Actes de la conférence CAID 2020

(Conference on Artificial Intelligence for Defense)

Organisée par

Table des matières

IA et systèmes embarqués

L'IA au service de la cyber-sécurité

L'IA et la sécurité des données

L’IA au service de la gestion de crise

IA et prise de décision

L’exploitation de signaux RADAR et images avec de l’IA

Simplification of Deep Neural Networks for image analysis at the edge

F. de Vieilleville [1], S. May [2], A. Lagrange [1], A. Dupuis [2], R. Ruiloba [1]*

[1] AGENIUM Space, 1, Avenue de l'Europe, 31400 Toulouse, France
(francois.devieilleville, adrien.lagrange, rosa.ruiloba)@agenium.com

[2] CNES, Centre National d'Etudes Spatiales, 18 Avenue Edouard Belin, 31400 Toulouse, France
(stephane.may, aurore.dupuis)@cnes.fr

Abstract. Cubesats platforms expansion increases the need to simplify payloads and to optimize downlink data capabilities. A promising solution is to enhance on-board software, in order to take early decisions automatically. However, the most efficient methods for data analysis are generally large deep neural networks (DNN) oversized to be loaded and processed on limited hardware capacities of cubesats. To use them, we must reduce the size of DNN while accommodating efficiency in terms of both accuracy and inference cost. In this paper, we propose a distillation method which reduces image segmentation deep neural network's size to fit into on board processors. This method is presented through a ship detection example comparing accuracy and inference costs for several networks.

Keywords: Deep Learning, Deep Neural Networks, distillation, small sats, FPGA, image processing on-board.

1 Introduction

Nowadays, cubesats platforms appear to be an attractive and low-cost tool to acquire image data from outer space. However, downlink data capabilities are the bottleneck of the cubesats platforms. It is thus necessary to reduce the volume of data to downstream either by data compression or by data selection. The compression ratio required for image payloads is too high to be obtained by existing compression methods (Buciluǎ et al., 2006; Frosst and Hinton, 2017; Hinton et al., 2015). On-board, feature extraction based on deep learning (DL) (Greenland et al., 2018) provides, by now, the most efficient solution for upstream data reduction. Moreover, it can provide high-value information directly exploitable. However, the most efficient DL models are usually cumbersome due to ensembling methodologies, e.g. hundreds of millions of parameters in the case of the winners of Airbus Ship Detection challenge proposed on

* Corresponding author

Kaggle[1] , and are not compatible with the limited performances of the processors available on board. Some missions integrate specific intelligence which can decide on storage or scenes acquisition regarding cloud detection, e.g. Forward Looking Imager (Greenland et al., 2018). On-board features extraction allows to decide on board if data should be stored and downlinked or not. It can also directly supply the information requested without downloading the full data image. Data reduction by DL-based feature extraction on board will improve the acquisitions capabilities for a defined bandwidth budget and will optimize data downlink providing useful information (Soukup et al., 2016). Instead of relying on ground operator, decisions can be made on board, with efficient algorithms. This aims to increase the payload's autonomy, to release burden on ground segments and to reduce costs (since less operators are required) for creating valuable products. The point here is that all these specific, long, tedious and complicated developments in terms of methodology, software, integration and verification can be replaced by embedded intelligence. For example, improving the reactivity may allow a simple constellation to continuously monitor events and to eventually downlink acquired/processed data when faced with predefined observed activities. Such scenario could also be compatible with satellites being interconnected and allowing downlink for the best positioned satellite on available reception stations. Moreover, the image processing chain on-board can be reprogrammable. The goal of the system design should be to make possible the replacement of the DL components by a new DL network, devoted to other feature extraction required by the mission. Thus, the same cubesat platform could be used for different user-driven missions contributing to optimize mission cost.

However, existing state-of-the-art deep neural networks (DNN) which perform the best on earth observation (EO) tasks are huge networks (several hundred million parameters). They are often aggregates of several models and require a lot of computation power to be run (Bianco et al., 2018). On the other hand, hardware (HW) for computation, in satellites with an image payload, is very limited (Lentaris et al., 2018) due to power consumption restrictions (between 1 and 10 W against 250 W for a NVIDIA Titan X) and dissipation problems. For on-board processing, the reduction of the computational burden linked to inference is the priority (Abdelouahab et al., 2018; Qi et al., 2018). Methods exist to reduce the number of operations of these huge high-performance networks, but the achieved reduction factors with most techniques, on fully convolutional networks, do not exceed a x4, x9 factor (Han et al., 2015a). Consequently, the best DNN cannot be used with a restricted computation power and there is no other choice but to use less reliable, less efficient and smaller networks. Although these smaller networks (<5 million parameters) fit in existing computation payload, they do not bring high accuracy making them potentially useless depending on the mission requirements. For example, the works of the Aiko society have focused on a library called Mirage AI aiming at improving on-board autonomy and, more generally, mission planning. Their work has focused on the reuse of simple or highly efficient networks in terms of performance/number of operations, e.g. ship detection with fast detectors (SSD (Liu et al., 2016), YOLO (Redmon and Farhadi,

[1] e.g. 6th place candidate : https://www.kaggle.com/c/airbus-ship-detection/discussion/71782

2018)) followed by a classification network (MobileNet, Howard et al., 2017). In this work, authors display a total size of 15MB for their networks with inference times of less than 1 second per image, the full image size and the final accuracy being not known. One strategy to alleviate this issue is to try to extract the meaningful parts of these large and complex high-performance DNNs, possibly aggregates or ensemble models, using a teacher/student training method called distillation approach. This approach allows us to take into account what has been learned by the huge teacher model as mapping to vectors of real values and to approximate this mapping as best as we can. The interesting part is that this approach allows small architectures to reach rapidly much higher performances than conventional training (Hinton et al., 2015), even though a state-of-the-art training procedure is used such as augmentation, recent optimizer, learning rate scheduling and policies, stratification on data, batch accumulation, etc. Moreover, this approach being a first step, the other classic simplification methods (pruning, low rank approximation, quantization, weight sharing, …) can still be used. Cumulating the reduction from this approach and classic methods, it is possible to reach high enough reduction rates, e.g. (Polino et al. 2018), to bring high performances networks on board. This paper aims to demonstrate it in the specific context of image segmentation of remote sensing images. The objective of this study is to reduce the size, in terms of number of parameters, of a given high performance network. In order to fit on cubesats processing payloads, the aim is to produce a new network with around 1 million free parameters. This size should fit in mid capacities COTS (Commercial Off-The-Shelf) and available radiation tolerant HW (Lentaris et al., 2018) used on board the satellites. This objective should be reached with a minimum accuracy loss, a maximum loss of 5% of the performances is the target of the study.

2 Simplification workflow implemented

The proposed workflow aims at reducing the number of free parameters drastically. The aim of this paper is to simplify networks reaching very high accuracy, obtained using all the state-of-the-art training strategies without restriction. However, classic methods to reduce neural networks are not sufficient for such huge models. In fact, pruning studies show that a reduction of a factor 2-3 in the number of free parameters in convolutional layers can be achieved without significant accuracy drop (Molchanov et al., 2016) even so, these results are dependent of the involved application and architecture. A similar work can be done on linear subspace approximations (e.g. SVD on matrix) when trying to reduce the number of free parameters in linear operators, like convolution or fully connected blocks. Here, the literature reports reduction factors around 2 or 3 (Lee et al., 2019). Then, quantization is often reduced to computations in int8 instead of float32. Despite a reduction of the memory footprint by a factor 4, this does not reduce the number of free parameters of the network. However, it does improve the global speed since less elementary operations are required when computing int8 arithmetic (Wei et al., 2019). The objective is thus to obtain a better

reduction using a distillation method, the other simplification techniques being considered complementary.

2.1 Considered workflow

The first step is to train a complex state-of-the-art DNN using the complete available data. This process should use all the conventional methods improving the results such as augmentation, recent optimization strategy, complex learning rate scheduling, etc. The idea is to benefit from all the complex learning methods available regardless of the cost to obtain the best teacher network. As an example, it is common to find huge models with more than 200 million free parameters to obtain the best performances such as the one developed in Kaggle-like competitions. In order to fit on cubesats processing payloads, the next step is to produce a new network with around 1 million free parameters as stated in the objective of this paper. The method adopted in this study is to use a distillation process in order to reach this objective. Distillation is a student-teacher learning method introduced by (Buciluǎ et al., 2006). It aims at transferring the knowledge of a big teacher network in a smaller DNN by training the small DNN to predict the output of the teacher model. The aim is to find a small network mimicking the results of a large network, and topology changes are allowed. Distillation have already been used in various application frameworks such as acoustic models (Hinton et al., 2015) or tabular data (Tan et al., 2018). In this paper, a distillation process is developed in the specific context of image segmentation as described in the next section. The distillation strategy is very straightforward as illustrated in Figure 1. The smaller student network is trained to predict the output of the teacher network using a weighted MSE loss function defined as follows:

$$MSE = \frac{1}{N}\sum_{n=1}^{N} w_n \|\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n\|_2^2 \quad (1)$$

where yˆ_n stands for the predicted probability vector gathering the probabilities to belong to each of the C classes, y_n for the reference probability vector produced by the teacher, w_n is the weight associated to sample n which depends of the class of the sample (in experiments, 0.9 for ‘ship’ and 0.1 for ‘non-ship’). We do not present the formula with associated weight for each class, but it is straightforward.

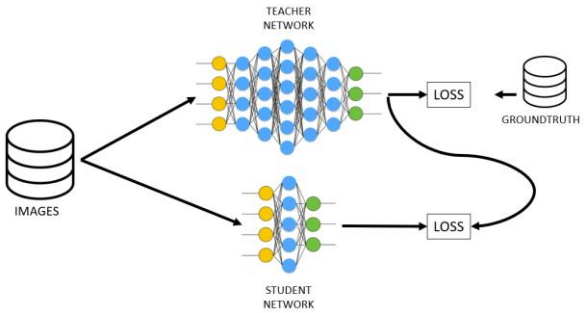


Fig. 1. Distillation process

2.2 Architecture modification

As stated earlier, distillation is the only method allowing a change of network architecture at the cost of a special training. It is actually possible to consider any architecture as student network. In this paper, we propose to start from standard architecture such as U-Net network and to implement some architecture modifications aiming at simplifying specific layers of the networks. The substitution of conventional convolutions by depthwise separable convolutions (Chollet, 2017; Howard et al., 2017) is in particular implemented. Such approach reduces the number of free parameters of these layers by a constant factor of 9. This proposition was implemented in the experiments described in Section 4. After this architecture simplification, the second modification proposed in this paper to reach the desired number of free parameters is the reduction of the number of features used in the output of the intermediate layers. It is a simple way to go from a specific architecture to an architecture with the desired number of parameters. These modifications are performed on a well-established network architecture, such as U-Net for segmentation task (Ronneberger et al., 2015). Applying these simplifications allows one to reach a desired number of parameters that can be fitted into a given HW platform.

3 Experiments

3.1 Task and dataset

The task that was used to test this reduction framework is the Airbus Ship Detection Challenge proposed on Kaggle[2] . The objective is to perform the best ships detection by segmenting remote sensing satellite images. The dataset is composed of more than 200k 3-channels RGB images of 768-by-768 pixels. Unfortunately, no information on the origin and resolution of the images is provided by Airbus. The dataset has very unbalanced classes since ships are small objects and, moreover, only around one quarter of the images contain ships for a total of around 200,000 labelled ships. Figure 2 shows a few examples of images with their associated ground truth segmentation.

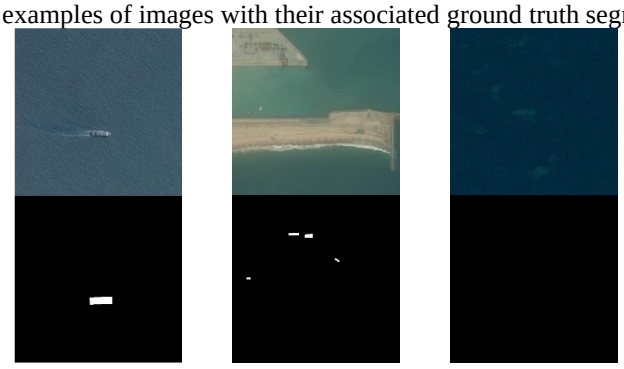


Fig. 2. Dataset Kaggle “Airbus Ship Detection”

[2] https://www.kaggle.com/c/airbus-ship-detection/discussion/71782

3.2 Compared methods

Since the goal of the experiment is to make the most performant ship detection networks suitable for execution on-board the satellites, obtaining these high-performance networks is the first step. In our experiments we started with a 37 million free parameters model based on a Ternaus16 architecture with a VGG16 encoder (Iglovikov et al., 2018). This network achieves high performances in the ADS ship detection competition, i.e., 3% below the top-1 Kaggle score. Thus, it is used as teacher networks in the distillation framework used to reduce the number of parameters of the model. For the network reduction part, 4 student models have been trained using the proposed distillation framework. All these networks are based on a standard architecture, specifically a U-Net architecture (Ronneberger et al., 2015). The number of parameters of this baseline architecture is gradually reduced from 4 million of free parameters to 0.5 million. Table 1 shows how the number of features, before each reduction of the spatial resolution (maxpooling layer), is gradually reduced to obtain a specific number of parameters. In the end, the network selection depends both on the hardware capacities and the performance requirements. In terms of memory occupancy, these networks with roughly 0.5 million parameters can fit on most standard FPGA, with approximately a few megabytes of BRAM (Block Random Access Memory (RAM), dual-port RAM module present into the FPGAs). Thus, if we can maintain high performances for such networks, they will be suitable for execution in FPGA on board the satellites and achieve the use case demonstration success. It has been demonstrated by the execution of the inference of the network “Distilled U-Net 4” (0,5 million parameters) with performance 0.757 in F2 -score, in a mid-range FPGA (the one included in the Xilinx ZCU102 test board) very similar of those used on-board the satellites. It should be also noted that the results produced by the networks have followed a final postprocessing step. The networks produce probabilities to belong to each class but the evaluation procedure, described in the next section, requires having binary masks as outputs. The first step of the postprocessing is thus a binarization of the output using a threshold value. The thresholding operation corresponds to choosing a specific operating point on ROC (Receiver Operating Characteristic) curve of the model. In the following experiments, several thresholds have been tested and the one corresponding to the best result on the validation set have been kept. The second and last step of the postprocessing involves performing a morphological opening on the mask (disk of radius 2 as structural element).

3.3 Results and discussion

The results obtained with the different models are show in Table 1. Their performances are compared in terms of F2-score (object-wise), the metric used in the original ADS challenge. F2-score is computed based on the number of true positives (TP), false negatives (FN), and false positives (FP) resulting from comparing the predicted object to all ground truth objects. In particular, a predicted object is considered a "hit" if its intersection over union with a ground truth object is greater than a given threshold. Then, the final score is generated by averaging the F2-score obtained with differ-

ent thresholds (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95) where the F2-score is computed with the following formula, with β=2,

$$F\beta - score = \frac{(1+\beta^2).TP}{(1+\beta^2).TP + \beta^2.FN + FP} \quad (2)$$

Table 1. Comparison of architecture and performances of all considered models.

Model	Number of free parameters	Number of feature maps by layers	Training time (1x Nvidia 1080Ti)	F2 score (object)
U-Net vanilla	31 M	64/128/256/512/1024	Long > 1 week	0.733
Ternaus16	37 M	64/128/256/512/512	Long > 1 week	0.827
Ensemble-DNN (1st place Kaggle competitor)	30 M (x10)	NA	NA	0.855
Distilled U-Net 1	4 M	64/128/256/512/1024	32 hours	0.781
Distilled U-Net 2	2 M	64/128/256/512/512	32 hours	0.780
Distilled U-Net 3	1 M	64/128/256/256/256	32 hours	0.774
Distilled U-Net 4	0.5 M	64/128/192/192/192	32 hours	0.757

From the results shown in Table 1, it is possible to draw several conclusions. First of all, we can see that there is indeed a drop of performances between the teacher network *Ternaus16* and the student networks (*Distilled U-Net 1, 2, 3 and 4*). The drop of performance appears to be around 5% for the biggest distilled model (*Distilled U-Net 1*, 4 M parameters) and remains stable until reduces for the model of 1 million parameters (*Distilled U-Net 3*). It seems that below 1 million parameters performances start to decrease more constantly. However, when comparing with U-Net vanilla (*U-Net vanilla*, first line in Table 1), learned from scratch, without distillation which is the original architecture of the distilled model, it is worth noting that even the performances of the distilled model with 0.5 million parameters (*Distilled U-Net* 4) gives better results. It proves that the distilled models can benefit from the high-performances teacher model even though if the architecture is entirely different (Ternaus 16 to U-Net).

4 Conclusion

Distillation is a generic and re-usable workflow for simplifying DL networks for defining new Earth Observation (EO) products generated on-board the satellites. It can reduce the uptaking cost of innovative deep learning technologies for on-board use.

This paper presented a distillation strategy that allows the user to reduce the size of a network to desired number of parameters in order to fit in a specific hardware.

A use case performing segmentation of EO images was implemented and showed, in particular, that is possible to reduce drastically the number of parameters with a minimal drop of performances. Moreover, the proposed reduction framework has the

advantage to be compatible with additional reduction methods (pruning, quantization, etc).

The reduced size of the models makes them compatible with the resources of some existing radiation hardened FPGA and COTS, using DPU and BRAM only. In case of design requirements which make the use of external memory mandatory (DDR), an error correction controller would be necessary (e.g. using CRC) to ensure robustness to radiations.

Future works include the study of the robustness to some radiation errors, in particular the impact of the SEU (single event upset) on inference quality for different sizes of models, the influence of additional reduction methods on the performances of the model and evaluate other performance results, such a comparison upon memory use and power consumption. Additionally, the presented method is to be tested on other datasets, more specifically additional use cases featuring planes or building detection are being considered.

Acknowledgements

The authors acknowledge the support from the CNES, French government Space Agency, and specially the DSO/SI/2A department, under the contract N° 190392/00 “Smart payloads” which allowed to perform a significant part of the work presented in this paper.

References

1. Abdelouahab, K., Pelcat, M., Serot, J., Berry, F., 2018. Accelerating cnn inference on fpgas: A survey. ArXiv Prepr. ArXiv180601683.
2. Bianco, S., Cadene, R., Celona, L., Napoletano, P., 2018. Benchmark Analysis of Representative Deep Neural Network Architectures. IEEE Access 6, 64270–64277. https://doi.org/10.1109/ACCESS.2018.2877890
3. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A., 2006. Model compression, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 535–541.
4. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., 2018. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. ArXiv180202611 Cs.
5. Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1251–1258.
6. Frosst, N., Hinton, G., 2017. Distilling a Neural Network Into a Soft Decision Tree. ArXiv171109784 Cs Stat.
7. Greenland, S., Ireland, M., Kobayashi, C., Mendham, P., Post, M., White, D., 2018. Development of a minaturised forwards looking imager using deep learning for responsive operations. ESA.
8. Han, S., Mao, H., Dally, W.J., 2015a. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. ArXiv Prepr. ArXiv151000149.

9. Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the Knowledge in a Neural Network. ArXiv150302531 Cs Stat.
10. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. ArXiv Prepr. ArXiv170404861.
11. Iglovikov, V., Seferbekov, S.S., Buslaev, A., Shvets, A., 2018. TernausNetV2: Fully Convolutional Network for Instance Segmentation., in: CVPR Workshops. pp. 233–237.
12. Lee, D., Kwon, S.J., Kim, B., Wei, G.-Y., 2019. Learning Low-Rank Approximation for CNNs. ArXiv190510145 Cs Stat.
13. Lentaris, G., Maragos, K., Stratakos, I., Papadopoulos, L., Papanikolaou, O., Soudris, D., Lourakis, M., Zabulis, X., Gonzalez-Arjona, D., Furano, G., 2018. High-performance embedded computing in space: Evaluation of platforms for vision-based navigation. J. Aerosp. Inf. Syst. 15, 178–192.
14. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. Ssd: Single shot multibox detector, in: European Conference on Computer Vision. Springer, pp. 21–37.
15. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J., 2016. Pruning convolutional neural networks for resource efficient inference. ArXiv Prepr. ArXiv161106440.
16. Qi, B., Shi, H., Zhuang, Y., Chen, H., Chen, L., 2018. On-board, real-time preprocessing system for optical remote-sensing imagery. Sensors 18, 1328.
17. Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. ArXiv Prepr. ArXiv180402767.
18. Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv150504597 Cs.
19. Soukup, M., Gailis, J., Fantin, D., Jochemsen, A., Aas, C., Baeck, P., Benhadj, I., Livens, S., Delauré, B., Menenti, M., 2016. HyperScout: Onboard Processing of Hyperspectral Imaging Data on a Nanosatellite, in: Proceedings of the Small Satellites, System & Services Symposium (4S) Conference, Valletta, Malta.
20. Tan, S., Caruana, R., Hooker, G., Lou, Y., 2018. Distill-and-compare: auditing black-box models using transparent model distillation, in: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society. ACM, pp. 303–310.
21. Wei, X., Liu, W., Chen, L., Ma, L., Chen, H., Zhuang, Y., 2019. FPGA-Based Hybrid-Type Implementation of Quantized Neural Networks for Remote Sensing Applications. Sensors, 2019 Feb 22;19(4). pii: E924. doi: 10.3390/s19040924.

Contrôle de l'environnement d'exécution d'un processus avec des relevés de Hardware Performance Counters (HPC)

Luc Bonnafoux[1], Francisco-Pierre Puig[2], Quentin Lhoest[2], Thomas Renault[2] et Adrien Benamira[2]

[1] Thales, Gennevilliers
[2] CentraleSupélec (Université Paris-Saclay), Gif-sur-Yvette
luc.bonnafoux@thalesgroup.com

Abstract. Ces dernières années de nouvelles classes d'attaques, exploitant des failles dans la micro-architecture des processeurs, ont été découvertes : les attaques par canaux cachés. Ces attaques abusent du partage des ressources matérielles (cache, pipeline d'exécution) dans les processeurs modernes pour, à partir d'un processus attaquant, inférer des informations sur un processus victime ; cela au dépend des méthodes de cloisonnement des processus traditionnelle, notamment la *Memory Management Unit* (MMU). Dans ce papier nous présentons comment les *Hardwares Performance Counter* (HPC), combinés au *machine-learning* peuvent servir d'outil de surveillance de l'environnement dans lequel s'exécutent des processus à protéger. Nous montrons qu'il est possible de détecter quand un processus est soumis à une attaque par canal caché, ici Zombieload, sur un processeur Intel. Enfin, nous montrons comment notre model permet d'apporter des informations permettant une analyse fine de la source de l'anomalie détectée.

Keywords: Hardware Performance Counter, attaque par canal caché, machine learning, interprétabilité de l'IA

1 Introduction

1.1 Les attaques par canaux cachés

Dans le but d'augmenter les performances d'exécution, l'architecture moderne des processeurs courants (Intel, ARM…) a été complexifiée pour permettre l'exécution de plusieurs processus parallèlement (*Hyper-threading* sur Intel), pour diminuer les temps d'accès disque (cache partagé) et optimiser l'utilisation du *pipeline* des processeurs (exécution spéculative, exécution désordonnée…). Cela a amené à partager plusieurs ressources matérielles entre les processus au sein des cœurs (cache, *Translation Lookaside Buffer* (TLB)…).
Les nombreuses barrières qui avaient été dressées pour permettre le cloisonnement des processus au niveau logique (MMU…) peuvent être contournées en tirant profit d'informations secondaires fuitant par le biais des ressources matérielles partagées,

comme par exemple le temps d'accès à ces ressources. Ainsi plusieurs attaques ont été découvertes au cours de ces dernières années mettant à mal ces micro-architectures modernes. Il est possible de citer Spectre [1], Meltdown [2]…

ZombieLoad.
L'attaque ZombieLoad utilise un défaut dans l'*Hyper-threading* pour observer les valeurs chargées ou stockées par les processus s'exécutant sur un même cœur de processeur. Dans [3], les auteurs montrent une faille de sécurité permettant d'exploiter les *fill buffer* partagés entre deux threads s'exécutant sur un même cœur. Lorsque des valeurs sont écrites ou chargées en mémoire, si la ligne correspondant à l'adresse n'est pas présente dans le cache L1 alors la valeur est stockée dans un *fill* buffer. Dans certaines conditions de faute, il est possible d'accéder temporairement au contenu du *fill buffer* et de faire fuiter la valeur dans un cache. Ensuite, cette valeur peut être retrouvée avec une attaque par canal caché telle que Meltdown. Ainsi, il est possible d'espionner de façon non sélective toutes les données passant par ce buffer.
Un prototype est disponible sur Github [4]. Cette faille touche principalement les processeurs Intel.

1.2 Les Hardware Performance Counter

Les *Hardware Performance Counter* (HPC) sont des compteurs situés dans la micro-architecture des processeurs modernes pour relever des métriques sur l'exécution du processeur : nombre d'instructions effectuées, nombre d'accès au cache, nombre de branches prises, nombre d'instructions arithmétiques effectuées… Ces compteurs sont généralement utilisés pour optimiser l'exécution d'un logicielle. Cependant ils peuvent aussi être détournés pour surveiller le comportement d'un programme.
L'utilisation de la mesure des HPC a plusieurs avantages. Le premier est qu'étant intégrés dans l'architecture des processeurs, le relevé de ces valeurs a un impact limité sur les performances du processeur. De plus, contrairement à des mesures logicielles les HPC peuvent être audités sans modifier le code source des binaires à analyser. Ainsi, l'outil de mesure peut être indépendant du code exécuté.
Il existe plusieurs outils permettant de relever ces compteurs ; on peut citer notamment l'outil *perf* en ligne de commande sous Linux, ou encore la librairie PAPI [5] que nous avons employée pour nos relevés. Cette librairie permet une résolution maximale bien plus importante que via l'outil *perf*. Bien que ce degré de résolution ne soit pas crucial pour le profilage de la performance, il peut l'être pour détecter les attaques par canaux cachés.

1.3 Détection d'anomalies au moyen des HPC

L'état de l'art fait référence de l'utilisation des HPC pour la détection d'anomalies dans l'exécution des processus. Il est possible de citer des travaux faits sur Spectre et Meltdown [6]. Ces attaques se basant sur les temps d'accès au cache pour faire fuiter de l'information, les modèles décrits dans ces publications utilisent principalement les HPC en lien avec les performances des caches (L3_TCM, L3_TCA…).

Les HPC ont aussi été utilisés pour détecter que des comportements anormaux d'applications non liés à des attaques par canaux cachés : Return Oriented Programing (ROP) [7], RowHammer [8]…

Dans les travaux décrits précédemment la mesure des HPC s'est faite directement sur le processus attaquant. Cependant, dans certaines circonstances il peut être souhaitable de non pas chercher une anomalie dans le comportement de l'ensemble des processus pour détecter les potentiels attaquants mais plutôt de surveiller les processus à protéger pour s'assurer qu'ils sont exécutés dans un environnement sain. C'est cette seconde approche que nous avons étudiée dans nos travaux et que nous allons détailler.

2 Conditions expérimentales

Nous avons choisi de nous intéresser à l'attaque ZombieLoad qui permet à un processus attaquant **A** de récupérer des informations manipulées par le processus **P** s'exécutant sur le même cœur que lui. Nous avons utilisé pour cette attaque un processeur Intel I5. **P** charge une valeur en mémoire et **A** cherche à savoir quelle est cette valeur au moyen de l'attaque ZombieLoad.

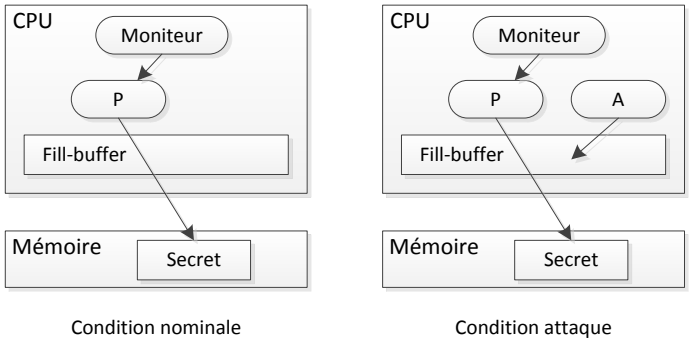


Fig. 1. Mise en œuvre de l'attaque ZombieLoad et des mesures.

Pour les expérimentations, un processeur Intel I5 a été utilisé. Sur ce processeur il y a 45 HPC disponibles qui ne peuvent être accédés que par groupe de cinq. Ils permettent de mesurer de nombreuses métriques telles que :

- Le fonctionnement des caches d'instructions et de données du niveau L1 au L3 à la fois en lecture et en écriture, par exemple : L3_DCR (L3 Data Cache Read), L2_DCM (L2 Data Cache Miss), CA_SNOOP (Cache Snoop)… ;
- Les branches d'exécutions prises ou leur prédiction, par exemple : BR_MSP (Branche Instruction Mispredicted), BR_TKN (Conditional Branch Taken)… ;
- Différentes informations sur la pipeline du cœur, par exemple : TLB_IM (Instruction TLB Misse), RES_STL (Cycle processor stalled on ressource)…

Le premier HPC de chaque groupe est toujours TOT_INS qui est le nombre d'instructions exécutées, nous avons ainsi constitués 11 groupes de HPC en cherchant à ne pas concentrer les mesures dans chaque groupe sur un seul constituant du processeur pour pouvoir avoir une vision large de l'activité sur celui-ci avec chacun des groupes.

L'exécution de **P** est en permanence mesurée et des valeurs de HPC sont relevées de manière périodique pour constituer des séries temporelles comprenant :

- Les cinq valeurs de HPC relevées ;
- Un label indiquant si l'attaque est en cours ou non ;
- Des informations annexes sur le mode opératoire du relevé (processus victime lancé en avec la commande sudo ou non ; niveau de priorité du processus victime ; présence d'autres programmes en tâche de fond sur le même cœur que le processus ou non…).

Les relevés ont été effectués avec plusieurs types de conditions externes car, expérimentalement, elles affectent beaucoup les valeurs des HPCs relevées. Les mesures sont effectuées à intervalle de temps régulier (à environ 10 kHz) et ce sont des compteurs d'événements : à chaque échantillonnage, les valeurs des compteurs sont relevées et remises à zéro.
Nous avons réalisé 2700 relevés de séries temporelles de 30 secondes chacune.

3 Prétraitement des données

Tout d'abord, la première série temporelle TOT_INS est utilisée pour redimensionner les autres séries relevées en nombre d'instruction exécutées par le programme au lieu du temps. Cela permet d'abstraire une partie du bruit de fond sur le processeur dû à l'exécution d'autres programmes entraînant un ralentissement aléatoire du processus observé.

Ensuite, un ré-échantillonnage est effectué pour avoir des points de mesures espacés d'un nombre constant d'instructions. Nous avons choisi de fixer la fréquence de ré-échantillonnage à une mesure toutes les 10^6 instructions. Puis, un fenêtrage est réalisé pour obtenir des fenêtres de mesures de taille fixe avec les hyper paramètres suivant : w la taille des fenêtres en nombre de mesures et s le décalage entre le début des fenêtres en nombre de mesures. En choisissant $s << w$ cela permet de garantir que le résultat des modèles entraînés sera invariant par translation. Nous avons pris w = 1000 mesures et $250 <= s <= 1000$ mesures, ce qui suppose que si une attaque est présente et cause une perturbation des mesures, le temps caractéristique de ces perturbations est au plus de 10^9 instructions pour pouvoir être observable par nos modèles. Finalement, chaque HPC est normalisé indépendamment des autres.

Une transformée de Fourier est appliquée ou non à chaque fenêtre en fonction des modèles de détection (notée TF dans la suite du document).

4 Analyse des mesures de HPC en boîte noire

Deux modèles de réduction de dimensions ont été évalués pour différencier les traces de données nominales des anomalies : l'Analyse en Composante Principale (ACP) et un auto-encodeur basé sur un réseau de neurones dense.

Ces modèles fonctionnent par une réduction de la dimension des données suivie par une reconstruction des valeurs. Dans une première phase d'apprentissage, les modèles sont entraînés pour minimiser l'erreur de reconstruction sur un jeu de données de test contenant uniquement des relevés nominaux. Ensuite, un seuil est fixé sur l'erreur de reconstruction acceptable. Nous avons choisi la valeur seuil en nous basant sur des exemples d'attaque pour lesquels nous avons calculé la valeur seuil permettant de maximiser la formule de score suivante :

$$\text{score} = \frac{TP}{TP+FN} \times \frac{TN}{TN+FP}.$$

Cependant, l'approche que nous avons choisie pour détecter au mieux une attaque donnée pourrait être améliorée en choisissant par exemple une valeur seuil correspondant à un écart type déterminé de l'histogramme d'erreur admis pour les données nominales.

Durant la phase opérationnelle, l'erreur de reconstruction est comparée à ce seuil. Au-delà d'une certaine valeur seuil il est considéré que c'est une anomalie.

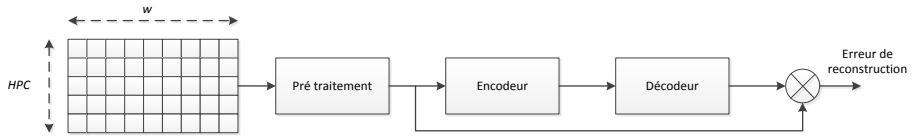


Fig. 2. Chaîne de traitements pour la construction de l'erreur.

L'évaluation de la performance des modèles est faite au moyen de la mesure F_1, telle que :

$$F_1 = 2 * \frac{précision * rappel}{précision + rappel}$$

Nous entraînons nos modèles sur 50000 fenêtres d'échantillonnage tirées de séries temporelles correspondants à des exécutions nominales. L'évaluation de la détection d'anomalies est faite avec un ensemble de 100000 fenêtres d'échantillonnage avec un pourcentage égal d'exécutions nominales et d'attaques.

Pour l'ACP, plusieurs tailles de dimension d'espace latent ont été testées ; les meilleurs résultats sont obtenus pour une taille 1. L'auto-encodeur est composé d'un encodeur de 2 couches de perceptrons séparés par une fonction ReLu ; il en va de même pour le décodeur. Une transformée de Fourier est appliquée lors du prétraitement, et l'espace latent de l'auto-encodeur est de dimension 32.

6

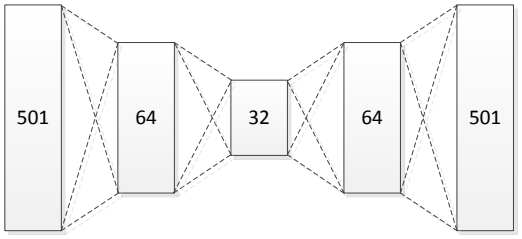


Fig. 3. Modèle d'auto-encodeur utilisé.

Les résultats dans **Tableau 1** représentent les résultats obtenus avec les groupes de HPC les plus significatifs.

Tableau 1: Résultats (F_1) pour les groupes de HPC les plus significatifs (en %).

	Groupe 3	**Groupe 5**	**Groupe 7**	**Groupe 9**
ACP avec TF	98,8 ± 0,0	95,6 ± 0,0	97,8 ± 0,0	94,7 ± 0,0
ACP sans TF	95,6 ± 0,0	94,8 ± 0,0	98,5 ± 0,0	97,2 ± 0,0
RN Dense	98,8 ± 0,0	95,0 ± 0,4	98,3 ± 0,0	96,9 ± 0,1

Les modèles développés permettent donc de détecter des anomalies de fonctionnement d'un processus précis pour certain des groupes de HPC que nous avons constitués. Cette solution nécessite cependant une supervision lors de la phase d'entraînement et a besoin d'être ré entrainée si le processus à surveiller change.

Ainsi la solution de détection d'anomalie proposée est appropriée pour des systèmes embarqués réalisant un nombre de tâches limité et ayant donc un comportement attendu constant dans le temps.

5 Interprétation des résultats

La confiance dans les alertes remontées par le modèle de détection d'anomalies est un élément clé pour faire accepter ce type de solution dans un produit de sécurité. Ainsi, il est nécessaire de pouvoir apporter des éléments de justification de la prise de décision pouvant être audités par un expert humain.

Des outils ont déjà été développés pour donner une mesure de l'importance de chaque variable dans la décision du modèle. Il est possible de citer Lime [9] et SHAP (***SH****apley* ***A****dditive ex****P****lanations*) [10]. Lime donne une interprétation locale d'une décision en se basant sur le gradient de chaque variable dans le modèle au point de mesure. SHAP se base sur les *shapley values* définies dans la théorie des jeux. Cependant, ces méthodes pâtissent de la grande dimensionnalité de nos données qui entraîne des calculs coûteux.

Nous proposons donc une méthode alternative basée sur les modèles de détection d'anomalie par auto-encoder. En effet, ils permettent de mesurer quelle valeur de HPC a été mal reconstruite. Cela permet d'avoir un premier niveau d'analyse en identifiant quel mécanisme, mesuré par un HPC, a eu un comportement anormal. L'erreur de reconstruction étant recalculée à chaque fenêtre temporelle, ceci permet de recons-

truire l'enchaînement des anomalies détectées sur les différents mécanismes du processeur.

Les figures **Fig. 4** et **Fig. 5** montrent les histogrammes d'erreur de reconstruction pour les HPC des groupes 3 et 9. Il apparaît clairement pour certains des HPC que l'histogramme d'erreur de reconstruction avec attaque (en orange) diffère de l'histogramme sans attaque (en bleu).

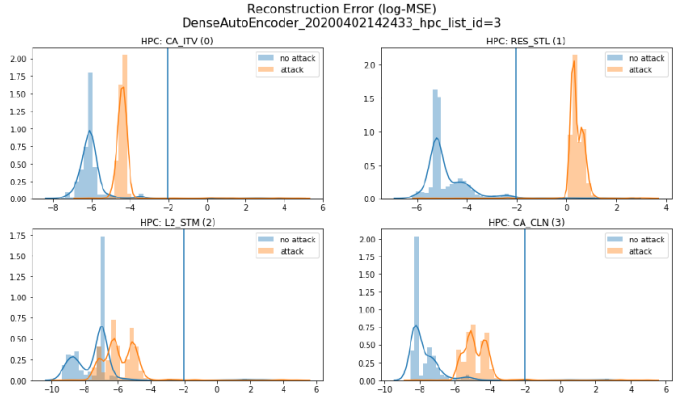


Fig. 4. Histogramme d'erreur de reconstruction pour le groupe de HPC 3.

