTE OUR IMPROVED REPRODUCTIONS AS DETAILED IN SECTION III-A

	1% labels		10% labels		100% labels	
	Linear	Finetune	Linear	Finetune	Linear	Finetune
Random init [1]	–	–	–	–	–	64.71
IN-sup init [1]	–	–	–	–	–	64.72
fMoW-MoCoTP init [1]	–	–	–	–	64.53	67.34
Random init *	–	19.29 (1.65)	–	51.87 (0.50)	–	65.39
IN-sup init *	32.41 (0.17)	39.43 (1.53)	43.86 (0.07)	57.32 (0.07)	50.25	66.01
fMoW-MoCoTP init *	**60.05 (0.11)**	**60.00 (0.43)**	**66.15 (0.11)**	**66.35 (0.75)**	**68.89**	**68.96**

TABLE III

RESULTS OF EACH METHOD ON EACH DATASET FOR VEHICLE DETECTION (%). THE VALUES ARE 'MEAN (SD)' ACROSS 3 REPLICATES WITH VARYING TRAINING SAMPLES.

Metric	F1			AP			mAP		
Training set	XXS	XS	S	XXS	XS	S	XXS	XS	S
Random init	26.0 (1.9)	55.1 (2.2)	74.7 (1.4)	12.3 (1.9)	46.9 (2.4)	75.3 (3.4)	2.2 (0.3)	8.4 (0.6)	16.3 (0.5)
IN-sup init	61.4 (0.5)	75.1 (0.8)	**79.5 (0.6)**	56.1 (0.9)	75.6 (0.5)	**80.9 (0.9)**	9.3 (0.4)	**14.9 (0.6)**	**19.7 (1.1)**
fMoW-MoCoTP init	**65.1 (0.8)**	**76.1 (0.4)**	**79.9 (0.3)**	**60.1 (1.6)**	**77.3 (0.1)**	**81.6 (0.5)**	**9.7 (0.2)**	**14.4 (0.4)**	**19.3 (1.0)**

in low-label regimes. Moreover, Fig. 3 shows an example of predictions for models finetuned on an XS dataset, that illustrates qualitative improvements of fMoW-MoCoTP against IN-sup init on both false positive and false negative detections.

2) Dominant vs. rare classes: Table IV shows the APs in level-2. fMoW-MoCoTP init achieves significantly higher results than Random init on these six classes. Also, fMoW-MoCoTP init achieves higher AP than IN-sup init on the three dominant classes (*Civilian*, *Military* and *Armored*), that account for ∼96.5% of the vehicles in our datasets. However, IN-sup init outperforms fMoW-MoCoTP init on the very rare *Launcher*, *Electronics* and *HE* classes, that cover ∼2.2% of the vehicles in our datasets. Since mAP gives equal importance to all classes, this translates into much closer values for mAP scores than level-1 AP scores between fMoW-MoCoTP init and IN-sup init methods, as one can see in Fig. 2. This might suggest that fMoW-MoCoTP init is *lazier* and mainly focuses on the dominant classes. The fMoW dataset used for pretraining contains a long-tailed distribution of semantic categories. One could hypothesize that this leads to representations being more skewed towards over-represented visual concepts than ImageNet, as the latter contains a balanced set of categories, and that such bias may also negatively impact the transfer to under-represented classes downstream. However, further work is needed to provide ground

TABLE IV

AP PER CLASS (%). RESULTS ON GSE AND LE ARE OMITTED BECAUSE THEY ARE CLOSE TO ZERO DUE TO THE POOR NUMBER OF EXAMPLES IN THE DATASETS. THE VALUES ARE 'MEAN (SD)' ACROSS 3 REPLICATES WITH VARYING TRAINING SAMPLES.

Dominant classes	Civilian			Military			Armored		
Training set	XXS	XS	S	XXS	XS	S	XXS	XS	S
Random init	14.4 (1.8)	48.1 (1.2)	75.4 (4.5)	3.1 (1.5)	14.8 (0.6)	29.1 (3.2)	0.0 (0.0)	3.9 (3.5)	14.9 (7.7)
IN-sup init	54.2 (1.0)	75.4 (0.8)	81.9 (0.7)	**17.2 (3.8)**	**25.0 (2.1)**	32.8 (1.9)	**0.8 (0.4)**	**5.7 (0.8)**	**21.4 (11.4)**
fMoW-MoCoTP init	**59.9 (1.0)**	**79.5 (0.8)**	**83.6 (1.1)**	**17.9 (0.9)**	**25.0 (1.4)**	**35.4 (3.0)**	0.2 (0.1)	**6.5 (1.6)**	**22.4 (11.3)**
Rare classes	**Launcher**			**Electronics**			**HE**		
Training set	XXS	XS	S	XXS	XS	S	XXS	XS	S
Random init	0.0 (0.0)	0.4 (0.3)	6.0 (2.8)	0.0 (0.0)	0.0 (0.0)	0.1 (0.1)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
IN-sup init	**0.3 (0.2)**	**9.4 (2.4)**	**18.8 (4.7)**	0.0 (0.0)	**1.5 (0.8)**	**5.0 (0.2)**	0.0 (0.0)	**0.3 (0.5)**	**1.1 (1.0)**
fMoW-MoCoTP init	0.1 (0.1)	4.1 (1.1)	13.0 (5.7)	0.0 (0.0)	0.4 (0.4)	1.6 (0.4)	0.0 (0.0)	0.0 (0.0)	0.0 (0.1)

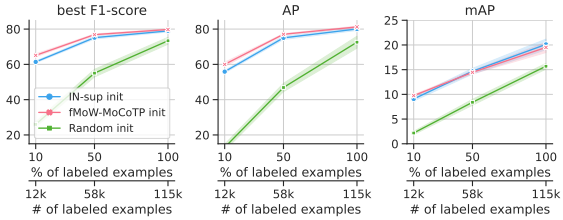


Fig. 2. Graphic view of the results of Table III.

Fig. 3. Example of cherry-picked predictions. Here the models have been finetuned on an XS dataset. White regions highlights errors specific to the top model, in solid and dotted style for false negatives and false positives respectively.

for this hypothesis.

VI. CONCLUSION

In this work, we explored the added value of in-domain SSL for a real-world defense-related remote sensing application: vehicle detection and classification on VHR optical satellite imagery. Considering this downstream task, we compared in-domain pretraining on the fMoW dataset with the traditional supervised ImageNet pretraining. We showed that self-supervised pretraining on fMoW is either competitive with or better than supervised ImageNet pretraining, despite using no upstream labels and 3× less upstream data. To study label efficiency, experiments were performed for different downstream dataset sizes, thereby mimicking different annotation budgets. We showed that in-domain SSL pretraining leads to better label efficiency than supervised pretraining on ImageNet. This result is particularly suitable for defense industry use cases, where labeling data is challenging. Further work could include additional studies such as increasing the amount of in-domain pretraining data, using a vehicle dataset that is more balanced in terms of classes (or deprived of its dominant classes), extending the range of sizes for the downstream dataset, and experimenting other SSL methods, including methods designed specifically for dense downstream tasks like detection [11], [23], [25].

Acknowledgments

This work was performed using HPC resources from GENCI–IDRIS (Grant 2021-AD011013097). Karteek Alahari was supported in part by the ANR grant AVENUE (ANR-18-CE23-0011).

REFERENCES

[1] K. Ayush, B. Uzkent, C. Meng, K. Tanmay, M. Burke, D. Lobell, and S. Ermon, "Geography-aware self-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 181–10 190.

[2] J. Bourcier, G. Dashyan, J. Chanussot, and K. Alahari, "Evaluating the label efficiency of contrastive self-supervised learning for multi-resolution satellite imagery," in *Image and Signal Processing for Remote Sensing XXVIII*, vol. 12267. SPIE, 2022, p. 122670K.

[3] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9912–9924, 2020.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[5] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[6] G. Christie, N. Fendley, J. Wilson, and R. Mukherjee, "Functional map of the world," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6172–6180.

[7] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[8] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 271–21 284, 2020.

[9] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] O. J. Hénaff, S. Koppula, J.-B. Alayrac, A. van den Oord, O. Vinyals, and J. Carreira, "Efficient visual pretraining with contrastive detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 086–10 096.

[12] N. Jean, S. Wang, A. Samar, G. Azzari, D. Lobell, and S. Ermon, "Tile2vec: Unsupervised representation learning for spatially distributed data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3967–3974.

[13] J. Kang, R. Fernandez-Beltran, P. Duan, S. Liu, and A. J. Plaza, "Deep unsupervised embedding for remotely sensed images based on spatially augmented momentum contrast," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 3, pp. 2598–2610, 2021.

[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[16] O. Mañas, A. Lacoste, X. Giro-i Nieto, D. Vazquez, and P. Rodriguez, "Seasonal contrast: Unsupervised pretraining from uncurated remote sensing data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9414–9423.

[17] M. Neumann, A. S. Pinto, X. Zhai, and N. Houlsby, "In-domain representation learning for remote sensing," in *AI for Earth Sciences Workshop at International Conference on Learning Representations (ICLR)*, 2020, pp. 1–20.

[18] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[20] V. Stojnic and V. Risojevic, "Self-supervised learning of remote sensing scene representations using contrastive multi-view coding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1182–1191.

[21] A. M. Swope, X. H. Rudelis, and K. T. Story, "Representation learning for remote sensing: An unsupervised sensor fusion approach," *arXiv preprint arXiv:2108.05094*, 2021.

[22] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 6827–6839, 2020.

[23] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li, "Dense contrastive learning for self-supervised visual pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3024–3033.

[24] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3733–3742.

[25] Z. Xie, Y. Lin, Z. Zhang, Y. Cao, S. Lin, and H. Hu, "Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 684–16 693.

[26] X. Zheng, B. Kellenberger, R. Gong, I. Hajnsek, and D. Tuia, "Self-supervised pretraining and controlled augmentation improve rare wildlife recognition in uav images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 732–741.

Anomaly detection in embedding space learned with a contrastive approach

Corentin Lamboley
IA3D
DGA-MI
Rennes, France

Tristan Bitard-Feildel *
IA3D
DGA-MI
Rennes, France

Amélie Chérubin *
IA3D
DGA-MI
Rennes, France

***Abstract*—Confidence in data is critical to learn safe and reliable AI models. In this paper, we explore the ability of embedding space learned through contrastive training to capture outliers and labeling errors. Ideally, embedding space learned with a contrastive approach should enforce proximity of data points in embedding space. This property of the embedding should facilitate the application of anomaly detection methods to detect erroneous data points. The study focus on the CIFAR-10 dataset. Once the embedding learned, we evaluated the ability of the anomaly detection methods to identify correctly erroneous data points using controlled noise through label flipping. We tested several anomaly detection methods with different hyper parameters. The data separation per class are easily observed in the embedding space and class outliers can be identified, highlighting the presence of data point outside the domain distribution of each class. The embedding space is also resilient to noisy data, and the tested anomaly detection methods can capture data points not corresponding to the classes used to learn the embedding. This preliminary work shows the potential growth of embedding space learned with unsupervised method to capture outliers and preprocess data.**

Keywords—data confidence, anomaly detection, self supervised learning, contrastive loss

I. Introduction

When performing a classification task, the maximum achievable accuracy depends on the quality and the quantity of data and on the appropriateness of the chosen learning algorithm [4]. Large datasets have enabled deep neural networks to achieve remarkable success in various domains, but this success is highly dependent on the quality of the training labels, which can be extremely very to create manually as datasets grow, especially when domain expertise is required.

Consequently, substitution labeling techniques have been developed such as crowdsourcing [1], model-assisted labeling, active-learning [2] and weak supervision [3]. Nevertheless, these methods inherently generate noisy labels decreasing the performance of deep neural networks, since they have high capacities and tend to overfit to these noisy labels, resulting in poor generalization performance.

To tackle this, methods have been developed to identify mislabeled data [4, 5] or to be robust to label noise [6, 7, 8, 9]. In particular, self-supervised contrastive learning [10] demonstrated great results in the past few years, and turned out to achieve good performance even in the presence of label noise [11]. Contrastive learning discards all the labels and learns representations by maximizing similarity between differently augmented views of the same data point via a contrastive loss in the latent space. A simple fully connected network can be trained on these representations in a supervised way, given the (potentially noisy) labels, and shows label noise robustness.

In this work, we present methods for outliers and errors detection, as well as learning with noisy labels techniques using representations from contrastive algorithms. Our contributions are two folds.

As a preliminary step, we evaluated the quality of learned representations with a Contrastive Learning approach, and their ability to provide label noise robustness. Second, we evaluated several anomaly detection methods on the learned representations. Anomaly detection algorithms are highly sensitive to the intrinsic data structure, and finding an appropriate method working with learned representations is a challenging task. Such, we investigated the relationship between incorporating noise and performances of anomaly detection algorithms.

II. Preliminaries

A. Contrastive Learning

The goal of contrastive learning methods is to learn an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart. A positive pair consists of two different points of view of the same example, obtained using data augmentation (e.g. Random cropping or Color distortion for image data).

Given this new dataset of augmented data, an encoder extracts features and creates an embedding for each example of each pair. The ResNet model is usually the choice for the image encoder, and the final goal of the contrastive approach is to find the correct weights for the encoder to match similar pairs together. The encoder creates representations, and in practice, we add a Projection Head to map the representations h_i to a lower dimensional vector z_i before calculating the contrastive loss.

This loss function tends to maximize the similarity between embedding for positive pairs and minimize it for negative pairs. A classical choice is the Noise Contrastive Estimation (NCE) defined as [24]:

$$l_{i,j} = -log \frac{\exp\left(\frac{z_i^T z_j}{\tau}\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp\left(\frac{z_i^T z_k}{\tau}\right)} \quad (1)$$

for the positive pair of augmented samples (i, j), where z_i is the output of the projection head of the augmented sample i, $\mathbb{1}_{[k \neq i]}$ is the indicator function and τ the temperature parameter. We compute the final loss across all positives pairs. Fig. 1 provides an example of the full process of this algorithm.