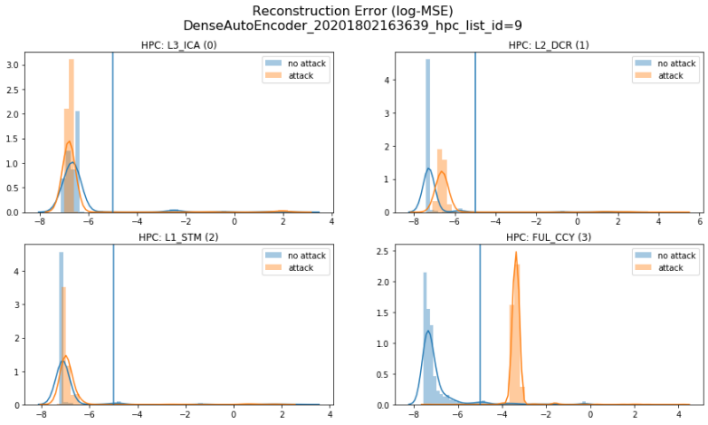


Fig. 5. Histogramme d'erreur de reconstruction pour le groupe de HPC 9.

Ainsi, on s'aperçoit que les HPC pour lesquels la distribution d'erreurs de reconstruction change lors d'une attaque sont :

- Dans le groupe 3, RES_STL, L2_STM, CA_ITV, CA_CLN
- Dans le groupe 5, REF_CYC
- Dans le groupe 7, NATIVE_MASK, STL_ICY
- Dans le groupe 9, L2_DCR, FUL_CCY

Ces HPC sont pour la plupart liés à des performances du cache mémoire ce qui montre que les modèles de détection réagissent très probablement à l'attaque par me-

sure du temps d'accès au cache fait par l'attaquant pour récupérer la valeur ayant été exfiltrée du *fill buffer*. Il est aussi notable que les HPC significatifs mesurent des performances de ressources partagées par l'ensemble des processus, tandis que les HPC mesurant des caractéristiques propres au processus victime (nombre de branches prises, nombre d'instruction arithmétiques effectuées…) ne sont pas modifiés par l'attaque. Ces HPC peuvent cependant servir à détecter des attaques internes au processus (dépassement de *buffer*, injection de code (ROP)…).

6 Conclusion

Nos travaux montrent que la surveillance d'un environnement d'exécution sur un processeur permet de détecter des attaques par canaux cachés. Cela est d'autant plus intéressant que ces attaques peuvent franchir les cloisons traditionnelles d'un système d'exploitation ou d'un hyperviseur pour espionner des processus victimes. La construction du modèle de détection s'est faite en boîte noire sans connaissance à priori sur les mesures remontées par les HPC. De plus, nous proposons une méthode permettant de gagner une information précise sur les sources d'anomalies dans l'exécution du processus permettant ainsi une interprétation à postériori de la prise de décision par le modèle.

Références

1. Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., Yarom, Y : Spectre Attacks : Exploiting Speculative Execution, *40th IEEE Symposium on Security and Privacy* 2019
2. Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., Hamburg, M. :Meltdown, Reading Kernel Memory from User Space, *27th (USENIX) Security Symposium* 2018
3. Schwarz, M., Lipp, M., Moghimi, D., Van Bulck, J., Stecklina, J., Prescher, T., Gruss, D. : Zombieload : Cross-privilege-boundary data sampling, *CCS*, 2019
4. ZombieLoad, https://github.com/IAIK/ZombieLoad (2020)
5. PAPI, https://icl.utk.edu/papi (2020)
6. Depoix, J., Altmeyer, P. :Detecting Spectre Attacks by identifying Cache Side-Channel Attacks using Machine Learning, *WAMOS2018*, August 2018, Wiesbaden
7. Wang, X., Backer, J. :SIGDROP : Signature-based ROP Detection using Hardware Performance Counters, *arXiv preprint arXiv :1609.02667*, 2016
8. Gulmezoglu, B., Moghimi, A., Eisenbarth, T., Berk, S. : FortuneTeller : Predicting Microarchitectural Attacks via Unsupervised Deep Learning, *arXiv preprint arXiv :1907.03651v1*, 2019
9. Tulio Ribeiro, M., Singh, S., Guestrin, C. : « Why Should I Trust You ? » Explaining the Predictions of Any Classifier, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1135-1144, 2016
10. Lundberg, S., Lee, S. : A Unified Approach to Interpreting Model Predictions, *in Advances in Neural Information Processing Systems 30*, pp. 4765-4774, Curran Associates, Inc., 2017

Intrusion Detection based on Electromagnetic Side-channel Signals Analysis

Fred Ngolè Mboula[††] Tristan Bitard-Feildel [†] Erwan Nogues[⋆†] [††]
CEA, Université Paris-Saclay, Gif-sur-Yvette, France,
fred-maurice.ngole-mboula@cea.fr
⋆ Univ. Rennes, INSA Rennes, IETR - UMR CNRS 6164, France
† DGA-MI, Bruz, France
Emails: firstname.lastname@def.gouv.fr

No Institute Given

Abstract. This paper proposes a novel approaches that exploit electromagnetic (EM) side-channel signals to design non-protocol based Intrusion Detection System (IDS). EM emanations side-channels are captured on power lines of an infrastructure. They are used to identify the presence of any type of electronic devices onto a physical network. To that purpose, this paper proposes an hybrid artificial intelligence (AI) approach and a deep learning based approach. Thy learn the structure of the EM unintentional emanations of the legit devices composing the infrastructure as a reference profile. In a second step, they record and analyse the current emanations to compare and detect any kind of unwanted emanations. Therefore these proposals can be used as a Intrusion Detection System (IDS) that can trig an alarm as soon as a intrusion is detected. The results show that intrusion can be detected in various scenarios whatever the activity of the legit computers of the network. Furthermore, the capture device used is based on inexpensive off-the-shelf components that makes the deployment onto real network easy.

Keywords: Side-channel signals, autoencoder, intrusion detection, dictionary learning, sparsity, recurrent neural networks

1 Introduction

1.1 Context and related works

Detecting intruders on a network is part of the analysis of Information Systems Security (INFOSEC). The goals of the intruder can be multiple: interception and listening of network traffic and exchanged data, commands injection, etc. Existing solutions for detecting intruders on a network are mainly based on the network's traffic analysis in order to detect any form of anomaly. Indeed, known techniques recover all network traces in order to filter legitimate traffic and detect traces that the intruder would generate (see for example [1, 2]). With the advent of wireless networks, intruders can now seek to integrate directly into

wireless traffic [3]. However, the proposed approaches assume the analysis of the targeted protocol. Therefore, an intruder complying with the protocol cannot be detected by the system as being an intruder, especially if he is listening passively. Another approach could be to use current consumption analysis. This type of approach would be based on the electricity power consumption of the intrude device. However the intruder might have a consumption much lower than that of the network considered which would make it undetectable by current analysis.

Unlike network's analysis based approaches, our method does not rely on any a priori knowledge on the network. It also does not rely on electricity power consumption but instead on Electro Magnetic (EM) side-channel signals which contain much more information. To that end, we propose two methods. The first one is based on a hybrid approach using a recurrent neural network and the second one is a full AI approach. In [4,5], the EM signal is used to detect abnormal behavior on a chip like, for example, the execution of a malicious program. The proposed methods are focused on local analysis with a EM probe . Moreover, whereas the former detects the intruder's activity effect on a single device's EM side-channel signal, we aim at detecting the intruder EM emanations in an aggregate of possibly several devices emanations. Thus, our methods can find applications in many fields such as networked (or isolated) computer systems, control and data acquisition systems, the Internet of Things, wired and wireless networks. We detail theses approaches in section 2, present some numerical experiments in section 3 and conclude in section 4.

2 Proposed methods

2.1 Side-channel signals sensing

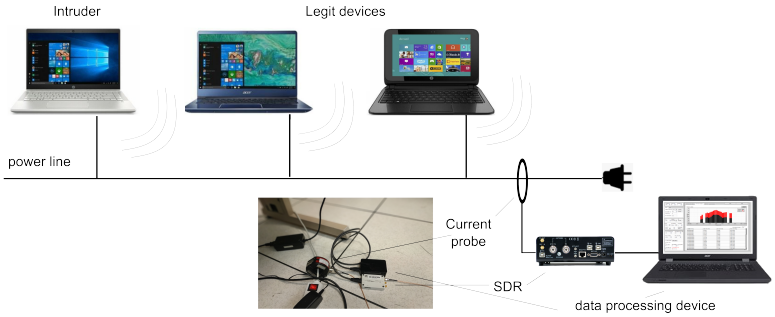


Fig. 1. Block diagram of the proposed method: the network is made of legit devices and an intruder. All generate EM emanations that couple onto the power lines. The proposed method captures the EM signal with a current probe and SDR system that sends the raw IQ data to a processing data device.

All electronic devices produce EM emanations that not only interfere with radio devices but also compromise the data handled by the information system. A third party may perform a side-channel analysis and recover the original information, hence compromising the system privacy. While pioneering work of the domain focused on analog signals [6], recent studies extend the eavesdropping exploit using an EM side-channel attack to digital signals and embedded circuits [7]. Side channels are used to retrieve the information completely. However, even though the information cannot be retrieved entirely, the EM emanation can embed another type of information such as the device type (computer, monitors, etc.) or activity cycle (sleep mode, idle, working, etc.)

We consider a generic scenario involving n legit devices and one unknown device (see Fig. 1). The EM emanations of both the legit and the unknown devices are sensed though an acquisition system which consists in a receiver and an analog digital converter (ADC). In our scenario, we consider a current probe to recover the EM by conduction. The receiver is a SDR device with its associated filters and ADC. This acquisition system senses the signals that are used for training and monitoring.

Two AI approaches are proposed to detect the unknown device, a hybrid AI approach based on a dictionary of activity pattern and a full AI approach based on an AutoEncoder.

2.2 Sparse modelling of side-channel signals

The first step of the learning phase is to detect and extract from the signals received from the capture system, segments corresponding to periods of *activity* of equipment whose EM emanation were measured. On these segments, the signals are expected to exhibit particular morphologies. These signals are complex-valued. Although the phase is certainly informative, we focus the analysis on the amplitude signal. The previously mentioned segments are then the continuous and non-extendable periods over which the amplitude takes values greater than a given threshold. This threshold constitutes a sensitive parameter, the choice of which will be discussed later. A typical example of amplitude signal is given in Fig. 2.

Activity patterns are extracted, registered w.r.t. their time-wise barycenters as illustrated in Fig 2 and zero-padded so that they have the same length.

These patterns are extracted for the all legit equipments and stacked into a matrix $\mathbf{P}$ of size $n \times m$, n being the number of activity patterns extracted and m their length. We factorize $\mathbf{P}$ into two sparse non-negative matrices: $\mathbf{P} = \mathbf{WD}$ where $\mathbf{W}$ and $\mathbf{D}$ are $n \times p$ and $p \times m$ matrices. We use the method presented in [8] via the python toolbox *NIMFA*[1]. This amounts to decomposing the activity patterns into simpler shared sub-features which are the are the p rows of the matrix $\mathbf{D}$.

In the following, we refer to $\mathbf{D}$ as the dictionary and to $\mathbf{D}$'s rows as the atoms. We note these atoms $\mathbf{d}_1, \cdots, \mathbf{d}_p$. By design, these atoms can be used sparsely to

[1] http://nimfa.biolab.si/index.html, accessed in Febuary 2020

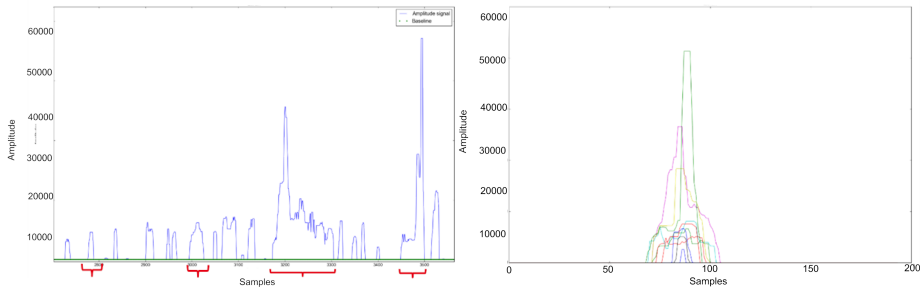


Fig. 2. On the left, amplitude signal: the braces indicate some activity segments. On the right: registered activity patterns.

reconstruct activity patterns. So for any given amplitude signal $\mathbf{x} = [x_1, \cdots, x_T]$ we can compute a sparse convolutive representation of $\mathbf{x}$ as follows:

$$\mathbf{x} = \sum_{i=1}^{p} \mathbf{h}_i * \mathbf{d}_i, \tag{1}$$

$*$ denoting the convolution product and $\mathbf{h}_i$ being a sparse non-negative weights vector of length T associated with the i^{th}. These weights determine the presence and the magnitude of the different atoms at different times in the magnitude signal.

We estimate them by solving an optimisation problem of the form:

$$\min_{\mathbf{h}_1, \cdots, \mathbf{h}_p} \frac{1}{2} \|\mathbf{x} - \sum_{i=1}^{p} \mathbf{h}_i * \mathbf{d}_i\|_2^2 + \sum_{i=1}^{T} w_i \|[\mathbf{h}_1[i], \cdots, \mathbf{h}_p[i]]\|_2, \quad \text{s.t. } \mathbf{h}_i \geq 0, \tag{2}$$

The weights $w_1, \cdots, w_T$ are hyper-parameters set in order to mitigate the l_2 norm induced bias, according to the strategy presented in [9]. Once this estimate has been made, we now have at each time i a weight vector $\mathbf{x}_i = [\mathbf{h}_1[i], \cdots, \mathbf{h}_p[i]]$ which characterizes the contribution of each atom to the magnitude of the amplitude signal at that time. The vector-valued signal $[\mathbf{x}_1, \cdots, \mathbf{x}_T]$ constitutes a new, more structured representation of the initial signal which will be used in the following for sequential modeling.

2.3 Sequential modeling of legit devices activity

At this stage, we have a new vector-valued representation of the amplitude signal. The first step is to quantize the new representation space in order to represent the amplitude signal using a finite vocabulary.

We recall that we have at this stage a sparse vector-valued representation of the training amplitude signal. The previously mentioned partition should only

be based on the vector values of significant l_2 norms, in other words, which are above background noise.

These vector values are thus selected based on a threshold on their l_2 norms. This threshold should be high enough for the noise related vectors to be filtered out. It can be set according to l_2 norms values histogram. The selected vectors are first standardized i.e. the means and standard deviations of each of their coordinates are set to 0 and 1 respectively. The standardized vectors are then clustered using the k-means algorithm. The number of clusters is another important parameter that will be discussed later. The partitions are then defined as being the regions of space closest to each of the centroids of the formed clusters. In a second step, we return to non-standardized vectors and before thresholding. We make a new selection from a lower threshold than the first used in order to retain more complete, although potentially more noisy, information of the amplitude signal processed. Then we recenter and rescale the set of selected vectors using the means and standard deviation previously calculated. We extend each of these vector by adding to it the number of time steps separating it from the next selected vector and assign each vector a partition number.

Let $\mathbf{u}_i$ denote the vector corresponding to the time step t_i and l_i his partition number. Each partition can be interpreted as a particular state of the legit devices activity. In order to capture legit devices activities regularity, we train a Long-Short Term Memory (LSTM) to predict the state or partition number l_k at time t_k, based on the sequence of $\mathbf{u}_i$ observed up to time t_{k-1}. This model is then used to detect anomalies as explained in the next subsection.

2.4 Intrusion analysis

The previous steps are performed using a reference training signal. Given a new amplitude signal, we compute its representation into the partitioned space previously built. It results into a sequence of vectors $\mathbf{v}_1, \cdots, \mathbf{v}_m$, and the corresponding partitions numbers $l_{\mathbf{v}_1}, \cdots, l_{\mathbf{v}_m}$. Let p_i denote the probability estimated by our sequential model that the vector $\mathbf{v}_1$ belongs to the cluster number $l_{\mathbf{v}_i}$ conditionally to the preceding vectors of the sequence:

$$p_i = pr(l_{\mathbf{v}_i} | \mathbf{v}_{i-1}, \cdots, \mathbf{v}_1). \tag{3}$$

The sequence $\mathbf{p} = p_1, \cdots, p_m$ characterizes the compatibility of the new amplitude signal with the model learn:

- values close to 1 indicate a strong fit with the model;
- conversely, values close to 0 indicate outliers from the model point of view.

Hence, Thus, a consistent drop in the values on the p_is would be an indicator of abnormal activity, including the activation of unknown equipment.

2.5 Full AI approach

In parallel to the hybrid AI approach an AutoEncoder based approach is also developed. AutoEncoders have previously been successfully used for anomaly

detection in various contexts [10]. The AutoEncoder learns to reconstruct input windows of fixed size from the uncorrupted signal. Only the signal with the legits computer is used to train the algorithm. Three independents recording of the two legits computers are used: $P_1^A + P_1^B, P_2^A + P_2^B, P_3^A + P_3^B$, with window size of 512 steps on the combined amplitude signal. After, training the same combinations of signals are used to compute the distribution of the reconstruction error. Then, the maximal reconstruction error is used as a threshold to detect corrupted signal.

To detect an anomaly, the suspicious signal is passed to the AutoEncoder and the 512 steps are tagged as malicious if the reconstruction error between the true input and the reconstructed output is above the threshold. A step of the suspicious signal is considered corrupted if half the tags associated to the step are tagged as malicious.

The experiments are conducted on the four combinations of two independents recording of the two legits computers combined with the signal of an intruder: $P_4^A+P_4^B+P^C, P_4^A+P_5^B+P^C, P_5^A+P_4^B+P^C, P_5^A+P_5^B+P^C$. For each combination the intruder is mixed 10 times at different random positions, leading to a total of 40 test samples.

3 Numerical experiments

3.1 Experimental Setup

The experimental setup is defined as follows: a current probe is installed the power strip where all the legit computers are plugged in as depicted in Fig. 1. A Radio Frequency (RF) power amplifier is inserted after the current clamp. The interception system is composed as follows the SDR device is an Ettus B205 mini receiving with a 20 MHz bandwidth to recover the emanations with a fine granularity [11]. The recovered signal is a digitized radio signal of the form 16-bit signed IQ samples. For the intrusion diagnosis, we consider a situation in which there are two legitimate computers (different from those used for dictionary learning) and potentially an *unknown* computer. Traces from the three computers are recorded separately and mixed offline following two scenarios. On the one hand, the legit computers complex traces are simply summed up. On the other hand, the traces are zeroed on different random slots before summation to emulate switching on and off. In both configurations, we use an uncorrupted segment of the synthetic trace to learn a sequential model, and we use it to analyze part of the trace to which we locally added the *unknown* computer's trace (see and Fig. 3).

The training amplitude signal corresponds roughly to 1.6 seconds. The corrupted amplitude signal is one third of the training amplitude signal.

3.2 Results

Dictionary learning The dictionary $\mathbf{D}$ of section 2.2 is computed based on complex traces recorded on the power cables of two computers and their screens

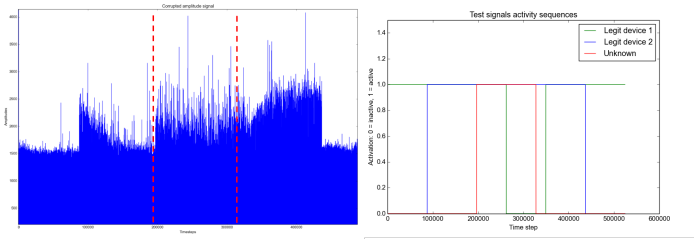


Fig. 3. Corrupted amplitude signal in the second scenario on the left; the intruder is active in the time slot between the dashed lines; on the right the periods of activity of each of the equipment are indicated.

using a current clamp as previously described. Activity segments are extracted using a threshold manually set to 40, based on the histogram of the values of the training amplitude signal. Thus the choice of this parameter can easily be automated. The number of atoms (parameter p) is chosen so that the matrix factorization error is negligible. We set it to 20.

Sequential modeling In order to partition the new representation space previously described, we set the number of clusters to 100. The more clusters there are, the smaller the partitions and the more sensitive the detection is to small disturbances due to the activity of an intruder. However, a high number of clusters requires more data for learning the sequential model. There is therefore a compromise to be found, with a fixed amount of data. This hyperparameter has to be fine tuned accordingly, based on the prediction accuracy of the sequential model on the training data. The vector values are selected using a threshold of 100 for clustering and a threshold of 10 for labelling and sequential model learning. These thresholds are chosen manually based on the histogram of the time step wise l_2 norms of the vector-valued representation of the training amplitude signal obtained by solving problem 2. The later thresholding yields an average compression of roughly 1/10 of the original amplitude signals.

We train a two layers LSTM with a hidden and output layers size of 3. We get a prediction accuracy of more than 80% on a test uncorrupted signal, which confirms that the underlying regularities present in the amplitude signal have been well preserved. This accuracy is empirically stable, regardless of the non deterministic results of the k-means clustering.

Intrusion detection Following the methodology described in Section , we calculate the signal $\mathbf{p}$ for the corrupted traces in the two scenarios. We then calculate in a sliding window of size 100 on the signal $\mathbf{p}$, the percentage of values greater than 0.5. One can see the results obtained in Fig. 4.

In both cases, activation of the unknown equipment results in a detectable increase in the number of implausible transitions in the analyzed sequence. However we observe that the break is less neat in the second scenario. This is simply due to greater statistical variability in the data in this case which makes the

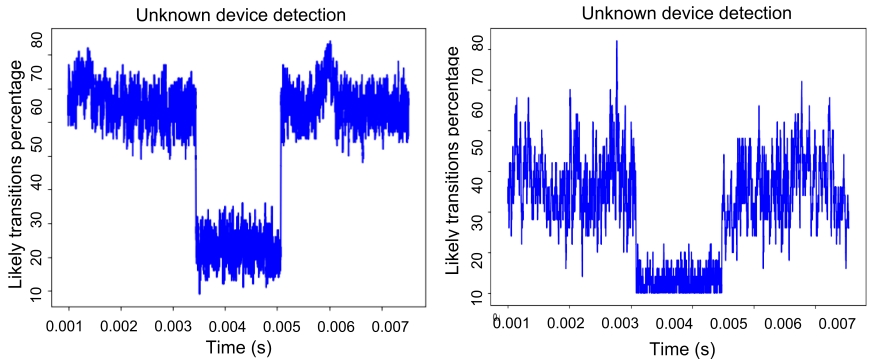


Fig. 4. Intrusion detection in the first scenario on the left and the second scenario on the right. The activation of the unknown equipment causes a detectable rupture thanks to the learned sequential model.

sequential model harder to train. In this case, therefore, more training data should be used. Besides, these results have to be consolidated with a thorough false detection rate analysis.

Full AI results An example of the full AI approach is presented on figure 5. The mixed signal ($P^A + P^B + P^C$) is presented on top of the figure, the activation/deactivation pattern at the bottom. The intruder presence is highlighted in red in the activation/deactivation pattern and the middle plot corresponds to the signals predicted as corrupted. Most of the corrupted signals is captured by the full AI approach.

The accuracy, F1-score and precision are computed for the 40 combinations of legits and intruders signals. The means and standard deviations for the metrics are: accuracy is 0.997 (std. 0.002), F1-score: 0.996 (std. 0.005), precision 0.9995 (std. 0.0008). Theses scores highlight the ability of AI approach to detect corrupted signals through the reconstruction error of a simple AutoEncoder.

4 Conclusion

We presented intrusion detection methods based on side-channel signal analysis. From these signals, recorded only for legit equipment, we learn a vocabulary and an operating syntax using a recurrent neural network and trained a full AI approach based on the reconstruction error between the input signal and the output of an AutoEncoder. The learnt model then allows us to detect deviations from the expected operation, indicating the activity of an unknown equipment. We evaluated this methodology on realistic data. The capture device used is based on inexpensive off-the-shelf components. In a configuration with two legit and one unknown equipment and under two different scenarios, we obtained convincing detection results. The continuation of this work will focus on adapting

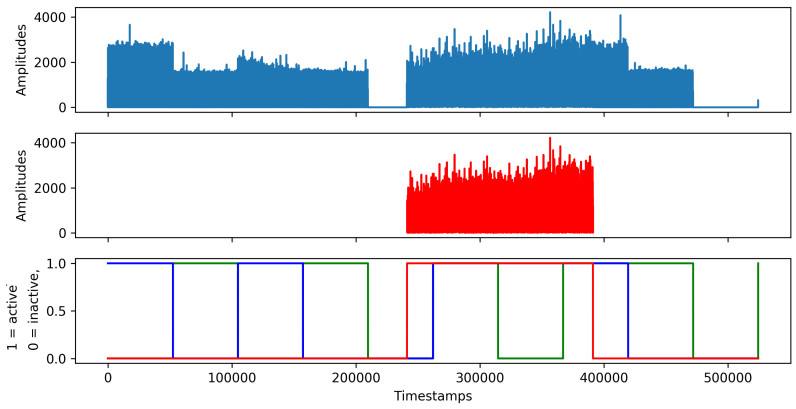


Fig. 5. Detection of corrupted amplitude signal by the full AI approach. Top, mixed signals of legit and corrupted signals. Middle: predicted signal as corrupted. Bottom, activation/deactivation pattern of legit (blue and green lines) computers and of the intruder (red line).

the proposed methodology to the monitoring of an arbitrary number of legit equipment.

References

1. M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2027–2051, thirdquarter 2016.
2. Zouheir Trabelsi and Khaled Shuaib, "Nis04-4: Man in the middle intrusion detection," *IEEE Globecom 2006*, pp. 1–6, 2006.
3. Ahmedur Rahman, C. I. Ezeife, and A. K. Aggarwal, "Wifi miner: An online apriori-infrequent based wireless intrusion system," in *Knowledge Discovery from Sensor Data*, Mohamed Medhat Gaber, Ranga Raju Vatsavai, Olufemi A. Omitaomu, João Gama, Nitesh V. Chawla, and Auroop R. Ganguly, Eds., Berlin, Heidelberg, 2010, pp. 76–93, Springer Berlin Heidelberg.
4. H. A. Khan, N. Sehatbakhsh, L. N. Nguyen, R. L. Callan, A. Yeredor, M. Prvulovic, and A. Zajic, "Idea: Intrusion detection through electromagnetic-signal analysis for critical embedded and cyber-physical systems," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019.
5. Nader Sehatbakhsh, Alireza Nazari, Monjur Alam, Frank Werner, Yuanda Zhu, Alenka Zajic, and Milos Prvulovic, "Remote: Robust external malware detection framework by using electromagnetic signals," *IEEE Transactions on Computers*, vol. 69, no. 3, pp. 312–326, 2019.
6. W. Van Eck, "Electromagnetic radiation from video display units: An eavesdropping risk?," *Computers & Security*, vol. 4, no. 4, pp. 269–286, 1985.

7. Florian Lemarchand, Cyril Marlin, Florent Montreuil, Erwan Nogues, and Maxime Pelcat, "Electro-magnetic side-channel attack through learned denoising and classification," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.
8. Hyunsoo Kim and Haesun Park, "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 05 2007.
9. Emmanuel J. Candès, Michael B. Wakin, and Stephen P. Boyd, "Enhancing sparsity by reweighted 1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 877–905, Dec 2008.
10. Raghavendra Chalapathy and Sanjay Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.
11. M. G. Kuhn, "Compromising Emanations of LCD TV Sets," *IEEE Transactions on Electromagnetic Compatibility*, vol. 55, no. 3, pp. 564–570, 2013.

Recent Trends in Statistical Analysis of Event Logs for Network-Wide Intrusion Detection

Corentin Larroche[1,2(✉)], Johan Mazel[1], and Stephan Clémençon[2]

[1] French National Cybersecurity Agency (ANSSI), Paris, France
first.last@ssi.gouv.fr
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
first.last@telecom-paris.fr

Abstract. Event logs are information-rich and complex data that keep track of the activity taking place in a computer network, and can therefore contain traces of malicious activity when an intrusion happens. However, such traces are scarce and buried under considerable volumes of unrelated information, making the use of event logs for intrusion detection a challenging research topic. We review some recent contributions to that area of research, focusing on the application of statistical analysis to various types of event logs collected over a computer network. Emphasis is put on the formalism used to translate the data into a collection of mathematical objects suited to statistical modelling.

Keywords: Event logs · Intrusion detection · Anomaly detection.

1 Introduction

With their often impressive volume as well as the abundant and valuable information they contain, event logs appear as an obvious candidate for big data and statistical learning methods. Among the possible benefits of their automated analysis, intrusion detection raises outstandingly great expectations: what if a model could make sense of this immense wealth of data, thereby gaining a subtle understanding of the normal behavior of a computer network and spotting hints of suspicious activity? Although such a seamless workflow hardly seems realistic in practice, this perspective has motivated a significant amount of research.

Our work aims to review some interesting contributions to the field of statistical analysis of event logs for network-wide intrusion detection that were published in the last ten years. We only consider detection methods which rely on statistical tools (excluding expert and signature-based systems) and look for evidence of malicious behavior in high-level event logs. In particular, fine-grained analysis of the behavior of a host, using for instance system calls or information flow tracking between system-level objects, is out of our scope. In addition to listing relevant contributions, we focus on one specific aspect whose importance we seek to emphasize, namely the transformation that is applied to the raw event logs in order to obtain a collection of mathematical objects suited to statistical modelling. Two main paradigms are identified: aggregation-based methods,

which represent the network as a set of independent entities whose behavior can be reduced to the events in which they appear, and interaction-based methods, which consider each event as an interaction between two or more entities and analyze the high-order relationships that emerge from these interactions. Building upon this dichotomy, we propose a classification of statistical intrusion detection methods based on event logs. To the best of our knowledge, this aspect of the intrusion detection workflow has not been emphasized in related surveys (e.g. [23,6]), which mainly focus on the algorithms used downstream.

The rest of this paper is structured as follows: we formally define the notions of event logs and intrusion detection in section 2, then give an overview of aggregation-based and interaction-based methods in sections 3 and 4, respectively. We conclude with some suggested research directions in section 5.

2 Definitions and Problem Statement

We first give a formal definition of the data and the problem that are considered here, and then introduce the data representation step, which underpins the classification presented in sections 3 and 4.

2.1 Event Logs and Intrusion Detection – Formal Definitions

Consider a computer network, defined as a set of entities of various types (e.g. users, hosts). An event is an interaction between two or more entities, associated with a timestamp, an event type and a dictionary containing additional information. In practice, this definition encompasses various data sources: authentications can for instance be represented as interactions between users and hosts, with additional information such as the authentication package used. Likewise, a NetFlow record can be seen as an interaction between two hosts, further characterized by the protocol used, the number of packets exchanged, etc.

Given a sequence of events, intrusion detection can be broadly defined as looking for a subset of this sequence corresponding to malicious activity. Note that the sequence can either be observed as a stream or readily available in full. Since finding the exact subset of malicious events is a rather unrealistic goal in practice, detection algorithms mostly aim to extract a collection of suspicious events (or event sets), preferably ranked by their probability of being malicious. The following assumptions can typically be made to better specify the problem:

Assumption 1 *Malicious events are scarce compared to benign ones.*

Assumption 2 *Malicious event sets are distinguishable from benign ones.*

Assumption 3 *Malicious events are connected with each other with respect to the entities they involve.*

This last assumption reflects the idea that an intruder typically uses already compromised entities (such as hosts and user credentials) to propagate further

into the network. Thus, a new event resulting from the intruder's actions has to involve at least one entity that already appears in a previous malicious event. Under these assumptions, intrusion detection can be phrased as a case of anomaly detection: given a high volume of data mostly reflecting normal (usual) activity, the goal is to find a small but cohesive subset that deviates from the norm.

2.2 Data Representation – A Crucial Step

At this point, it should be noted that event logs are peculiarly complex data. Three important aspects stand out: first, time is a fundamental dimension here since the moment when an event appears, as well as the events that precede it, significantly matter in deciding whether it is normal or anomalous. Secondly, the notion of normality of an event is tightly related to the likelihood of the involved entities being associated with each other, and events involving some common entities may actually be disseminated byproducts of a single deliberate sequence of actions (as per assumption 3). Therefore, this combinatorial dimension has to be taken into account. Finally, entities and events can be of several types with different semantics (e.g. users and hosts, authentications and process creations), and this heterogeneity adds another layer of complexity to the data.

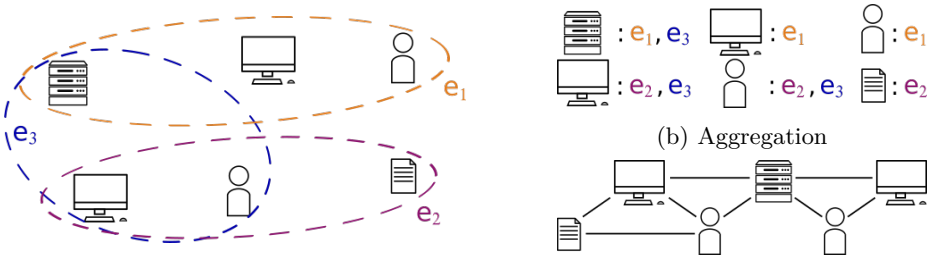
These specific characteristics make it nontrivial to apply standard anomaly detection algorithms to event logs. Indeed, even though such algorithms exist for various kinds of data (e.g. vectors in Euclidean spaces [7], discrete sequences [8], graphs [2]), none of these perfectly fits the complex nature of the input considered here. An intermediary representation is thus needed to translate the logs into a collection of simpler mathematical objects, while preserving enough information to enable detection of malicious activity. Due to its critical importance, this representation is the main criterion we use to categorize the contributions that are reviewed here. Two central paradigms are identified: the first one relies on aggregation, essentially treating entities as mutually independent and summarizing the events in which they are involved to model their behavior. In contrast, the second paradigm attempts to preserve the relational nature of event logs by directly modelling how entities interact with each other. These two paradigms are illustrated in figure 1 and described in more detail in the next two sections.

3 Divide and Conquer – Aggregation-Based Approach

We begin with the aggregation-based paradigm, first explaining the main intuition underlying it, and then exploring its practical applications in more detail.

3.1 General Definition

The fundamental idea behind aggregation-based models is that the set of events involving a given entity can be understood as a history of this entity's behavior. Therefore, summarizing this set into a mathematical object enables comparison of an entity's activity during a given time window with, for instance, its own



(a) Raw events (b) Aggregation (c) Interaction

Fig. 1: Illustration of the two representation paradigms: given events viewed as interactions between entities (a), aggregation-based models (b) represent each entity as the set of events involving it, while interaction-based models (c) consider high-order relationships between entities.

past activity or that of presumably similar entities. Such an object should be carefully defined to preserve evidence of malicious behavior when it exists, while being simple enough to be fed to a standard anomaly detection algorithm.

The anomaly detection language introduced in [18] can be taken as a general framework for aggregation-based methods. In particular, it introduces the notions of *extent* and *baseline*. The former is defined as the conjunction of an entity or group of entities (*entity extent*) and a time period (*temporal extent*), whereas the latter sets the scope of the comparison: an entity's behavior can be compared with the behavior of other entities in the same temporal extent (*cross-sectional baseline*), with the behavior of the same entity in other temporal extents (*longitudinal baseline*) or with both (*simultaneous baseline*). Together, these two notions are the high-level inputs of the anomaly detection process.

3.2 Usual Aggregation Keys and Mathematical Objects

In order to instantiate the general idea of aggregation-based modelling, the first important step is to define how to aggregate events (i.e. the entity extent to use). We now describe some widely used aggregation keys, along with the usually associated mathematical objects (see table 1 for a summary).

First of all, a common intrusion detection scheme consists in monitoring the activity of each user in order to spot compromised accounts. This amounts to aggregating events by user, and the aggregated events can then be summarized into a feature vector, with features often made out of event counts [10,14,29]. In that case, the activity of a given user during a period of time is simply characterized by the number of actions of each possible type taken by this user. Emphasis can alternatively be put on the order in which these actions happen, thus representing the set of events as a discrete sequence [25,5]. More complex sequence-based methods can also leverage the inter-arrival times of events [32]. Finally, some authors picture event sets as graphs, which can for instance be defined by focusing on a user's authentications: each vertex represents a host,

Table 1: Aggregation-based methods, grouped by aggregation key and mathematical object.

Agg. key \ Object	Scalar or vector	Sequence	Graph
User	[10,12,14,29]	[25,5,32]	[16,24]
Host	[26,31,4]	[11,21]	-
Entity pair	[20,27,17]	[1]	-

and directed edges between vertices stand for authentications from source hosts to destination hosts. The obtained graph can then be transformed into a feature vector [12] or directly analyzed using graph-oriented tools [24]. Another kind of graph can also be built by associating a vertex with each event and adding links between events sharing some specific traits [16].

An alternative to user-based aggregation is its host-based counterpart, which aims to model the usage pattern of a host rather than the behavior of a user. Abstracting the data into vectors of event counts remains a popular method in this context [26,31], but other ideas have also been proposed. In particular, graph-based features can be derived for hosts as well, using network traffic metadata to build a host communication graph [4]. Communications between hosts can also be treated from the point of view of a single host as a discrete sequence of sources or destinations [11,21].

Finally, a more fine-grained understanding of the activity contained in the logs can be achieved by aggregating events by user-host or host-host pair. This is a first step towards handling the combinatorial nature of events, while still considering each pair separately from the others. The usual multivariate count-based approach is still relevant here [27], and discrete sequences can be used as well [1]. However, because the mere existence of an interaction between two entities already carries significant information, it is possible to use even simpler mathematical objects (e.g. a single interaction count [20] or a boolean [17]).

4 All Intertwined – Interaction-Based Approach

We now turn to the interaction-based paradigm, once again starting with a general explanation before reviewing its applications.

4.1 General Definition

Unlike aggregation-based methods, which essentially break the complex entanglement of events into a collection of subsets that are then treated as independent, the interaction-based approach attempts to capture the intricate patterns of association between entities. In particular, the relationships between different interactions (e.g. events that involve some common entities) are explicitly taken into account, possibly unveiling high order dependencies between entities and cohesive sets of anomalous events (as defined in assumption 3). This makes

Table 2: Interaction-based methods and the objects they rely upon.