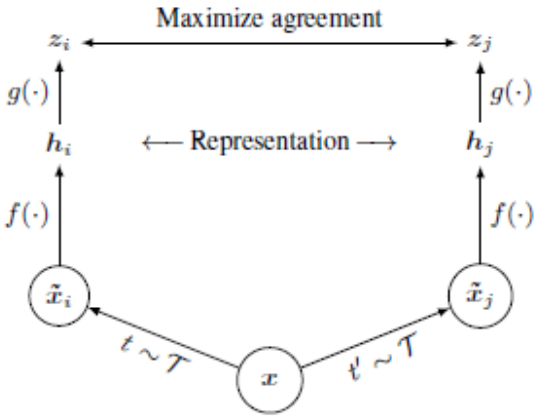


Fig. 1. Simple representation of the contrastive learning algorithm. Two different data augmentations (t and t' from the same family of augmentations) are applied to each data example to obtain two correlated views. The encoder $f(.)$ and the projection head $g(.)$ are trained to minimize the contrastive loss. After the training is completed, we discard the projection head and keep the encoder and representations h for downstream tasks.

B. *Label noise generation*

For our experiments, we use the CIFAR10 dataset and follow the same protocol as [12]. For label noise, we work with asymmetric label noise, where noisy labels are obtained by randomly flipping the true labels. This mimics real-world noise.

Let y^* be the unknown, true, uncorrupted label and $\tilde{y}$ the observed, noisy label. We generate noisy data from clean data by randomly switching some labels of training examples to different classes non-uniformly according to a randomly generated noise transition matrix $Q_{\tilde{y}|y^*} \coloneqq p(\tilde{y} = i|y^* = j)$, $\forall i, j \in [m]$ the set of unique class labels. Noise transition matrices have different traces for different noise levels, and have different sparsities. See Appendix A for the noise matrices used in our experiments.

Definition (sparsity). A statistic defined by the fraction of zeros in the off-diagonals of $Q_{\tilde{y}|y^*}$. A high sparsity quantifies non-uniformity of label noise, common to real-world datasets. Zero sparsity means that every noise rate in $Q_{\tilde{y}|y^*}$ is non-zero, while a sparsity equal to one means that there is no noise because all off-diagonals of $Q_{\tilde{y}|y^*}$ are zeros.

C. *Outliers detection*

Outliers are items in a dataset that are rare and deviate from the general data distribution. It is as if they were generated by a different mechanism. The detection of outliers is an active area of research in the field of data mining, and has various practical applications such as credit card fraud detection, telecom or medical analysis. In particular, classification algorithms mislabeled instances can be considered outliers and be removed from the training set to improve the performance of the classifier.

There are a great number of different outlier detection algorithms, and we propose to use the following ones in our future experiments:

- Auto-Encoder: A fully connected neural network that uses the reconstruction error as the outlier score [23].
- Isolation Forest: An ensemble-based method that isolates anomalies using tree classifiers [15].
- Average/Median KNN: A simple proximity-based method using the average or median of the k nearest neighbors as the outlier score [21].
- LOF: "Local Outlier Factor" An algorithm also using the k nearest neighbors to estimate density. The samples with low densities are considered outliers. [17].
- CBLOF: The cluster-based version of LOF that can use any clustering algorithm [18].
- KDE: "Kernel Density Estimation" A well-known method using kernels with a particular bandwidth to estimate the probability density function [16].
- PCA: "Principal Component Analysis" A method for outliers' detection using a Principal Component Classifier [20].
- COPOD: "Copula-based Outlier Detection" that constructs an empirical copula to predict tail probabilities to determine the level of "extremeness" of a sample [19].
- ECOD: "Empirical Cumulative Outlier Detection" using the empirical cumulative distribution per dimension of the data to predict tail probabilities across dimensions [22].

III. Experiments

A. *Exploring label noise robustness and representation's quality*

In this experiment, we first demonstrate how Contrastive Learning improves label noise robustness, even when using sub-optimal hyperparameters to generate representations in the latent space. Moreover, we evaluate the algorithm's ability to separate samples into relevant clusters, and the side effects of generating representations in an unsupervised way.

B. *Finding labelling errors using anomaly detection*

If Contrastive Learning provides a certain robustness against label noise, then it also allows for the discovery of labeling errors in the latent space of representations.

First, a sample's representation created by the encoder tend to be close to similar samples representations and far from others, independently of the associated noisy label. Therefore, in the latent space, samples are globally regrouped into clusters corresponding to their true label.

This has the consequence for a mislabeled example to be far away from the cluster associated with its true label. To some extent, a labeling error can be seen as an outlier in the distribution of representations of its associated class. Given that, it is possible to use outliers' detection methods on the latent space to detect labeling errors. Section IV.C benchmarks the best outliers' detection methods described earlier for this purpose.

The first problematic in dealing with outlier detection algorithms is the way of measuring their performance. In most cases, these algorithms generate an anomaly or outlier score that does not correspond to a probability of being an outlier.

Hence, considering samples with a score higher than 0.5 as outliers may not work well.

Yet, without knowing the threshold to use for a particular method that separates well errors from clean samples, it is still possible to speculate how good this method can be at the task of separating the distribution of errors and clean samples. To do so, we will use AUC ROC scores.

The process to test out an outlier detection method is the following:

- The dataset is the representations of the 50000 CIFAR10 images generated with the encoder, with the labels flipped according to a noise matrix.
- Class by class, we use the algorithm to assign an outlier score to every sample belonging to this particular class, and the scores are normalized. A sample belongs to a class if its noisy labels correspond to this class. In theory, we consider that outliers are errors and should not belong to that particular class.
- Knowing both the true and noisy labels, we can calculate the AUC ROC of this algorithm for each class, and the global AUC ROC along all the samples.

Finally, we can compare the performance of many outlier detection methods with various hyperparameters, on different noise and sparsity levels, with data produced by Contrastive learning.

IV. Results

A. Robust training under label noise on CIFAR10

For the first experiment, we use contrastive learning with a ResNet18 encoder to produce representations of CIFAR10 images in a 512 dimensions vector, with a projection head of size 128 and a batch size of 256. We also choose the temperature parameter of the NCE loss to be 0.5, and we train for 600 epochs.

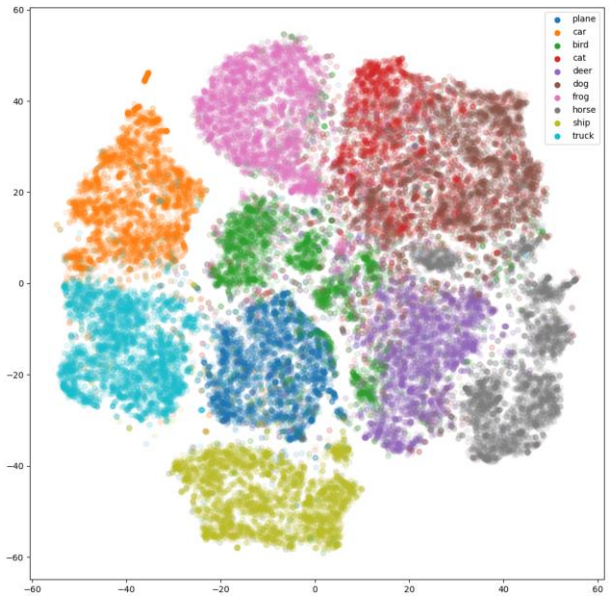


Fig. 2. The representations of CIFAR10 images obtained via contrastive learning, represented as 2 dimensions points thanks to t-SNE [13] dimensionality reduction. Each color correspond to a true label.

Noise	Sparsity							
	0.0	***0.1***	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***
With Cross-Entropy loss								
0.0	90.72							
0.1	90.56	90.66	90.54	90.75	90.53	90.57	90.64	90.53
0.2	89.58	89.98	89.50	89.78	90.50	90.16	89.89	88.87
0.3	89.51	89.37	88.61	89.67	89.40	89.00	85.55	87.46
0.4	87.69	87.03	86.89	86.05	83.44	77.43	72.44	66.90
0.5	82.57	76.90	69.32	73.06	62.79	65.75	54.60	56.49
0.6	65.82	61.80	58.96	54.00	51.47	50.22	45.11	40.44
0.7	41.26	35.35	34.00	36.86	35.46	26.20	27.23	28.54
With ELR loss								
0.0	91.31							
0.1	91.02	91.22	91.00	91.04	91.01	91.10	90.90	90.95
0.2	90.51	90.63	90.63	90.63	90.57	90.54	90.69	90.64
0.3	90.16	90.02	90.41	90.43	90.70	90.21	89.91	90.29
0.4	89.80	88.79	87.15	89.85	85.75	82.42	79.96	71.97
0.5	85.78	80.83	68.11	73.55	64.44	74.17	46.91	56.21
0.6	70.89	73.67	57.21	64.93	42.39	57.36	40.77	40.24
0.7	42.43	36.48	29.71	42.23	40.52	31.17	23.40	21.10

Table 1. Top-1 accuracies of the MLP on different noise and sparsity levels, using the Cross-Entropy loss in the first table, and a noise robust loss (ELR) in the second table, with ResNet18 as encoder.

Fig. 2 shows that Contrastive Learning creates globally well separated representations of the CIFAR10 images in the latent space, except for the labels cats and dogs where there is no distinct separation. It also seems that the clusters are intuitively organized in the sense that animals are separated from vehicles, cars, and trucks are close to each other, as well as for cats and dogs, and in the middle, birds are not so far from planes.

For further experiments, we trained this model for 2000 epochs. t-SNE representations [13] and contrastive losses over epochs can be seen in Appendix B. We use the resulting ResNet18 encoder to represent every image of the CIFAR10 dataset in the latent space, creating a dataset of embeddings on which a simple Multilayer Perceptron (MLP) can be trained. As a first step, we used the Cross-Entropy loss and trained the network on the embeddings in a supervised way, given the noisy labels generated in II.B.

Furthermore, we also trained this network with the noise robust loss function called Early Learning Regularization (ELR) [9], which prevents memorization effect of neural networks by capitalizing on the early learning phase. The use of this loss significantly improve accuracy in almost all noise configurations compared with Cross-Entropy, as Table 1 shows. For comparison, we use as baselines a ResNet18 with Cross-Entropy loss, and a ResNet18 with ELR loss (See Appendix C).

As intended, training an MLP on top of the encoder significantly improves performance compared to the baselines approaches. One may also notice that the method is not robust to sparsity under high-level noise.

A promising future work would be to push the algorithm further by using deeper models, larger batches, and to train for more epochs to observe the resulting representations. This

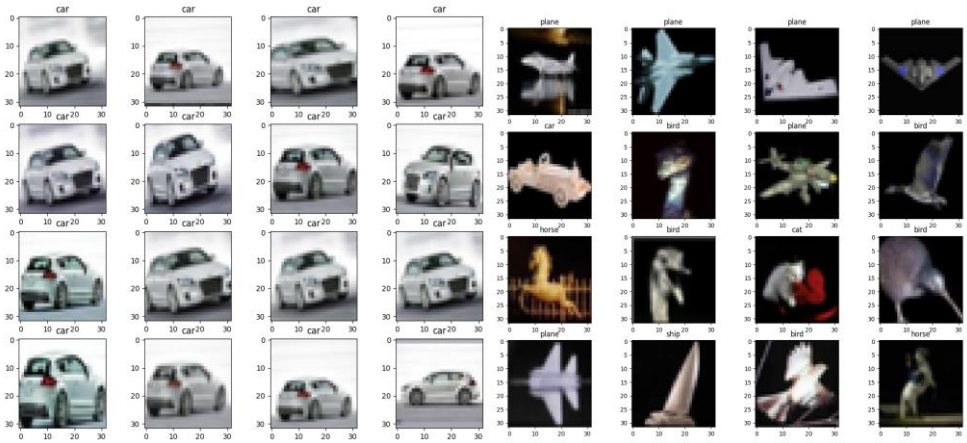


Fig. 4. On the left, 16 randomly chosen CIFAR10 images from the cluster circled in red in Fig. 3. On the right, 16 randomly chosen images from the cluster circled in Blue in Fig. 3.

should increase the accuracies, and may help for the errors' detection methodology described in III.B.

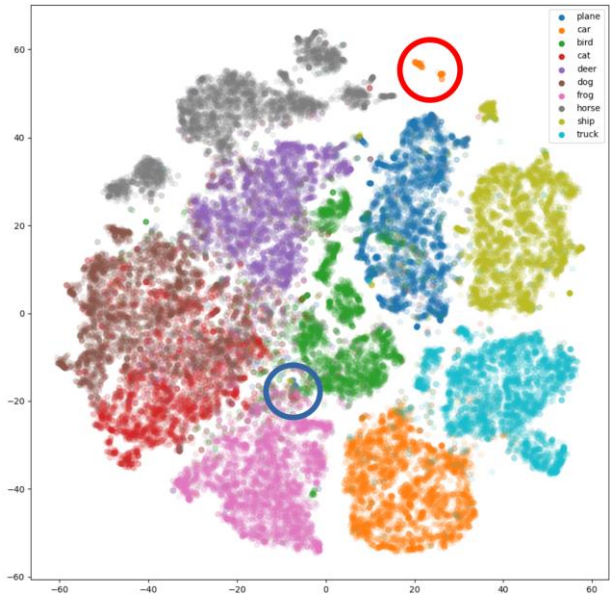


Fig. 3. The t-SNE representations of CIFAR10 images with ResNet18 encoder for 2000 epochs. Each color correspond to a true label.

B. *Clusterization of the latent space*

Fig. 3 presents the t-SNE representations of the CIFAR10 images, obtained via Contrastive Learning with a ResNet18 after 2000 epochs. The representations seem to separate classes, but to generate out-of-distribution clusters as well. In particular, we circled two remarkable small clusters. The red circle indicates an out-of-distribution cluster of cars, while the blue circle is a group of samples that are close to each other but do not belong to the same class. This could be a side effect of applying t-SNE for visualization, or an intrinsic behavior of the learned representations of the Contrastive Learning algorithm.