Object	**Graph**	**Bipartite graph**	**Knowledge graph**	**Hypergraph**
References	[13]	[9,30,19,22]	[15]	[28,3]

interaction-based modelling different from aggregation by entity pair, which only compares the activity of a given pair with its own past activity or with that of other pairs, without looking for dependencies between them.

This paradigm thus implies a more global view of the activity happening inside a computer network, and as a consequence, it can lead to more complex objects and models. The next section reviews some of these (see table 2 for a summary of interaction-based methods and associated mathematical objects).

4.2 Existing Models and Underlying Objects

Unsurprisingly enough, graphs are a popular tool for abstracting event logs in an interaction-based formalism. More specifically, events can typically be translated to user-item interactions, which in turn yields a bipartite user-item interaction graph. Practical examples include bipartite authentication graphs, whose vertices are users and hosts, with each edge indicating an authentication of a user on a host. Communications between hosts can also be understood as a graph.

Detection models in this setting can for instance rely on the community structure of the graph, marking an unusually high number of inter-community edges as a network-wide anomaly [19]. Alternatively, a similarity measure between vertices can be designed using the structure of the graph, enabling the search for outliers [9]. Suspicious entities can then be identified, providing more fine-grained information than a global model. However, even more targeted alerts can be obtained by building a statistical model of historical interactions, which can then be used to predict how likely two given entities are to interact with one another. This can typically be done through collaborative filtering, a method relying on the intuition that if user A usually interacts with the same items as user B, then A should be more likely to interact with a new item if B already has. Having built such a model, one can then assign probabilities to new interactions, raising alerts on highly improbable ones [30]. This method can be further enriched by integrating a temporal aspect [13,22] or using attributes of the entities as additional input [22].

Graphs, however, can only handle dyadic interactions. This can be a problem when working with events that involve more than two entities (e.g. a user running an executable on a host). A way to circumvent this limitation is to depict the events themselves as vertices which are linked to the entities they involve, resulting in a heterogeneous graph (also called knowledge graph) carrying more fine-grained information. Looking for outliers in this graph can then reveal suspicious events [15]. Alternatively, events can be explicitly described as polyadic interactions (i.e. hyperedges in a hypergraph) and analyzed as such with dedicated tools, including pattern mining [28] and representation learning [3].

5 Conclusion – Research Directions

Having reviewed the main trends in the literature, we now conclude with some possible research directions. As for aggregation-based methods, one of the main challenges lies in correlating anomalies detected for different entities (based on assumption 3). Indeed, having separated the events into independent subsets, it is then nontrivial to assess whether different subsets that appear anomalous on their own can be traced back to a single trail of malicious activity. Interesting ideas regarding this problem can be found in [20,26,17]. Another area for improvement is the use of entities' roles to build better models: instead of only using past activity of a single entity to determine whether its current behavior is anomalous, it can be interesting to include events related to presumably similar entities. However, defining groups of similar entities, especially with limited background information about the network, is challenging (see [10] for a purely behavioral method and [14] for an approach relying on organisational roles).

Finally, possible research directions in the interaction-based paradigm include more accurate modelling of temporal dynamics: even though some recent contributions [13,22] have tackled this issue, the highly complex temporal variations of activity in computer networks (including periodic automated behavior, seasonal patterns and long-term drift) still provide room for improvement. The heterogeneity of interactions also calls for richer models, as most published methods only consider a single kind of interaction, thus ignoring the relationships that could exist between events of different types. Despite these limitations, interaction-based modelling seems to be a promising approach, and we think it deserves increased attention from the research community.

References

1. Adilova, L., Natious, L., Chen, S., Thonnard, O., Kamp, M.: System misuse detection via informed behavior clustering and modeling. In: DSN-W (2019)
2. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Min. Knowl. Discov. **29**(3), 626–688 (2015)
3. Amin, M.R., Garg, P., Coskun, B.: Cadence: Conditional anomaly detection for events using noise-contrastive estimation. In: AISec (2019)
4. Bohara, A., Noureddine, M.A., Fawaz, A., Sanders, W.H.: An unsupervised multi-detector approach for identifying malicious lateral movement. In: SRDS (2017)
5. Brown, A., Tuor, A., Hutchinson, B., Nichols, N.: Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In: MLCS (2018)
6. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surveys Tuts. **18**(2), 1153–1176 (2015)
7. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM comput. surv. **41**(3), 1–58 (2009)
8. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection for discrete sequences: A survey. IEEE Trans. Knowl. Data Eng. **24**(5), 823–839 (2010)
9. Chen, Y., Malin, B.: Detection of anomalous insiders in collaborative environments via relational analysis of access logs. In: CODASPY (2011)

10. Eldardiry, H., Bart, E., Liu, J., Hanley, J., Price, B., Brdiczka, O.: Multi-domain information fusion for insider threat detection. In: S&P Workshops (2013)
11. Heard, N., Rubin-Delanchy, P.: Network-wide anomaly detection via the dirichlet process. In: ISI (2016)
12. Kent, A.D., Liebrock, L.M., Neil, J.C.: Authentication graphs: Analyzing user behavior within an enterprise network. Comput. Secur. **48**, 150–166 (2015)
13. Lee, W., McCormick, T.H., Neil, J., Sodja, C.: Anomaly detection in large scale networks with latent space models. arXiv preprint arXiv:1911.05522 (2019)
14. Legg, P.A., Buckley, O., Goldsmith, M., Creese, S.: Automated insider threat detection system using user and role-based profile assessment. IEEE Syst. J. **11**(2), 503–512 (2015)
15. Leichtnam, L., Totel, E., Prigent, N., Mé, L.: Sec2graph: Network attack detection based on novelty detection on graph structured data. In: DIMVA (2020)
16. Liu, F., Wen, Y., Zhang, D., Jiang, X., Xing, X., Meng, D.: Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In: CCS (2019)
17. Liu, Q., Stokes, J.W., Mead, R., Burrell, T., Hellen, I., Lambert, J., Marochko, A., Cui, W.: Latte: Large-scale lateral movement detection. In: MILCOM (2018)
18. Memory, A., Goldberg, H.G., Senator, T.E.: Context-aware insider threat detection. In: AAAI Workshops (2013)
19. Moriano, P., Pendleton, J., Rich, S., Camp, L.J.: Insider threat event detection in user-system interactions. In: MIST (2017)
20. Neil, J., Hash, C., Brugh, A., Fisk, M., Storlie, C.B.: Scan statistics for the online detection of locally anomalous subgraphs. Technometrics **55**(4), 403–414 (2013)
21. Passino, F.S., Heard, N.A.: Modelling dynamic network evolution as a pitman-yor process. Found. Data Sci. **1**(3), 293 (2019)
22. Passino, F.S., Turcotte, M.J., Heard, N.A.: Graph link prediction in computer networks using poisson matrix factorisation. arXiv preprint arXiv:2001.09456 (2020)
23. Patcha, A., Park, J.M.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. Comput. netw. **51**(12), 3448–3470 (2007)
24. Powell, B.A.: Detecting malicious logins as graph anomalies. J. Inf. Secur. Appl. **54**, 102557 (2020)
25. Rashid, T., Agrafiotis, I., Nurse, J.R.: A new take on detecting insider threats: exploring the use of hidden markov models. In: MIST (2016)
26. Sexton, J., Storlie, C., Neil, J.: Attack chain detection. Stat. Anal. Data Min. **8**(5-6), 353–363 (2015)
27. Shashanka, M., Shen, M.Y., Wang, J.: User and entity behavior analytics for enterprise security. In: BigData (2016)
28. Siadati, H., Memon, N.: Detecting structurally anomalous logins within enterprise networks. In: CCS (2017)
29. Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., Robinson, S.: Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In: AAAI Workshops (2017)
30. Turcotte, M., Moore, J., Heard, N., McPhall, A.: Poisson factorization for peer-based anomaly detection. In: ISI (2016)
31. Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., Li, K.: ai^2: training a big data machine to defend. In: BigDataSecurity (2016)
32. Yuan, S., Zheng, P., Wu, X., Li, Q.: Insider threat detection via hierarchical neural temporal point processes. In: BigData (2019)

Unsupervised methodology to detect anomalies in network communications

Romain Burgot, Alric Gaurier, Louis Hulot, and Léo Isaac-Dognin

Capgemini Invent France, romain.burgot@capgemini.com, alric.gaurier@capgemini.com, louis.hulot@capgemini.com, leo.isaac-dognin@capgemini.com

Abstract. Security Information and Event Management systems are increasingly challenged by growing threats. Companies chances of experiencing a breach in the next two years have risen to 30% in 2019 [12]. Furthermore, the number of anomalies detected by classical SIEM systems is often too high to be efficiently analyzed by security operations center analysts, and include significant proportions of false positives - according to one study, only 26% of received alerts are related to actual breaches [14]. Moreover, a classical SIEM provides little help to cyber analysts in understanding the context, interconnection, or significance of alerts, such as by regrouping events in clusters of linked abnormal behavior. Therefore, IT networks surveillance including artificial intelligence (AI) is an important field of research. AI, in the form of machine learning algorithms, can facilitate incident detection, analysis, and prioritization, by regrouping linked abnormal behaviors in order to give a better understanding of what is happening in the network.

In this paper we propose a methodology to contextualize network communications with aggregation-based metrics, detect anomalies with autoencoders, qualify their significance, and group anomalies in a way that is understandable and useful to cybersecurity analysts in their day-to-day.

This methodology is applied to the CICIDS2017 dataset which includes several types of attacks (Brute Force, Web Attacks, Dos/DDos, Port Scans, Infiltration, etc) carried out with a range of tools (Patator, Injection, Infected Cool Disk, Botnet ARES, etc). Thanks to the approach presented 11 out of 15 breaches were automatically detected on at least one of the machines compromised during the breach. Contextualization and explanation outcomes are shown and interpreted with the help of corresponding anomaly clusters.

1 Introduction

Cyber threats are evolving and intensifying. Indeed, hackers and threats are diverse and well financed. They are endangering all companies and institutions: in 2018 20% of companies reported losses of more than $50 millions due to cyber-attacks [13]. This calls into question the tools and mainstream methodologies used to prevent security breaches and exploits. Cybersecurity solutions based on threat dictionaries are proving their limits because they are not able to detect new kinds of attacks, by definition. The lack of insights provided by those solutions is a problem, because cyber analysts spend an increasing time - up until 206 days 2019 – in qualifying alerts [12]. AI is a way to enhance Security Operations Centers by analyzing more data in the same amount of time, reducing false alerts and guiding security analysts with enhanced insights on detected anomalies.

Cybersecurity and AI are fields of research that find more and more mutual subjects. Intrusion detection is a growing field of application for machine learning researchers, for instance :

- Unsupervised methodologies with a specific feature engineering using recurrent neural network [6] or Isolation Forest [4] [8] has been studied
- Supervised approach with neural network [2]
- Anti-phishing has been studied by deeplearning researchers [5]
- KPIs of web application are managed with variational autoencoder to ensure undisrupted business [10]

Besides, some researchers in graph theory construct models rooted in reality by using both structural and temporal metrics (studying the dynamic of network communications) designed to identify the role behind an IP in a network [9]. Moreover, to accelerate the growth of applications in this field, some researchers deliver datasets of network activity as close as possible to real-life threats [7], while others review the pros and cons of each publicly available datasets [15].

In this paper, we first propose a methodology to structure network communications and create metrics based on cybersecurity and data science technical knowledge. This structured data is then used to automatically create a model of network expected behavior using an autoencoder, a type of artificial neural network. Anomalies are then identified, qualified, and finally clustered into related events to be analyzed by a security analyst by giving them the clearest possible view of what happened.

The methodology we developed is based on packet captures, or alternatively, standards bi or uni-directional flows. The packet capture has to be located at relevant points of the network, but the data we use is only a summary of these captures, commonly known as netflows.

We believe that data collection and its pre-processing is an important part of any relevant intrusion detection systeme (IDS) and that our work can be enhanced with deep packet inspection or the use of tools such as Zeek. We also consider that a reasonable understanding of the process of collecting and pre-processing data is required to make use of said data. Therefore, we chose

a recent and labeled dataset of packet captures. Nevertheless, several datasets cited in [15] could be used in future work in order to test the performance of our methodology with more data and against other types of malicious behavior.

We further enhance this summary of communications (netflows), with aggregation based metrics. We then train a model that uses only a small part of the information available in all the fields to replicate all the fields of a record. These records are compared with the expected records in order to detect anomalies. Therefore, the detection is based on the expected behavior of the network that has been learned by the model and not on previous known attacks dictionaries.

Finally, an alert explainability methodology is presented along with a clustering of anomalies. Using this developed methodology, a cyber analyst only receives interesting alerts gathering thousands or millions of communications with information to make sense of what is happening on the network.

2 Data

Our approach is based on the analysis of netflow data. The netflow format is a summary of packets exchanged between IP source/destination, port source/ destination and protocol on 5 minutes time windows. For such a 6-tuple, several flow metrics (number of packets, size in bytes, etc) are computed to lead to the netflow format.

Our approach has been tested on the CICIDS 2017 dataset [1]. This dataset is composed of network data recorded in a small network, communicating with hundreds of computers over the course of 5 days. During this week a subset of computers and servers are attacked using several approaches.

The dataset is considered as semi-labelled. On one hand it is labelled with the kind of attack used, against which IP address and at what time. On the other hand the environment of an attacked IP address (like Firewall or other) is not labelled even if it is also facing an abnormal behavior without being directly attacked. The results presented in this paper are based on netflow metrics and custom metrics computed on the netflow of the CICIDS 2017.

3 Architecture of development

Our implementation has been realized with Spark, Tensorflow, ElasticSearch, Kibana and Linkurious. The power of Spark is used to compute netflow metrics and custom metrics. TensorFlow and Spark H20 is used for the different algorithms: anomaly detection, identification of an IP role, clustering of anomalies. Kibana is used to create dashboards for security analysts to get insights from the algorithms results, whereas Linkurious is used to visualize the network as a graph and to easily discover which part of the network is being attacked.

This architecture may not be essential for the relatively small dataset used to test our method, has the advantage of ensuring scalability. Our system can

[1] https://www.unb.ca/cic/datasets/ids-2017.html

easily handle large volumes by adding workers to the Spark cluster, while H2O enables us to use more advanced machine learning or statistical methodologies than are available in Spark.

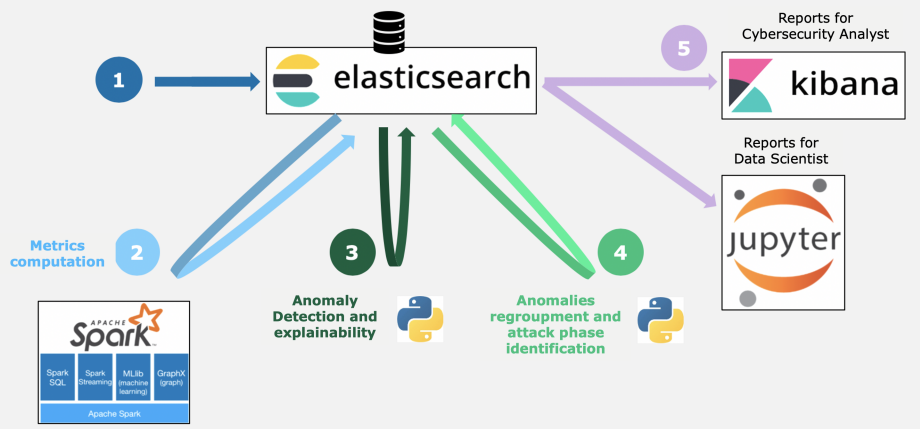


Fig. 1: Architecture of development

4 Feature engineering

We have computed 126 different features. These metrics or features are used to contextualize a netflow row, thus adding more information to the row than its initial scope (the 6-tuple). Those features can be split in 3 categories that are described below.

4.1 Features designed by cybersecurity experts

The approach and work presented in this paper was defined thanks to a partnership between experts in cybersecurity and the authors of this paper. The aim of this partnership was to merge two different points of view and technical approaches: the data science expertise and cybersecurity expertise were brought by separate teams that were able to challenge each other at each stage of the work.

The cybersecurity expertise enabled the team to conceive specific metrics. These metrics were designed to identify some of the behavior that are observed and characterize different phases of an attack (reconnaissance, pivoting, elevating privilege, etc), thus they will give insights to the security analyst about what happened or what was the intention of the attacker.

As an example, for every IP we computed the number of packets sent to any IP to port 22 (SSH) with a flag SYN during the last 5 minutes. If this metric deviates from the expected range, the IP is likely be facing a SYN port scan, indicating an intention to find a device vulnerable to remote exploitation.

4.2 Features from data scientists

Some other features were created based on aggregations over several key values of our logs (IP source, IP dest, IP source and dest) and rolling windows of several size (5 minutes, 30 minutes, 1 hour, 2 hours). As an example, we computed the number of bytes exchanged between 2 IP addresses during the last hour divided by the number of packets exchanged between those IP addresses over the course of 1 hour. While the value of the features used may not all feel intuitive or straightforward to SOC security analysts, they help the autoencoder represent the expected behavior of an IP address. Most of the features were developed with the help of cybersecurity experts that collaborated on the approach developed in this paper.

4.3 Feature from graph theory approach

One of the features is the output of a clustering algorithm. Indeed, it is possible to qualify a machine role thanks to features based on graph-theory [9]. This role is important to assess the expected behavior of a machine. For example a computer does not face the same kind of flow than a firewall.

The information of the role of a device in an IT park may be in a configuration management database for example, but it might not be up-to-date. This approach can replace such a database. We vectorize our logs with Latapy et al. [9] features and we use hierarchical linkage on our data with ward variance minimization algorithm to compute the machine clusters with a hierarchical clustering algorithm. Each cluster is then interpreted as a kind of machine and this information is used to contextualize a netflow row like any other features.

5 Modeling

A type of neural network, an autoencoder, is used to model the expected behavior of the computers. The expected behavior is compared with the actual behavior in order to detect anomalies. An autoencoder compresses and creates encoded data from its input and then decodes the encoded data into an output. When the autoencoder is well trained the output and the input of the neural network are similar. This means that the neural network is able to compress the input and to decompress it, getting an output close to the input. When the network is well trained on the 'normal' behavior of computers and servers within the network, the difference between normal logs and the output of the model is close to zero while the difference between logs during breaches or abnormal events and the output of the model is significant. It is possible to study this level of similarity between logs and expected logs during a period of time with no abnormal behavior from the computers and servers. So a threshold on this level of similarity is set in order to split the anomalies from the normal logs.

6 Results

One of the aims of this paper is to detect attacks on an IT system based on netflow data. With the 126 features, the model detects 11 attacks over 15 present in the CICIDS 2017 dataset. An attack is said detected if the model creates an alert on at least one IP victim or attacker during the attack.

Table 1: Performance on CIC IDS 2017

Attack Type	**Technology used**	**Performance**
Brute Force	SSH-Patator	Detected
	FTP-Patator	Detected
Dos / DDos	Slowloris	Detected
	GoldenEye	Detected
	Slowhttptest	Detected
	Hulk	Detected
	LOIT	Detected
HeartBleed	-	Detected
Web Attack	Brute Force	Not Detected
	XSS Injection	Not Detected
	SQL Injection	Not Detected
Infiltration	DropBox	Detected
	Cool Disk	Detected
Port Scan	Nmap	Detected
Botnet	ARES	Not Detected

We found only one other approach using an unsupervised model on the CICIDS 2017 dataset in the literature, claiming to quantify its performance on the entire dataset with an AUC of 0.84 [6]. Insufficient performance quantification per attack can lead to misinterpretation of the results because the different attacks do not contain the same amount of netflow rows. So if an attack over fifteen represents 50% of the logs labeled as an attack, and that the model performs well on this particular attack, then the performance will be higher than 50% whereas it is detecting only one type of attack over fifteen.