Fig. 4 shows the images inside the two clusters. We are now better able to understand how the model generates representations. The red cluster contains similar cars with the same color and the same background, and the blue cluster contains images from different classes on a black background.

Having a cluster of samples from the same class is not a problem since a classifier will easily match it to the associated label. However, clusters like the blue one raise a problem and can be hard to identify for a classifier.

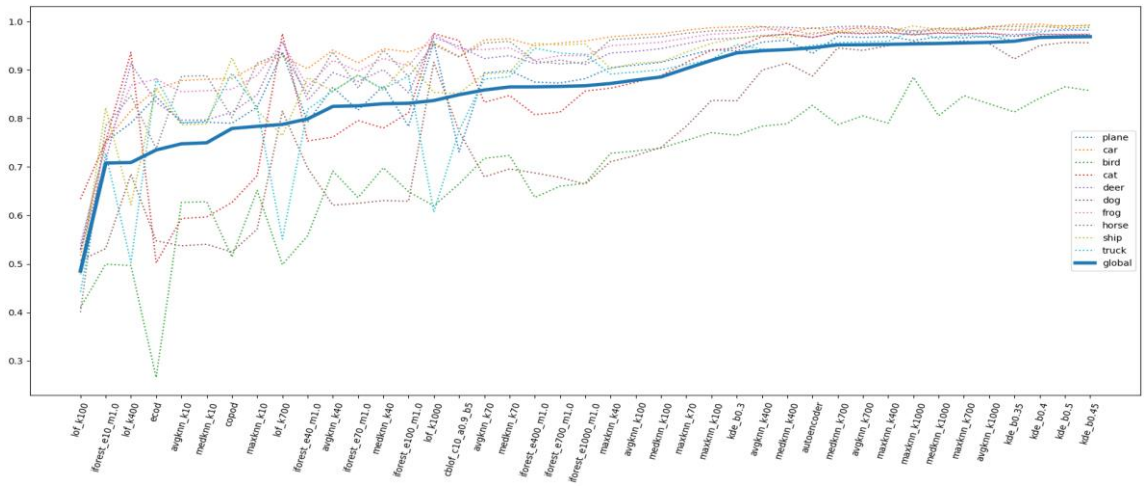


Fig. 5. AUC ROC scores with 30% noise and 20 % sparsity, for every outlier detection models. We compute the score for every class, and globally. The models are sorted by global AUC ROC to make the visualization clearer.

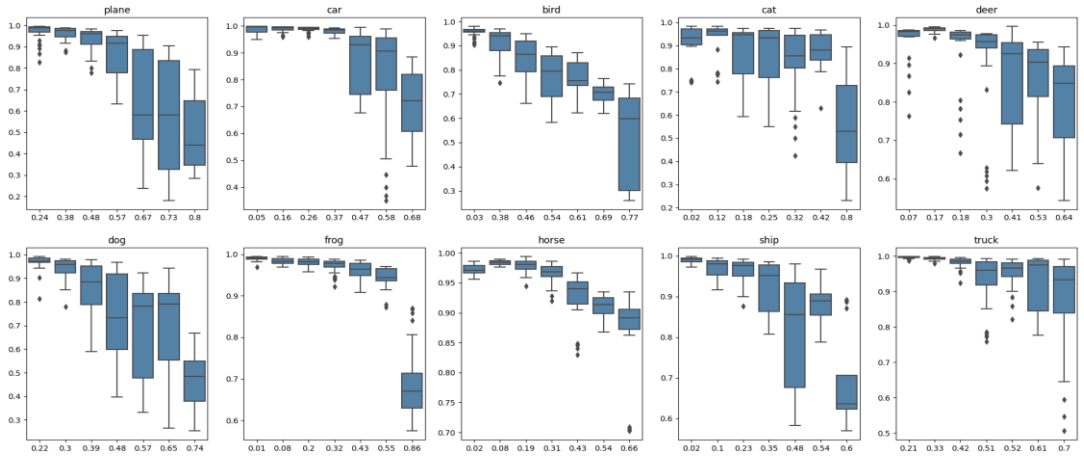


Fig. 6. Class by class distributions of AUC ROC scores computed by KDE models in all noise configurations. i.e. for the class "plane", the subplot on the upper left shows, for every noise rate of this class, the distribution of all the AUC ROC computed by KDE models (five models with various bandwidth parameter)

To overcome this phenomenon, an option is to change data augmentations applied to original samples. Adding stronger color distortion may help to avoid focusing on the background of certain images and improve performance. In general, Contrastive Learning is highly dependent on the data augmentations used [14], and the quality of the set of augmentations is very specific to the dataset itself.

C. Finding errors and outliers

Fig. 5 reports AUC ROC scores computed on the predictions of eight different outlier detection methods with various hyperparameters, on a particular noise and sparsity setting (0.3 / 0.2). We take the average scores across the five seeds to get a representative value, instead of over-interpreting scores from one noise configuration. Results with other noise and sparsity settings are in Appendix D.

The first observation we can make is that the Kernel Density Estimation (KDE) technique outperforms every other methods when comparing the global AUC ROC score (0.97), while keeping a relatively narrow distribution of scores along the ten classes. Still, we can see that it struggles more with the "bird" class. In contrast, KNN-based methods also perform well when the number of neighbors is high enough. For the "maxknn_k1000", which takes the maximum distance to the 1000 nearest neighbors in the class, the global AUC ROC is still high, but the scores of all classes are closer together. In other words, this method does not "sacrifice" any class.

Similarly, the Auto-Encoder performs well, even though we did not tune any hyperparameters. It might need some parameters tuning to perform to its full potential, but still got an AUC ROC around 0.95.

Conversely, the ECOD and COPOD methods do not perform well. The LOF model neither, whereas some combinations of hyperparameters made the IsolationForest algorithm reach an AUC ROC of 0.87.

To compare the difficulties in finding labeling errors inside the classes of CIFAR10, we took the five KDE models of various bandwidths and plotted in Fig. 6 the distributions of AUC ROC for all noises. Unsurprisingly, for all classes, the scores globally drop with the increase of label noise. What is more interesting is the disparity between the classes of when this drop occurs. This dissimilarity might come from a particular disposition of noise matrices, or from an intrinsic difficulty for the algorithm to detect labeling errors in this class. Plots using KNN-based models and all models are in Appendix D.

V. Conclusion

Data cleaning can be used as a preprocessing step to train robust AI models. In this paper, we investigate how to use embedding spaces learned through a contrastive learning approach to detect anomalous data. To evaluate several anomaly detection methods, we controlled the noise associated to each class.

First, we found that when the dataset contained labeling errors, classifiers trained on embedding spaces performed better than classifiers trained directly on the data input. It would be interesting to compute a confidence statistic from the embedding of a data point in future work. This confidence statistic could be used to weight the loss accordingly during the learning phase of the classifier.

Second, we can identify groups of data points inside the embedding space that share a strong intrinsic similarity with each other but are less similar to data points of their own class. Based on this observation, we compared the projection of data points with and without noise to evaluate anomaly detection algorithms. We observed that the performance of anomaly detection algorithms depends strongly on the initial parameters. The more robust methods scales with high hyperparameters values such as KNN or KDE (k=1000 or bandwidth=0.5). The performances also vary inter-classes, with "cat", "bird" and "plane" being the most difficult classes to separate from input noise.

Further work is required to evaluate how to train an embedding space using contrastive learning, taking into account the properties of the available dataset. Ideally, one should be able to choose the learning parameters based on properties such as batch size and embedding dimension to optimize the projection. We could test deeper models, such as ResNet50, to see if the performances of the anomaly detection methods applied to the embedding space scale by the size of the network. Similarly, large batch size might affect the outcome of the embedding space.

Another direction could be the addition of a constraint on the loss of the embedding space, such as a spherical embedding or decision boundary to learn a confidence score

associated with the data heterogeneity with respect to the dataset.

REFERENCES

[1] R. A. Krishna, K. Hata, S. Chen, et al. “Embracing error to enable rapid crowdsourcing”, in : Proceedings of the 2016 CHI conference on human factors in computing systems, p. 3167-3179, 2016.

[2] B. Settles, “Active learning literature survey”, 2009.

[3] A. Ratner, S. H. Bach, H. Ehrenberg, et al. “Snorkel: Rapid training data creation with weak supervision”, in : Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases, p. 269, NIH Public Acces, 2017.

[4] C. E. Brodley, M. A. Friedl. “Identifying mislabeled training data”, Journal of artificial intelligence research, vol. 11, p. 131-167, 1999.

[5] C. Northcutt, L. Jiang, I. Chuang. “Confident learning: Estimating uncertainty in dataset labels”, Journal of Artificial intelligence research, vol. 70, p. 1373-1411, 2021.

[6] H. Zhang, M. Cisse, Y. N. Dauphin, et al. “Mixup: Beyond empirical risk minimization”, arXiv preprint arXiv:1710.09412, 2017.

[7] B. Han, Q. Yao, X. Yu, et al. « Co-teaching : “Robust training of deep neural networks with extremely noisy labels”, Advances in neural information processing systems, vol 31, 2018.

[8] J. Li, R. Socher, S. C. Hoi. “DivideMix: Learning with noisy labels as semi-supervised learning”, arXiv preprint arXiv:2002.07394, 2020.

[9] S. Liu, J. Niles-Weed, N. Razavian, et al. “Early-learning regularization prevents memorization of noisy labels”, Advances in neural information processing systems, vol. 33, p. 20331-20342, 2020.

[10] T. Chen, S. Kornblith, M. Nozouri, et al. “A simple framework for contrastive learning of visual representations”, in : International conference on machine learning, PMLR, p. 1597-1607, 2020.

[11] Y. Xue, K. Whitecross, B. Mirzasoleiman. “Investigating why contrastive learning benefits robustness against label noise”, arXiv preprint arXiv:2201.12498, 2022.

[12] C. Northcutt, L. Jiang, I. Chuang. “Confident learning: Estimating uncertainty in dataset labels”, Journal of artificial intelligence research, vol. 70, p. 1373-1411, 2021.

[13] L. Van der Maaten, G. Hinton. “Visualizing data using t-SNE”, Journal of machine learning research, vol. 9, no 11, 2008.

[14] Y. Tian, C. Sun, B. Poole, et al. “What makes for good views for contrastive learning?”, Advances in neural information processing systems, vol. 33, p. 6827-6839, 2020.

[15] F. T. Liu, K. M. Ting, Z. H. Zhou. “Isolation forest”, in : 2008 eigth ieee international conference on data mining, IEEE, p. 413-422, 2008.

[16] L. J. Latecki, A. Lazarevic, D. Pokrajac, “Outlier detection with kernel density functions”, in : International workshop on machine learning and data mining in pattern recognition, Springer, Berlin, Heidelberg, p. 61-75, 2007.

[17] M. M. Breunig, H. P. Kriegel, R. T. Ng, et al. “LOF: Identifying density-based local outliers”, in : Proceedings of the 2000 ACM SIGMOD international conference on management of data, p. 93-104, 2000.

[18] Z. He, X. Xu, S. Deng. “Discovering cluster-based local outliers”, Pattern recognition letters, vol. 24, no 9-10, p. 1641-1650, 2003.

[19] Z. Li, Y. Zhao, N. Botta, et al. “COPOD: Copula-based outlier detection”, in : 2020 IEEE international conference on data mining (ICDM), IEEE, p. 1118-1123, 2020.

[20] M. L. Shyu, S. C. Chen, K. Sharinnapakorn, et al. “A novel anomaly detection scheme based on principal component classifier”, Miami univ coral gables FI dept of electrical and computer engineering, 2003.

[21] F. Angiulli, C. Pizzuti. “Fast outlier detection in high dimensional space”, in : European conference on principles of data mining and knowledge discovery, Springer, Berlin, Heidelberg, p. 15-27, 2002.

[22] Z. Li, Y. Zhao, X. Hu, et al. “ECOD: Unsupervised outlier detection using empirical cumulative distribution functions”, IEEE transactions on knowledge and data engineering, 2002.

[23] P. Vincent, H. Larochelle, Y. Bengio, et al. “Extracting and composing robust features with denoising autoencoders”, in : Proceedings of the 25th international conference on machine learning, p. 1096-1103, 2008.

[24] K. Sohn. “Improved deep metric learning with multi-class n-pair loss objective”, Advances in neural information processing systems, vol. 29, 2016.

VI. Appendix

A. *Noise matrices used in our experiments*

We generated noise matrices with noises in (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7) and sparsities in (0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7), Fig. 5 presents these noise matrices.