Therefore, we chose to not provide any measures of performance for our implementation. Choosing relevant measures of performance could be the subject of a paper in itself :

- At which spatial/temporal granularity is the measure relevant?
- How can one ensure that performance measures are not biased towards attack types that produce more communications?

In any case, we cannot claim to offer the best implementation. Our aim is to present a general methodology that provides insights of sufficient value to a security analyst to save them time in their day to day activities. In doing so, our

approach ought to ensure that it would not produce even more logs requiring analysis (which would simply increase the resulting analyst workload).

7 Post processing

7.1 Explainability

Autoencoders are highly interpretable by design. Indeed, autoencoder algorithms detect anomalies by comparing outputs and inputs. The quality of the reconstruction is quantified by a reconstruction error that is linked to the anomaly score. In the case of our topic - detection of abnormal logs - a more accurate reconstruction indicates that logs are 'close' to the normal behavior. Explainability is possible by studying the model's reconstruction error: input features that have been badly reconstructed can be interpreted as a cause of the anomaly. In this way, our model detects abnormal logs and also points to the features that led to an alert, especially if the metrics constructed with cybersecurity experts are part of the reconstruction loss. Indeed, these metrics are both highly interpretable and significant given that they are specifically designed to identify behavior that is characteristic of malicious behavior.

Table 2: Example of model explanation for Infiltration and Port Scan

Attack Type	**1st Explanation**	**2nd Explanation**
Infiltration (DropBox)	Mean packets upload download ratio per src-dest IP during the last hour	Mean bytes upload download ratio per src-dest IP during the last hour
Port Scan (Nmap)	Packets with no answer to dest IP during the last hour	SYN Flag count

There are obviously alternatives to using autoencoders and their error reconstruction. For example, one could use an Isolation Forest model and tree SHAP values, which may even seem more accurate from a theoretical perspective. Nevertheless reconstruction error from autoencoder has the advantage of being a virtually-free output of the modelling, saving computation workload and providing an easily understandable explanation.

7.2 Clustering anomalies with Non-Negative Matrix Factorization

Another mathematical tool named Non-Negative Matrix Factorization is used to group anomalies that may be linked with each other. This approach can be computed on any duration, it can be performed on a year of detection or an hour. Given a chosen period and anomalies detected within, a weighted matrix

is created with destination IP as rows and source IP as columns. At each node a metric based on the anomaly scores in the chosen period is defined. The factorization is then applied to this matrix in order to get a score of anomalies per IP (source and destination), and then sub-graphs of the network are isolated while computing an anomaly score for each of them. This groups together sub-graphs of the IT network with alerts that are linked.

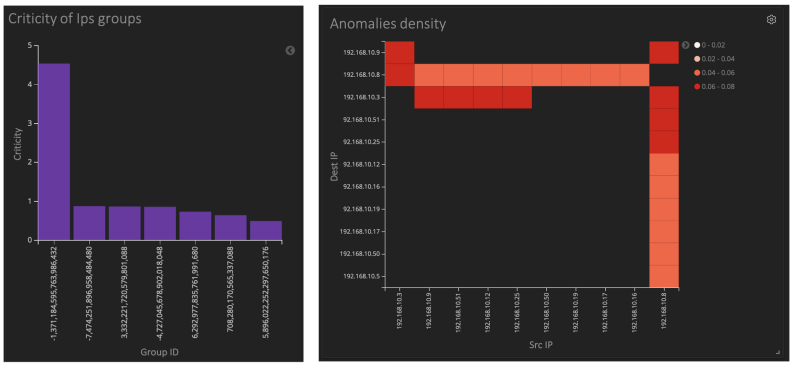


Fig. 2: Clustering of IPs sorted by group criticity

7.3 Visualization

Finally, we believe it is important to equip security analysts with two radically different ways of visualizing results.

The first one focuses on the time sequence of behavior, with dashboards specifically designed to highlight behavior belonging to a given attack phase, e.g. reconnaissance, pivoting, or elevating. We implemented these visualizations, as well as the overall summary dashboard with Kibana. Equipped with these dashboard, the security analyst first obtains a holistic picture of what is happening, then can use the visualization to filter and/or zoom on specific parameters. The analyst can even choose to drill back down to raw logs (netflow).

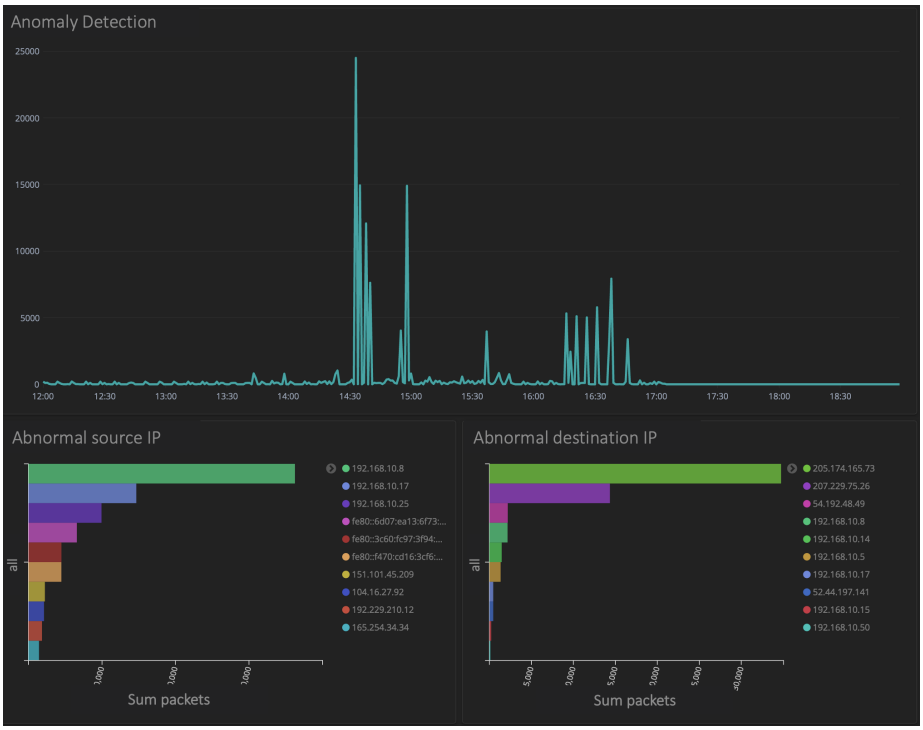


Fig. 3: Anomaly detection dashboard during an infiltration and a port scan

The second way to visualize results, is to use a graph/spatial representation, and replay the sequence. This enables the analyst to get a more detailed view of how the machines grouped within an anomaly cluster have been interacting, and the attack spread across these machines.

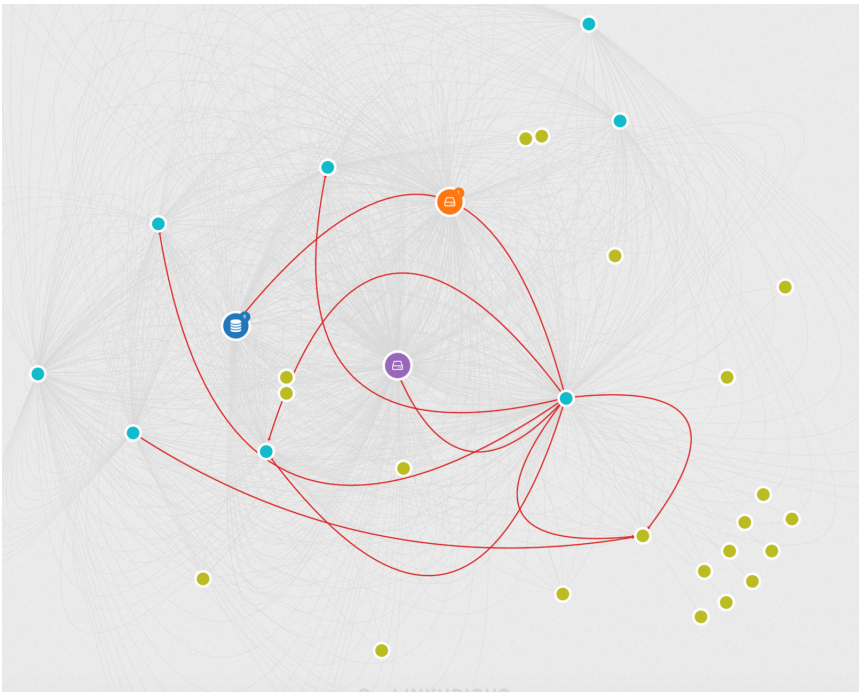


Fig. 4: Graph representation of the attacked network during an infiltration and a port scan

8 Conclusion

In this paper we proposed a methodology to detect anomalies in a network with netflow and deep learning. The methodology includes a way to create context around a log, to detect anomalies in those logs and a way to interpret the anomalies in order to render the big picture of detected potential attacks.

Our approach has been tested on the CICIDS 2017 dataset. We have been able to detect 11 out of the 15 attacks present in the CICIDS 2017 dataset, and provided a synthesis of insights on these attacks through indicators of explainability for each alert.

Netflow data contains a lot of information about flow, it however does not provide information such launch/end of processes, account privilege change, or authentication logs etc. As a next step, the contextualization of an event could be enhanced with information found in other types of data like system logs. The model would then benefit from a multi-channel view and would enhance its representation of the normal network behavior. Moreover, this approach will need to be tested in real, operational conditions and be further challenged by target users.

References

1. DU, Min, LI, Feifei, ZHENG, Guineng, et al. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In : Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017. p. 1285-1298.
2. IERACITANO, Cosimo, ADEEL, Ahsan, GOGATE, Mandar, et al. Statistical analysis driven optimized deep learning system for intrusion detection. In : International Conference on Brain Inspired Cognitive Systems. Springer, Cham, 2018. p. 759-769.
3. LI, Dan, CHEN, Dacheng, SHI, Lei, et al. Multivariate anomaly detection for time series data with generative adversarial networks. arXiv preprint arXiv:1901.04997, 2019.
4. MARTEAU, Pierre-François, SOHEILY-KHAH, Saeid, et BÉCHET, Nicolas. Hybrid Isolation Forest-Application to Intrusion Detection. arXiv preprint arXiv:1705.03800, 2017.
5. NGUYEN, Minh, NGUYEN, Toan, et NGUYEN, Thien Huu. A deep learning model with hierarchical lstms and supervised attention for anti-phishing. arXiv preprint arXiv:1805.01554, 2018.
6. RADFORD, Benjamin J., APOLONIO, Leonardo M., TRIAS, Antonio J., et al. Network traffic anomaly detection using recurrent neural networks. arXiv preprint arXiv:1803.10769, 2018.
7. SHARAFALDIN, Iman, LASHKARI, Arash Habibi, et GHORBANI, Ali A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In : ICISSP. 2018. p. 108-116.
8. SUN, Li, VERSTEEG, Steven, BOZTAS, Serdar, et al. Detecting anomalous user behavior using an extended isolation forest algorithm: an enterprise case study. arXiv preprint arXiv:1609.06676, 2016.
9. VIARD Tiphaine, LATAPY Matthieu: Identifying roles in an IP network with temporal and structural density. In : arXiv:1406.4969
10. XU, Haowen, CHEN, Wenxiao, ZHAO, Nengwen, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In : Proceedings of the 2018 World Wide Web Conference. 2018. p. 187-196.
11. YOUSEFI-AZAR, Mahmood, VARADHARAJAN, Vijay, HAMEY, Len, et al. Autoencoder-based feature learning for cyber security applications. In : 2017 International joint conference on neural networks (IJCNN). IEEE, 2017. p. 3854-3861.
12. Cost of a data breach Report – 2019, IBM
13. Reinventing Cybersecurity with Artificial Intelligence, Capgemini Research Institute, 2019
14. Securing what is now and what is next – CISO cybersecurity reports Series 2020, Cisco, 2020
15. Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes and Andreas Hotho : A Survey of Network-based Intrusion Detection Data Sets. In : arXiv:1903.02460, 2019.

Novelty detection on graph structured data to detect network intrusions

Laetitia Leichtnam[1], Eric Totel[2], Nicolas Prigent[3], and Ludovic Mé[3]

[1] Centrale Supélec, Univ. Rennes, IRISA, France
laetitia.leichtnam@centralesupelec.fr
[2] IMT Atlantique, IRISA, Rennes, France eric.totel@imt-atlantique.fr
[3] LSTI, St-Malo, France neeko@neekotech.fr
[4] INRIA, Univ. Rennes, IRISA, France ludovic.me@inria.fr

Abstract. It is difficult to detect new types of attacks in heterogeneous and scalable networks in time without generating too many false alarms. While supervised anomaly detection techniques are often used to that end, security experts generally do not have labeled datasets. That's why unsupervised learning, that does not require labeled data, should be preferred.
With *sec2graph* [5], we introduced a representation of security events in the form of a graph linking what we defined as *security objects* and proposed a method for anomaly detection based on auto-encoders. This representation allows a rich description of the events that are analyzed. In this paper, we apply our approach to the CICIDS2018 dataset and show that out method outperforms classical event modeling and anomaly detection approaches.

Keywords: Network Intrusion Detection System · Knowledge graph · Autoencoder

1 Introduction

Security Operational Centers (SOC) ensure the collection, correlation, and analysis of security events on the perimeter of the organization they defend. A SOC must detect and analyze internal and external attacks. This mission is hard because security analysts must supervise numerous highly-distributed and heterogeneous systems that use multiple communications protocols.

To address this challenge, anomaly detection techniques have been proposed. These techniques are often build on supervised learning, that requires labeled data during the learning phase. However, security experts often do not have such labeled data sets from their own event logs and data labeling is expensive [2]. As an alternative, an unsupervised anomaly detection technique called "novelty detection", based on auto-encoding, can be used. This technique is generally used when the amount of abnormal data available is insufficient to build explicit models for non-normal classes [8].

In this paper, we first present the model sec2graph, a unified graph representation for heterogeneous network logs. It integrates in a single graph heterogeneous information found in the events and thus allows the construction of a rich vision of the normal situation. We then present a process to efficiently encode this unified graph so that an auto-encoder can learn the normal situation and then detect abnormal activities with two different strategies. We finally use CICIDS2018 dataset [9] to evaluate the ability of the learned model to detect anomalies.

Our contributions are, therefore:

- The definition of two different strategies of detection based on the sec2graph's novelty detector;
- Experimental results on the CICIDS2018 dataset showing that our approach brings a significant improvement over deep anomaly detection algorithms.

This paper is organized as follows: our global approach, named *sec2graph*, is presented in Section 2. This section synthesizes the approach described in [5] and presents a detection strategy based on the auto-encoders. Anomaly detection results and comparative analysis with other methods on the CICIDIS2018 dataset are discussed in Section 3. Finally, conclusions are presented in Section 4.

2 The sec2graph approach

Network event logs are used as an input for the whole *sec2graph* process. Section 2.1 explains how we build a graph of security objects from network events; Section 2.2 explains how we encode this graph into vectors able to be handled by an auto-encoder; Section 2.3 explains how anomalies can be detected by the auto-encoder.

2.1 Building Security Object Graphs from Network Events

A log file can be described as a sequence of events resulting from the observation of activity in the network. Each event is made of several fields and some of these fields are particularly relevant to identify links between events. For each type of event, we identify the most relevant fields and create one or several *Security Objects* (SOs). A SO is a set of attributes, each attribute corresponding to a particular event field.

For example, a network connection event leads to four SOs: a source `IP Address` SO, a destination `IP Address` SO, a `Destination Port` SO and finally, the `NetworkConnection` SO itself that regroups attributes corresponding to the fields we identified as less important to create relations between events. For each type of event, we designed a translation into a set of linked SOs. Thus, each event is represented by a subgraph. As example, the SOs and the links created from logs extracted from the Zeek IDS tool [7] conn.log log files are illustrated in the shema of Figure 1.

conn.log	ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto service duration orig_bytes resp_bytes conn_state local_orig local_resp missed_bytes history orig_pkts orig_ip_bytes resp_pkts resp_ip_bytes tunnel_parents

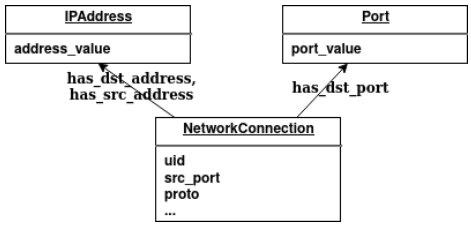


Fig. 1. Building a graph from one log event of the conn.log file

To build the complete graph, we take as an input the network events coming from various log files. From each event, and according to it type, we extract the SOs and the links between them as described before. We then take each SO of the sub-graph. If this SO already exists in the global graph (for instance, a same *IPAddress* was already identified in a previous event), we replace the SO in the new sub-graph by the SO that already exists in the global graph. Therefore, if an event contains an SO that was already found in a previous event, the old sub-graph will be linked to the new sub-graph through this SO.

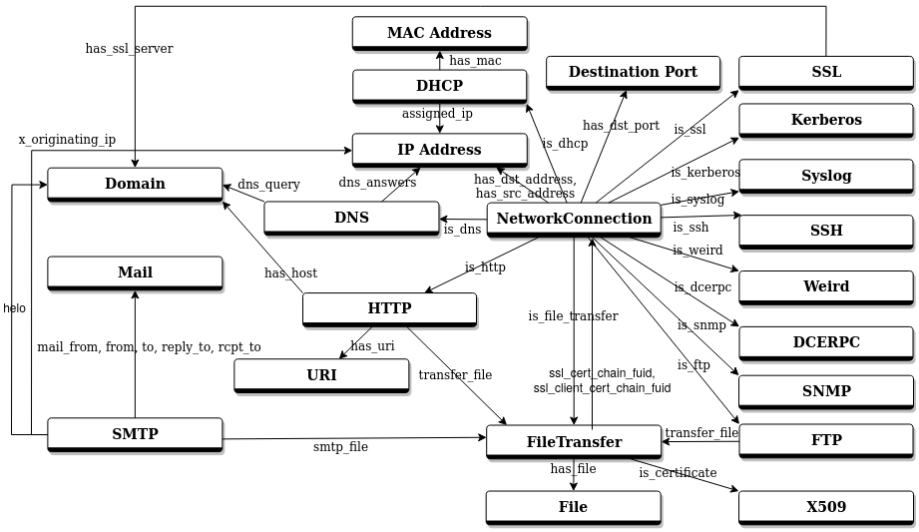


Fig. 2. Complete Security Objects and Relations Model Representation

The graph model in Figure 2 shows the different types of SOs (nodes of the graphs) and their semantic links (edges of the graphs). For clarity reason, we have not represented the attributes of the SOs on this figure. Our model is suited to the pieces of information that are representative of network events. It can also evolve easily according to the needs of the analysts. More details on the building of the SO graph can be found in [5].