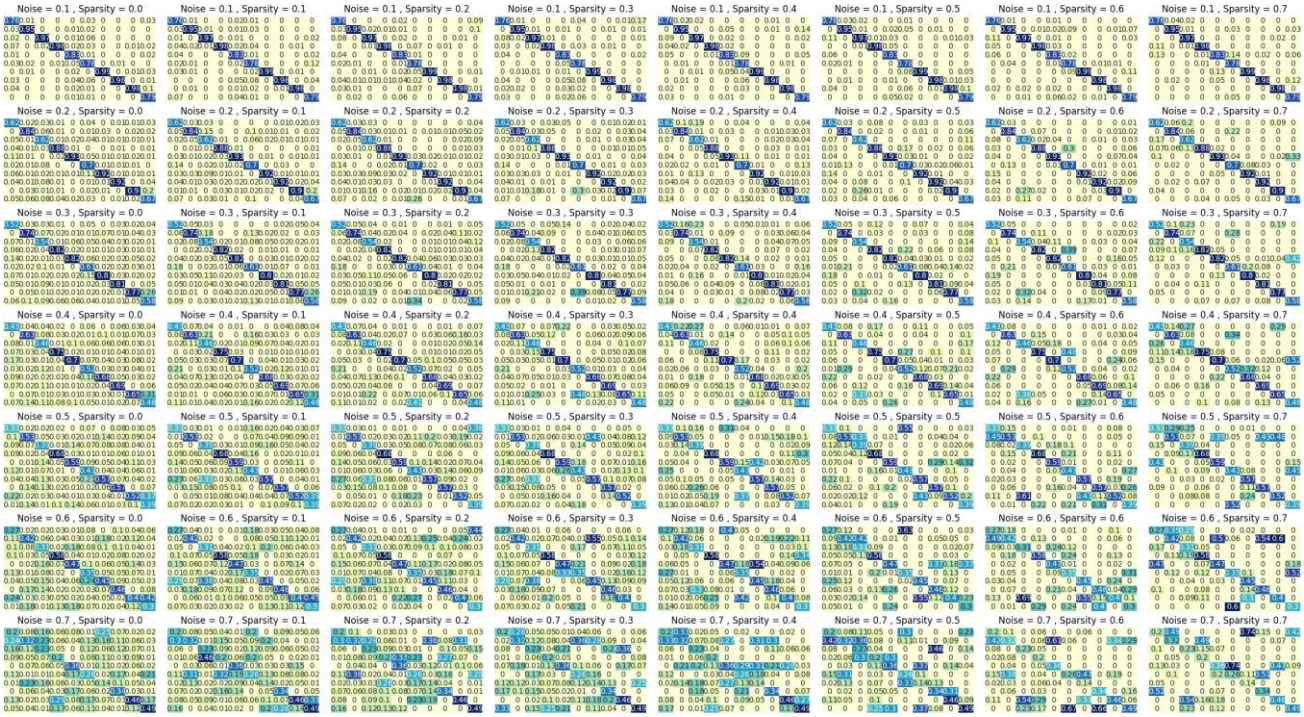


Fig. 7. Noise matrices with seed 0.

B. *Evaluation of Contrastive learning with ResNet18 for 2000 epochs and batches of 256.*

The loss functions shown in Fig. 8 tells that the model is starting to overfit on the training dataset, but the testing loss still decreases after 2000 epochs.

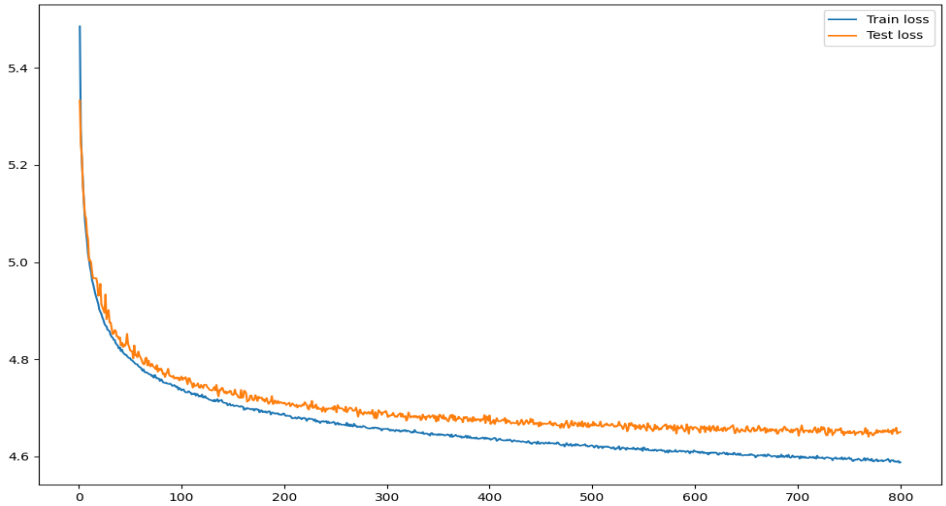


Fig. 8. Evolution of the loss function of Contrastive learning with ResNet18 encoder with batches of 256 over 2000 epochs.

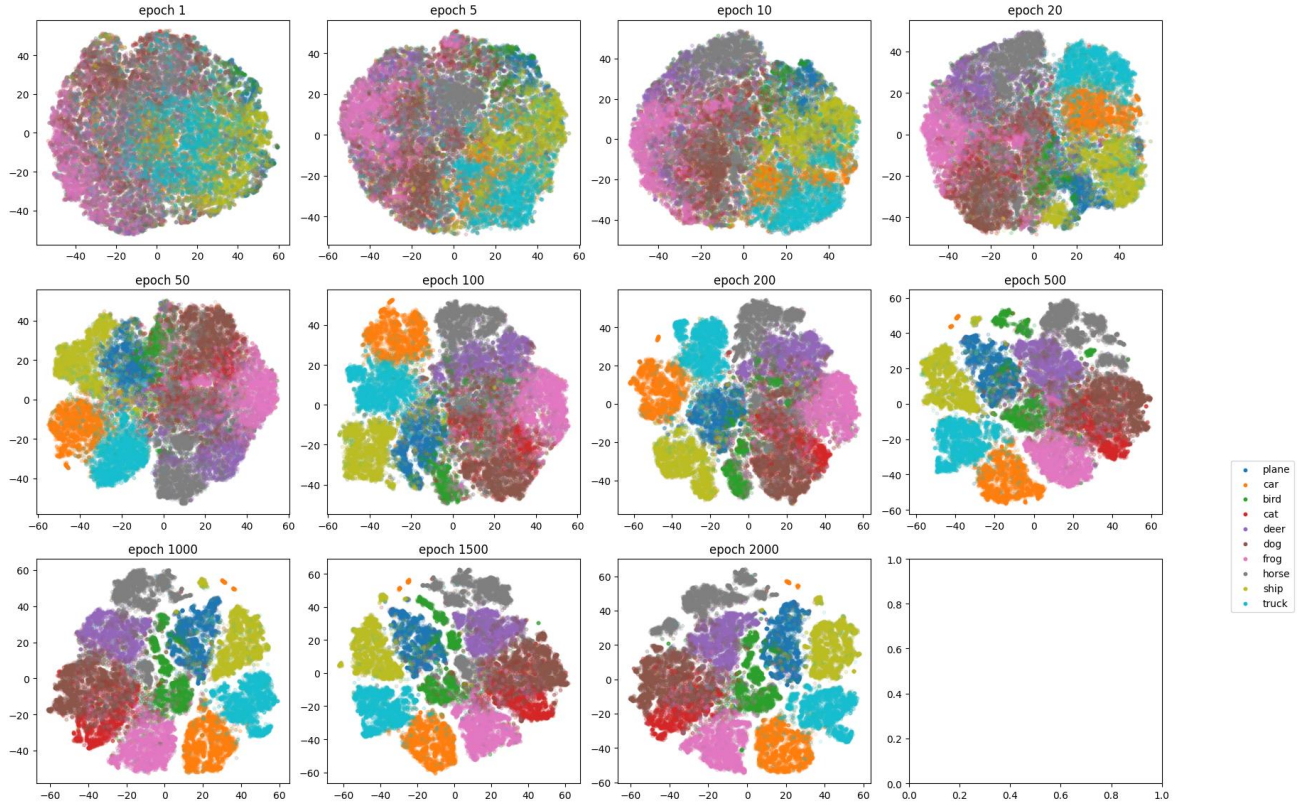


Fig. 9. Evolution of the loss function of Contrastive learning with ResNet18 encoder over 2000 epochs.

Fig. 9 shows the evolution of the t-SNE representations, and we can clearly see how the Contrastive Learning algorithm is splitting the dataset of representations into more and more clusters that globally correspond to their true class.

C. Results of experiments with baseline ResNet18, with Cross-Entropy and ELR loss.

Noise	**Sparsity**							
	0.0	***0.1***	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***
Baseline ResNet18 with Cross-Entropy loss								
0.0	87.64							
0.1	79.88	81.56	79.58	79.26	81.02	80.56	81.25	81.32
0.2	79.15	76.46	78.90	77.67	79.14	76.97	76.22	78.45
0.3	71.76	74.16	72.34	76.00	74.33	75.88	73.56	72.08
0.4	70.71	68.81	66.30	69.00	69.64	65.80	69.59	61.68
0.5	58.45	59.27	60.31	57.91	55.09	59.50	48.61	54.90
0.6	52.77	48.44	42.93	47.79	41.89	48.56	39.18	42.37
0.7	32.20	34.81	35.60	33.46	30.76	31.85	27.89	30.41
Baseline ResNet18 with ELR loss								
0.0	88.44							
0.1	87.31	87.47	86.16	86.79	87.32	87.65	87.54	87.32
0.2	84.65	85.50	85.76	85.13	86.30	86.17	86.10	86.69
0.3	82.62	81.76	83.70	82.96	83.94	84.78	83.33	84.95
0.4	76.80	77.13	74.50	77.62	72.79	71.22	67.19	67.35
0.5	65.01	64.01	60.83	62.58	48.84	62.30	44.13	49.56
0.6	52.29	51.90	47.92	53.40	37.78	47.91	38.25	32.60
0.7	33.16	33.67	28.80	32.97	33.68	27.65	25.90	24.86

D. Results of Outlier detection : Distributions of AUC ROC

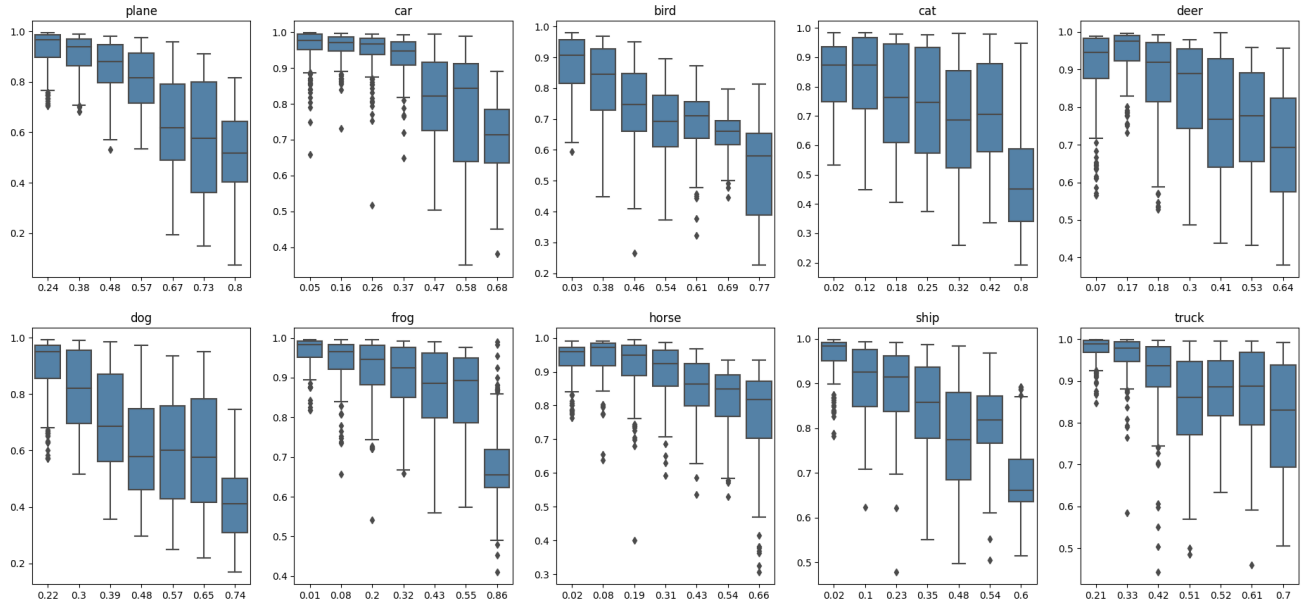


Fig. 10. Class by class distributions of AUC ROC scores computed by all models in all noise configurations

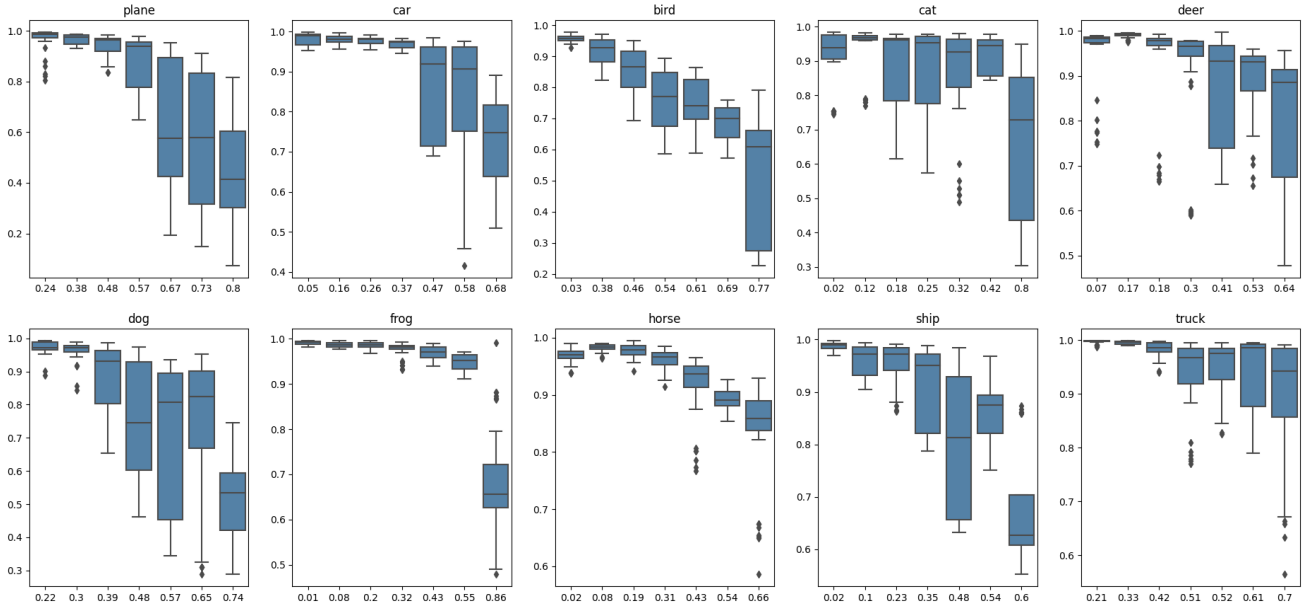


Fig. 11. Class by class distributions of AUC ROC scores computed by KNN-based models in all noise configurations