2.2 Encoding the graph for machine learning

The second step of sec2graph transforms the graph we computed in a structure that can be processed efficiently by a machine learning algorithm. In our case, the encoding method must encode both the structure of the graph (i.e., the links between the SOs) and the specific information associated with both the nodes (SOs) and the edges (links). Moreover, the result of the encoding should be of reasonable size while it should contain enough information to detect anomalies. Since there does not exist a single best method to encode our graph, we had to design one that was tailored to our specific case.

A given SO can be linked to several events, normal or abnormal. An edge, on the other hand, is only related to the event that led to its construction. Therefore, an anomaly is not carried by the node (an IP address or a port are not abnormal per se) but by the edges that link the SOs together. Consequently, we have chosen to encode our graph by encoding each of its edges. Our representation takes into account the structure of the graph, information contained in SO's attributes and the type of the edges. To this end, we encode an edge as a vector resulting of the concatenation of information on (a) the type of this edge, (b) the attributes of its source node, (c) the attributes of its destination node.

To encode the edge type and the attributes of the nodes, we use common machine learning techniques to transform numerical and categorical data such as the number of packets transferred or the protocol that was used (tcp, udp or icmp) into a binary vector. The basic principle consists in determining a function which associates a category for each value of each attribute, regardless of the type considered. Then, the category is encoded in a binary vector by using the one-hot-encoding techniques. The result of this process is a fixed-dimension binary vector encoding an edge that can now be processed by an auto-encoder.

2.3 Novelty Detection with an Auto-encoder

We use an auto-encoder for novelty detection as already proposed by [1, 3, 6] in the security field where novelty is viewed as an anomaly that may be caused by an attack. An auto-encoder learns a representation (encoding) of a set of pieces of data, typically for dimensional reduction. To do so, it learns a function that sets the outputs of the network to be equal to its inputs. It is made of two parts : an encoder and a decoder. The encoder compresses the input data into a low-dimensional representation, and the decoder generates a representation that is as close as possible to its original input from the reduced encoding.

Anomaly detection methods based on auto-encoders use it to first learn the "normal" behavior by using dataset with benign data. Then, it is assumed that attacks will generate "abnormal" observations that the auto-encoder has never seen and. Therefore, it will not be able to reconstruct the data identically. As a consequence, by computing the difference between the input and the output, we can determine an error, called reconstruction error. If this error is above a determined threshold, an analyst is then able to detect anomalies in a dataset.

While classical approaches seek to identify anomalies linked to events, our approach seeks to identify anomalies related to the links between objects. To refer to the case of anomalies on events, we have considered two strategies called *max* and *mean*. The first one consists in considering as abnormal any event containing at least one link exceeding a detection threshold (*max* strategy). The second (*mean* strategy) consists in computing the mean of the reconstruction errors of all the links associated with an event. If this average exceeds our detection threshold, the event is considered abnormal. In other words, the first strategy suppose that the anomaly is carried by one link, whereas the second strategy assumes that the anomaly is carried by all the links of the same event.

In fact, the max strategy is based on the idea that an anomaly is held on a link of the subgraph representing an attack type event while the mean strategy takes into account the sum of all the anomalies on all the links representing this event. In other words, the later takes into account both strong local anomalies and the sum of weak signals.

3 Implementation and experimental results

This section details our implementation choices and a comparison of the sec2graph approach with other approaches based on anomaly detection.

3.1 Configuration

We choose to use the CICIDS2018 dataset that is made of ten pcap encompassing millions of events. This dataset was generated at the Canadian Cybersecurity Institute and contains ten days of mixed traffic, benign and attacks such as DoS, DDoS, BruteForce, XSS, SQL injection, infiltration, and botnet activities. The CICIDS2018 dataset is the most recent one that models a complete network configuration with a wide variety of components. The data set is also labeled, allowing us to quantify the effectiveness of our method. To generate log files from the capture files, we used the Zeek IDS tool [7] (formerly Bro) that can generate network and application logs such as connections, http communications or file transfers.

In addition to the number of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN), we evaluate our results using the following standard measures: Precision, Detection Rate also known as Recall (DR), True Negative Rate (TNR) and False Positive Rate (FPR). Precision gives the ratio of true abnormal events over events reported as abnormal. DR gives the proportion of events correctly detected as abnormal out of all truly abnormal events. TNR is the proportion of normal events correctly classified among all normal events and FPR the proportion of normal events incorrectly classified among all normal events. Formally, these criteria can be defined as follows:

$$Precision = \frac{TP}{TP+FP};\ DR = \frac{TP}{TP+FN};\ TNR = \frac{TN}{FP+TN};\ FPR = \frac{FP}{FP+TN}$$

3.2 Defining an optimal threshold for detection

In this section, we present the experiments conducted to determine the threshold value to be used for the anomaly score. The analyst sets this threshold value according to his or her supervisory context, lowering the threshold value if it is more important for the analyst not to miss any attacks than to have to eliminate a large number of false positives.

We determine the value of the threshold as follows: first, we consider all the events on the first day of the CICIDS2018 dataset in time windows where there is no attack. With this data, we determine the rate of false positives according to the detection threshold. We obviously want the lowest possible false-positive rate.

The curves in Figure 3 shows the evolution of the FPR as a function of the detection threshold for the strategies *max* (left) and *mean* (right). A threshold of 0.0018 gives us an FPR of 0,46% for the *max* strategy while a threshold of 0.001 gives us an FPR of 0,25% for the *mean* strategy. For both strategies, we see in the figure that the FPR do not decrease significantly when we increase the detection threshold above 0.0018 for the *max* strategy and above 0.001 for the *mean* strategy. In addition, increasing the detection threshold too much can induce a high false negative rate.

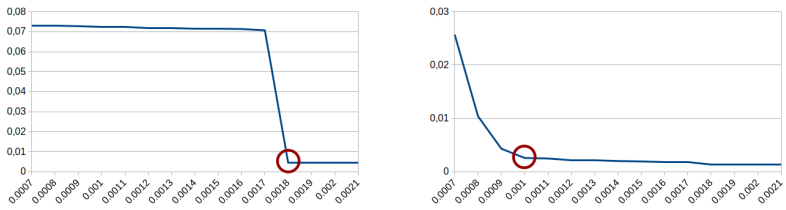


Fig. 3. False Positive Rate (FPR) according to the value of the detection threshold for *max* strategy (left) and *mean* strategy (right)

We conclude that a threshold higher than 0.0018 for the *max* strategy and a threshold higher than 0.001 for the *mean* strategy should be retained.

To validate our approach on how to choose the right detection threshold, we computes the Precision value for several detection threshold values with the *mean* strategy. On the figure 4, it can be seen that for all type of attacks the precision does not significantly increases. On the contrary, for FTP Bruteforce, DoSSlowHTTPTest and Infiltration attacks, it significantly decreases. We, therefore, conclude that the method leading to choose the threshold value of 0.001 is relevant to have a good precision rate. We perform the same kind of approach for the *max* strategy and obtain similar results.

3.3 Comparison with other anomaly detection algorithms on the same dataset

To compare our results with the state of the art, we took the results of a study on intrusion detection using deep learning methods [4] that uses the same data set as we do, the CICIDS2018 data set.

Ferrag *et al.* [4] compares the results of seven supervised and unsupervised classical deep learning algorithms applied to this dataset: Deep Neural Network (DNN), Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN), that are all supervised algorithms as well as Restricted Boltzmann Machine (RBM), Deep Belief Network (DBN), Deep Boltzmann Machine (DBM) and Auto-Encoder (AE) that are unsupervised algorithms.

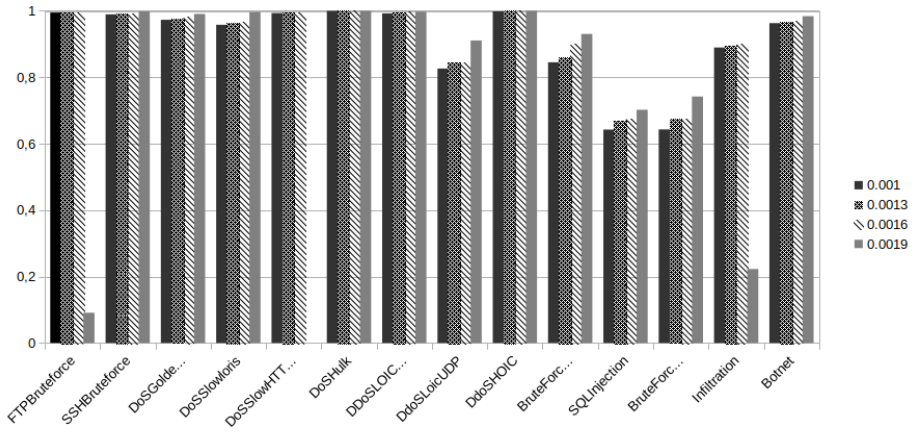


Fig. 4. Values of Precision for the range of variation of the threshold leading to a significant evolution of these values (on sec2graph's *mean* approach).

The table 1 provides a comparison of the results obtained by Ferrag on the CI-CIDS2018 dataset with those of sec2graph using the previously determined optimal value for the detection limit for the two strategies *max* and *mean*.

Table 1. Comparison of True Negative Rate (TNR) and Detection Rate (DR) for each type of attack and for different methods (in %).

	DNN	RNN	CNN	RBM	DBN	DBM	DA	sec2graph (*max*)	sec2graph (*mean*)
TNR (BENIGN)	96.915	98.112	98.914	97.316	98.212	96.215	98.101	99.538	99.743
DR SSH-Bruteforce	100	100	100	100	100	100	100	100	100
DR FTP-BruteForce	100	100	100	100	100	100	100	0.03	100
DR Brute Force-XSS	83.265	92.182	92.101	83.164	92.281	92.103	95.223	99.573	100
DR Brute Force-Web	82.223	91.322	91.002	82.221	91.427	91.254	95.311	100	100
DR SQL Injection	100	100	100	100	100	100	100	100	100
DR DoS-Hulk	93.333	94.912	94.012	91.323	91.712	93.072	92.112	100	100
DR DoS-SlowHTTPTest	94.513	96.123	96.023	93.313	95.273	95.993	94.191	0	100
DR DoS-Slowloris	98.140	98.220	98.120	97.040	97.010	97.112	97.120	100	100
DR DoS-GoldenEye	92.110	98.330	98.221	92.010	97.130	97.421	96.222	100	100
DR DDOS-HOIC	98.640	98.711	98.923	97.541	97.211	97.121	96.551	99.997	100
DR DDOS-LOIC-UDP	97.348	97.118	97.888	96.148	96.122	96.654	96.445	86.932	100
DR DDOS-LOIC-HTTP	97.222	98.122	98.991	96.178	97.612	97.121	97.102	100	100
DR Botnet	96.420	98.101	98.982	96.188	97.221	97.812	97.717	100	100
DR Infiltration	97.518	97.874	97.762	96.411	96.712	96.168	97.818	2.815	100

The values in this table show that the sec2graph approach is clearly superior to the other approaches in terms of FPR, regardless of the strategy chosen. The sec2graph approach with the *medium* strategy offers a 100% detection rate for all types of attacks while having a false positive rate of only 0.25%.

However, we note that the *max* approach gives poor results for the detection of FTP-Bruteforce, DoS-Slow-HTTPTest and Infiltration attacks. It gives an average score for the DDOS-LOIC-UDP attack but also high scores for all other attacks where the detection rate surpasses the one of other learning machine algorithms.

In fact, the *mean* strategy takes into account both strong local anomalies and the sum of weak signals. The *max* strategy could outperforms the *mean* strategy only if a

link with a low anomaly score compensates for the anomaly of a link with a high score. At the time of writing, we have not encountered such cases.

4 Conclusion

We proposed in this paper a graph representation of security events that underlines the relationship between them. We also proposed an unsupervised technique built on an auto-encoder to efficiently detect anomalies on this graph representation with two different strategies to compute the anomaly score. This approach can be applied to any data set without prior data labeling. Using the CICIDS2018 dataset, we showed that the use of graph structures to represent security data coupled with an auto-encoder gives results that are better than common deep anomaly detection methods (supervised and unsupervised).

To further improve our detection results, we plan to use another kind of auto-encoder (LSTM auto-encoder) to take temporal links between events into account to complement to logical links that we already take into account. Another area for improvement is related to the usability and interpretability of results by a security analyst. Here, the idea is to present to the analyst a graphical view of the detected anomalies, based on the SOs graphs that we have defined. We believe that this would help the analyst eliminating false positives or reconstructing global attack scenarios.

References

1. Al-Qatf, M., Lasheng, Y., Al-Habib, M., Al-Sabahi, K.: Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access (2018)
2. Anagnostopoulos, C.: Weakly supervised learning: How to engineer labels for machine learning in cyber-security. Data Science for Cyber-Security (2018)
3. Andresini, G., Appice, A., Di Mauro, N., Loglisci, C., Malerba, D.: Exploiting the auto-encoder residual error for intrusion detection. In: IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (2019)
4. Ferrag, M.A., Maglaras, L., Moschoyiannis, S., Janicke, H.: Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications **50**, 102419 (2020)
5. Leichtnam, L., Totel, E., Prigent, N., Mé, L.: Sec2graph: Network attack detection based on novelty detection on graph structured data. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 238–258. Springer (2020)
6. Min, E., Long, J., Liu, Q., Cui, J., Cai, Z., Ma, J.: SU-IDS: A semi-supervised and unsupervised framework for network intrusion detection. In: International Conference on Cloud Computing and Security (2018)
7. Paxson, V.: Bro: a system for detecting network intruders in real-time. Computer networks (1999)
8. Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Processing (2014)
9. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP (2018)

Luring of adversarial perturbations

Rémi Bernhard[1,2], Pierre-Alain Moëllic[1,2], and Jean-Max Dutertre[3]

[1] CEA Tech, Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne, F-13541 Gardanne, France
[2] CEA, LETI, F-13541 Gardanne, France
{remi.bernhard,pierre-alain.moellic}@cea.fr
[3] Mines Saint-Etienne, Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne, F-13541 Gardanne, France
dutertre@emse.fr

Abstract. Adversarial examples is a major threat against the deployment of machine learning-based systems, more particularly with deep neural network models. Many defenses have been proposed to detect them, render them inoffensive or make the model more robust against them. It is a substantial challenge since adversarial examples set in both white and black-box paradigms with powerful attacks. We propose an innovative deceiving approach against typical black-box transfer threats. Our method is based on the *luring effect* that aims to trick the adversary into choosing false directions to fool the target model. For that purpose, we add a removable neural network that is trained thanks to a loss function acting on the logits sequence order. Our deception-based method only needs to have access to the predictions of the target model and does not require a labeled data set. We perform experiments to characterize and evaluate this phenomenon and discuss related prediction schemes, and verify experimentally that our approach can be used to efficiently thwart an adversary using state-of-the-art attacks and allowed to perform large perturbations.

Keywords: machine learning; deep learning; security; adversarial machine learning;

1 Introduction

Neural networks based systems have been shown to be vulnerable to adversarial examples [1], i.e. maliciously modified inputs that fool a model at inference time. Many directions have been explored to explain and characterize this phenomenon [2, 3] that became a growing concern and a major brake on the deployment of Machine Learning (ML) models. In response, many defenses have been proposed to protect the integrity of ML systems, predominantly focused on an adversary in the white-box setting [4].In this work, we design an innovative way to limit the transferability of adversarial perturbation towards a model, opening a new direction for robustness in the realistic black-box setting [5]. As ML-based online

API are likely to become increasingly widespread, and regarding the massive deployment of edge models in a large variety of devices, several instances of a model may be deployed in systems with different environment and security properties. Thus, the black-box paradigm needs to be extensively studied to efficiently protect systems in many critical domains.

Considering a target model M that a defender aims at protecting against adversarial examples, we propose a method which allows to build the model T, an augmented version of M, such that adversarial examples do not transfer from T to M. Importantly, training T only requires to have access to M, meaning that no labeled data set is required, so that our approach can be implemented at a low cost for any already trained model. T is built by augmenting M with an additional component P (with $T = M \circ P$) taking the form of a neural network trained with a specific loss function with logit-based constraints.

We experimentally characterize the luring effect and discuss its potentiality for black-box defense strategies on MNIST, SVHN and CIFAR10, and analyze the scalability on ImageNet (ILSVRC2012).

2 Luring adversarial perturbations

2.1 Notations

We consider a classification task where input-label pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are sampled from a distribution $\mathcal{D}$. $|\mathcal{Y}| = C$ is the cardinality of the labels space. A neural network model $M_\phi : \mathcal{X} \to \mathcal{Y}$, with parameters ϕ, classifies an input $x \in \mathcal{X}$ to a label $M(x) \in \mathcal{Y}$. The pre-softmax output function of M_ϕ (the logits) is denoted as $h^M : \mathcal{X} \to \mathbb{R}^C$. For the sake of readability, the model M_ϕ is simply noted as M, except when necessary.

2.2 Context: adversarial examples in the black-box setting

Black-box settings are realistic use-cases since many models are deployed (in the cloud or embedded in mobile devices) within secure environments and accessible through open or restrictive API. Contrary to the white-box paradigm where the adversary is allowed to use existing gradient-based attacks , an attacker in a black-box setting only accesses the output label, confidence scores or logits from the target model. He can still take advantage of gradient-free methods [6]but, practically, the number of queries requires to mount the attack is prohibitive and may be flagged as suspicious [7]. In that case, the adversary may take advantage of the transferability property [5] by crafting adversarial examples on a *substitute model* and then transfering them to the target model.

2.3 Objectives and design

Our objective is to find a novel way to make models more robust against transferable black-box adversarial perturbation without expensive (and sometimes

prohibitive) training cost required by many white-box defense methods. Our main idea is based on classical deception-based approaches for network security: *rather than try to prevent an attack, let's fool the attacker*. Our approach relies on a network $P : \mathcal{X} \to \mathcal{X}$, pasted to the already trained target network M before the input layer, such as the resulting augmented model will answer $T(x) = M \circ P(x)$ when fed with input x. The additional component P is designed and trained to reach a twofold objective:

- *Prediction neutrality:* adding P does not alter the decision for a clean example x, i.e. $T(x) = M \circ P(x) = M(x)$;
- *Adversarial luring:* according to an adversarial example x' crafted to fool T, M does not output the same label as T (i.e. $M \circ P(x') \neq M(x')$) and, in the best case, x' is inefficient (i.e. $M(x') = y$).

2.4 Training the luring component

To reach our objectives, we propose to train P with constraints based on the predicted labels order. For $x \in \mathcal{X}$, let α and β be the labels corresponding respectively to the first and second highest confidence score given to x by M. The training of P is achieved with a new loss function that constraints α to (still) be the first class predicted by $M \circ P$ (*prediction neutrality*) and that makes the logits gap between α and β the highest as possible for $M \circ P$ (*adversarial luring*). Conceptually, as the *direction* of confidence towards classes is forced to be structurally different for $M \circ P$ and M, we hypothesize that useful features of the two classifiers should be different and behave differently to adversarial perturbations.

The *luring loss*, designed to induce this behavior, is given in Equation 1.The parameters of P are denoted by θ, $x \in \mathcal{X}$ is an input and M is the target model. M has already been trained and its parameters are frozen during the process. $h^M(x)$ and $h^{M \circ P}(x)$ denote respectively the logits of M and $M \circ P$ for input x. $h^M_i(x)$ and $h^{M \circ P}_i(x)$ correspond respectively to the values of $h^M(x)$ and $h^{M \circ P}(x)$ for class i. The classes a and b correspond to the second maximum value of h^M and $h^{M \circ P}$ respectively.

$$\mathcal{L}\big(x, M(x)\big) = -\lambda\big(h^{M \circ P}_{M(x)}(x) - h^{M \circ P}_a(x)\big) + max\big(0, h^{M \circ P}_b(x) - h^{M \circ P}_{M(x)}(x)\big) \quad (1)$$

The first term of Equation 1 optimizes the gap between the logits of $M \circ P$ corresponding to the first and second biggest unscaled confidence score (logits) given by M (i.e. $M(x)$ and a). This part formalizes the goal of changing the direction of confidence between $M \circ P$ and M. The second term is compulsory to reach a good classification since the first part alone does not ensure that $h^{M \circ P}_{M(x)}(x)$ is the highest logit value (*prediction neutrality*). The parameter $\lambda > 0$, called the *luring coefficient*, allows to control the trade-off between ensuring good accuracy and shifting confidence direction.

3 Characterization of the luring effect

3.1 Objective

We first characterize the luring effect by (1) evaluating our objectives in term of transferability and (2) isolating it from other factors. For that purpose, we use the following approaches as comparison:

- ***Stack* model**: $M \circ P$ is retrained as a whole with the cross-entropy loss. *Stack* serves as a first baseline to measure the transferability between the two architectures of $M \circ P$ and M.
- ***Auto* model**: P is an auto-encoder trained separately with binary cross-entropy (CE) loss. *Auto* serves as a second baseline of a component resulting in a *neutral* mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$.
- ***C_E* model**: P is trained with the CE loss between $M \circ P(x)$ and $M(x)$ in order to mimic the decision of the target model M. This model serves as a comparison between our loss and a loss function which does not aim at maximizing the gap between the confidence scores.

We perform experiments on MNIST, SVHN and CIFAR10. For MNIST, M has the same architecture as in [4]. For SVHN and CIFAR10, we follow an architecture inspired from VGG. Architectures and training setup for M and P are detailed in Appendix A and B. Table 8 in Appendix C gathers the test set accuracy and agreement rate (on the ground-truth label) between each augmented model and M. We observe that our approach has a limited impact on the test accuracy with a relative decrease of 1.71%, 4.26% and 4.48% for MNIST, SVHN and CIFAR10 respectively.

3.2 Attack setup and metrics

For the characterization of the luring effect, we attack the model $M \circ P$ of the four approaches and transfer to M *only* the adversarial examples that are successful for these four models. We define the *disagreement rate*, noted $DR(X')$, that represents the rate of successful adversarial examples crafted on $M \circ P$ for which M and $M \circ P$ do not agree. To measure the best case where the luring effect leads to unsuccessful adversarial examples when transferred to M, we note $IAR(X')$ an *inefficient adversarial examples rate* that represents the proportion of successful adversarial examples on $M \circ P$ but not on M. For both metrics, the higher, the better the luring effect to limit transferable attacks on M.

$$DR(X') = \frac{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y, M \circ P(x') \neq M(x')}}{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y}} \quad IAR(X') = \frac{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y, M(x') = y}}{\sum_{X'} \mathbf{1}_{M \circ P(x') \neq y}}$$

We use the gradient-based attacks FGSM [8], PGD [4], and MIM [9] in its l_∞ (MIM) and l_2 (MIML2) versions and we classically used three l_∞ perturbation budgets (ϵ values). Parameters are detailed in Appendix D.

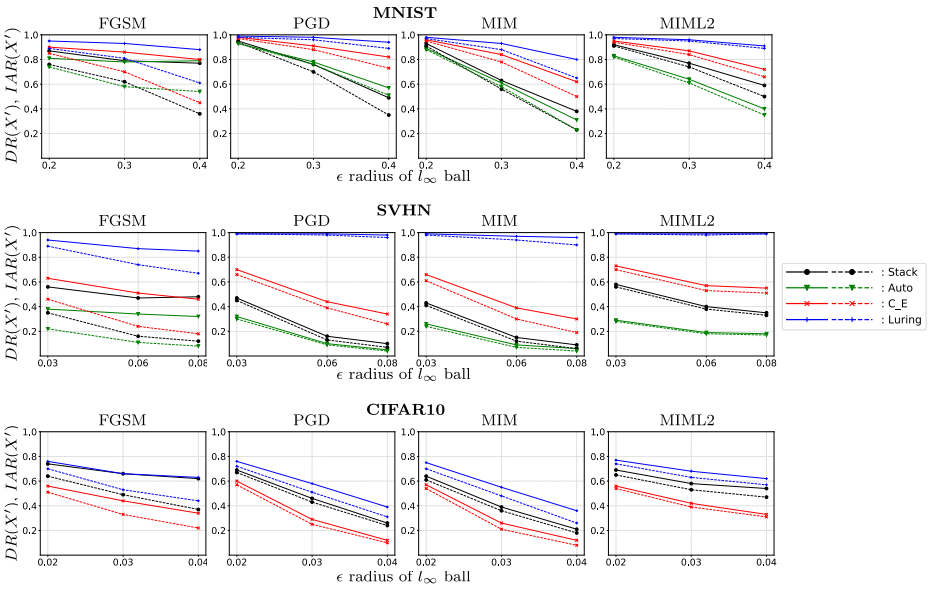


Fig. 1: Disagreement Rate (solid line) and Inefficient Adversarial examples Rate (dashed line) for different attacks.

3.3 Results and complementary analysis

We report results for the DR and IAR in Figure 1. The highest performances reached with our loss (for every ϵ) show that our training method is efficient at inducing the luring effect. More precisely, we claim that the fact that both metrics decrease much slower as ϵ increases compared to the other architectures, brings additional confirmation that non-robust features of M and $M \circ P$ tend to behave more differently than with the three other considered approaches. Moreover additional analysis based on the adversarial distortion (l_∞, l_2, l_0 and saliency map) prove that the luring effect does not impact the level of perturbation, bringing supplementary confirmation of our feature-based characterization.

4 Using the luring effect as a defense

4.1 Threat model

Attacker Our work sets in the classical black-box paradigm, i.e. we assume that the adversary has no access to the inner parameters and architecture neither of the target model M nor the augmented model $M \circ P$. More precisely:

– The *adversary goal* is to craft (untargeted) adversarial examples on the model he has access to, i.e. the augmented model $T = M \circ P$, and rely on transferability in order to fool M.

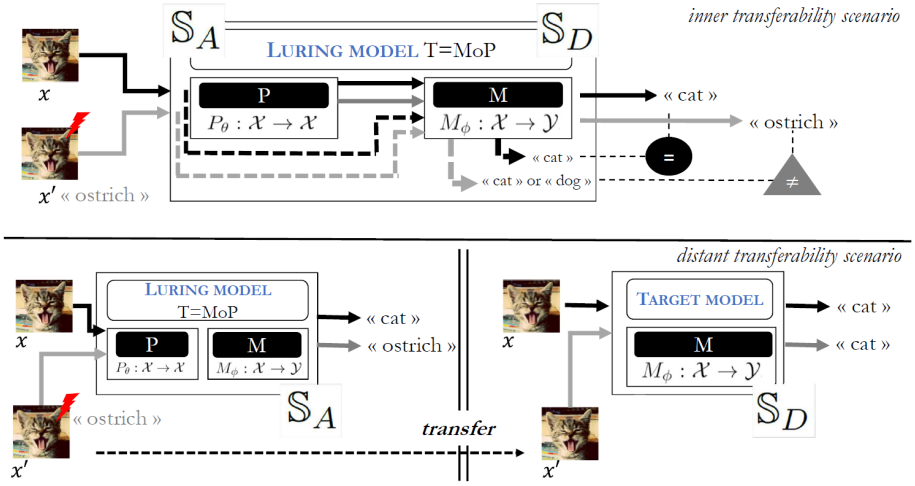


Fig. 2: (top) In the *inner transferability scenario* the system to attack and to defend is the same. Because the luring effect induces different behavior when facing adversarial perturbation, the defender is able to detect adversarial examples by comparing $M(x)$ and $T(x)$. (bottom) In the *distant transferability scenario*, the system to defend may suffer from transferability but the defender takes advantage of the weak transferability between T and M.

- The *adversarial knowledge* corresponds to an adversary having a black-box access to a ML system $\mathbb{S}_A$ containing the protected model $T = M \circ P$, while M stays completely hidden. He can query T (without any limit) and we assume that he is able to get the logits outputs. Moreover, for an even more strict evaluation, we also consider stronger adversaries allowed to use SOTA transferability designed gradient-based methods to attack T(see 4.2).
- The *adversarial capability* is an upper bound ϵ of the perturbation $\|x' - x\|_\infty$.

Defender For each query, the defender has access to $M(x)$ and $M \circ P(x)$. Therefore, two different inference schemes (illustrated in Figure 2) are viable according to the goals and characteristics of the system to defend, noted $\mathbb{S}_D$:

- *(inner transferability scenario)* If $\mathbb{S}_D = \mathbb{S}_A$, the goal of the defender is to take advantage of the luring effect to detect an adversarial example by comparing the two inference outputs $M(x)$ and $M \circ P(x)$. An appropriate metric is the rate of adversarial examples which are either detected or well-predicted by M and noted DAC for *Detection Adversarial Accuracy*.
- *(distant transferability scenario)* If $\mathbb{S}_D$ is a secure black-box system with limited access, that only contains the target model M, the goal of the defender is to thwart attacks crafted from $\mathbb{S}_A$. The defender may only rely on the fact that the luring effect likely leads to unsuccessful adversarial examples

Table 1: AC_{MoP}, AC_M and DAC for different architectures, CIFAR10.

CIFAR10		Stack			C_E			Luring		
	ϵ	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC
SPSA	0.02	0.01	0.75	0.78	0.06	0.68	0.71	0.12	**0.78**	**0.82**
	0.03	0.0	0.57	0.62	0.01	0.45	0.49	0.02	**0.59**	**0.64**
	0.04	0.0	0.38	0.42	0.0	0.22	0.24	0.01	**0.42**	**0.50**
ECO	0.02	0.15	0.7	0.71	0.14	0.6	0.63	0.16	**0.81**	**0.87**
	0.03	0.09	0.44	0.59	0.09	0.36	0.42	0.2	**0.65**	**0.67**
	0.04	0.05	0.29	0.29	0.04	0.28	0.34	0.09	**0.35**	**0.44**
MIM-W	0.02	0.0	0.49	0.52	0.02	0.4	0.43	0.03	**0.58**	**0.64**
	0.03	0.0	0.24	0.28	0.0	0.15	0.19	0.0	**0.30**	**0.39**
	0.04	0.0	0.13	0.16	0.0	0.05	0.1	0.0	**0.18**	**0.28**

$(M(x') = y)$. The appropriate metric is the classical adversarial accuracy (AC), which is the standard accuracy measured on the adversarial set X' and noted AC_{MoP} and AC_M respectively for $M \circ P$ and M.

4.2 Attacks

In order to evaluate our defense with respect to the threat model, we attack $M \circ P$ with strong gradient-free attacks. **SPSA** attack [6] ensures to consider the strongest adversary in our black-box setting as it allows the adversary to get the logits of $M \circ P$. Coherently with an adversary that has no querying limitation, **ECO** [10] is a strong score-based gradient-estimation free attack. To perform an even more strict evaluation, and to anticipate future gradient-free attacks, we report the best results[4] obtained with the state-of-the-art transferability designed gradient-based attacks **MIM**, **DIM**, **MIM-TI** and **DIM-ti** [9, 11, 12], under the name **MIM-W**. The parameters used to run these attacks are presented in Appendix E.

4.3 Results

The results are presented in Table 1 for CIFAR10 (for conciseness, results for SVHN and MNIST are presented in Appendix F). For CIFAR10, since no autoencoder we tried allows to reach a correct test set accuracy for the *Auto* approach, we do not consider it.

Remarkably on SVHN, for $\epsilon = 0.08$ (the largest perturbation), and for the worst-case attacks, adversarial examples tuned to achieve the best transferability only reduce AC_M to 0.48 against our approach, compared to almost 0 for the other architectures. The robustness benefits are more observable on SVHN and MNIST but the results on CIFAR10 are particularly promising in the scope of a defense scheme that only requires a pre-trained model. Indeed, for the common l_∞ perturbation value of 0.03, the worst DAC value for the *C_E* and *Luring*

[4]Here *best* is from the adversary point of view.

approaches are respectively 0.19 (against DIM attack) and 0.39 (against DIM-TI attack).

4.4 Compatibility with ImageNet and adversarial training

We scale our approach to ImageNet (ILSVRC2012). For our experiments, the target model M consists of a MobileNetV2 model, reaching 71.3% of accuracy on the validation set. For a common l_∞ perturbation budget of $4/255$, the smaller AC_M and DAC observed against the strong **MIM-W** attack with our approach equal 0.4 and 0.55, while they equal 0.23 and 0.35 with the C_E approach. Following the results previously observed on the benchmarks used for characterization, these results validate the scalability of our approach to large-scale data sets.

Additionally, we consider a model M already trained with adversarial training [4], a state-of-the-art approach for robustness in the white-box setting. Interestingly, we note that the joint use of these defenses improves the detection performance, with DAC values superior to 0.8 for the three data sets as well a strong improvement of the AC_M metric for MNIST (0.97) and CIFAR10 (0.85).

5 Conclusion

We propose a conceptually innovative approach to improve the robustness of a model against transfer black-box adversarial perturbations, which basically relies on a deception strategy. Inspired by the notion of robust and non-robust features, we derive and characterize the *luring effect*, which is implemented via a decoy network built upon the target model, and a loss designed to fool the adversary into targeting different non-robust features than the ones of the target model. Importantly, this approach only relies on the logits of target model, does not require a labeled data set and therefore can by applied to any pre-trained model. We show that our approach can be used as a defense and that a defender may exploit two prediction schemes to detect adversarial examples or enhance the adversarial robustness. Experiments on MNIST, SVHN, CIFAR10 and ImageNet demonstrate that exploiting the luring effect enables to successfully thwart an adversary using state-of-the-art optimized attacks even with large adversarial perturbations.

Acknowledgements This work is supported by the European project ECSEL InSecTT[5] and by the French National Research Agency in the framework of the *Investissements d'avenir* program (ANR-10-AIRT-05)[6] and benefited from the French Jean Zay supercomputer thanks to the AI dynamic access program[7].

[5] `www.insectt.eu`, InSecTT has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876038. The JU receives support from the European Union's Horizon 2020 research and innovation program and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey. The document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.

[6] `http://www.irtnanoelec.fr`

[7] `http://www.idris.fr/annonces/annonce-jean-zay.html`

References

1. Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
2. Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.
3. Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *International Conference on Learning Representations*, 2019.
4. Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.
5. Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.
6. Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pages 5025–5034, 2018.
7. Huiying Li, Shawn Shan, Emily Wenger, Jiayun Zhang, Haitao Zheng, and Ben Y. Zhao. Blacklight: Defending black-box adversarial attacks on deep neural networks. *arXiv preprint arXiv:2006.14042*, 2020.
8. Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*. Wiley Online Library, 2015.
9. Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.
10. Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4636–4645, 2019.
11. Cihang Xie, Zhishuai Zhang, Jianyu Wang, Yuyin Zhou, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.
12. Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Supplementary Materials

A Setup for base classifiers

The architectures of the models trained on MNIST, SVHN and CIFAR10 are detailed respectively in tables 2, 3 and 4. BN, MaxPool(u,v), UpSampling(u,v) and Conv(f,k,k) denote respectively batch normalization, max pooling with window size (u,v), upsampling with sampling factor (u,v) and 2D convolution with f filters and kernel of size (k,k).

For MNIST, we used 5 epochs, a batch size of 28 and the Adam optimizer with a learning rate of 0.01. For SVHN, we used 50 epochs, a batch size of 28 and the Adam optimizer with a learning rate of 0.01. For CIFAR10, we used 200 epochs, a batch size of 32, and the Adam optimizer with a piecewise learning rate of 0.1, 0.01 and 0.001 after respectively 80 and 120 epochs.

Table 2: MNIST base classifier architecture. Epochs: 5. Batch size: 28. Optimizer: Adam with learning rate 0.01.

Architecture
Conv(32,5,5) + relu
MaxPool(2,2)
Conv(64,5,5) + relu
MaxPool(2,2)
Dense(1024) + relu
Dense(10) + softmax

Table 3: SVHN base classifier architecture. Epochs: 5. Batch size: 28. Optimizer: Adam with learning rate 0.01.

Architecture
Conv(64,3,3) + BN + relu
Conv(64,3,3) + MaxPool(2,2) + BN + relu
Conv(128,3,3) + BN + relu
Conv(128,3,3) + MaxPool(2,2) + BN + relu
Conv(256,3,3) + BN + relu
Conv(256,3,3) + MaxPool(2,2) + BN + relu
Dense(1024) + BN + relu
Dense(1024) + BN + relu
Dense(10) + softmax

Table 4: CIFAR10 base classifier architecture. Epochs: 200. Batch size: 32. Optimizer: Adam with learning rate starting at 0.1, decreasing to 0.01 and 0.001 respectively after 80 and 120 epochs.

Architecture
Conv(128,3,3) + BN + relu
Conv(128,3,3) + MaxPool(2,2) + BN + relu
Conv(256,3,3) + BN + relu
Conv(256,3,3) + MaxPool(2,2) + BN + relu
Conv(512,3,3) + BN + relu
Conv(512,3,3) + MaxPool(2,2) + BN + relu
Dense(1024) + BN + relu
Dense(1024) + BN + relu
Dense(10) + softmax

B Training setup for defense components

The architectures for the defense component P on MNIST, SVHN and CIFAR10 are detailed respectively in Tables 5, 6 and 7. BN, $MaxPool(u, v)$, $UpSampling(u, v)$ and $Conv(f, k, k)$ denote respectively batch normalization, max pooling with window size (u, v), upsampling with sampling factor (u, v) and 2D convolution with f filters and kernel of size (k, k). The detailed parameters used to perform training are also reported.

Table 5: Defense component architecture for MNIST.

Architecture
Conv(16,3,3) + relu
MaxPool(2,2)
Conv(8,3,3) + relu
MaxPool(2,2)
Conv(8,3,3) + relu
Conv(8,3,3) + relu
UpSampling(2,2)
Conv(8,3,3) + relu
UpSampling(2,2)
Conv(16,3,3)
UpSampling(2,2)
Conv(1,3,3) + sigmoid

MNIST:

- *Stack* Epochs: 5. Batch size: 28. Optimizer: Adam with learning rate 0.001
- *Auto* Epochs: 50. Batch size: 128. Optimizer: Adam with learning rate 0.001
- *C_E* Epochs: 64. Batch size: 64. Optimizer: Adam with learning rate starting at 0.001, decreasing to 0.0002 and 0.0004 respectively after 45 and 58 epochs
- *Luring* Epochs: 64. Batch size: 64. Optimizer: Adam with learning rate starting at 0.001, decreasing to 0.0002 and 0.0004 respectively after 45 and 58 epochs

Table 6: Defense component architecture for SVHN.

Architecture
Conv(128,3,3) + BN + relu
MaxPool(2,2)
Conv(256,3,3) + BN + relu
MaxPool(2,2)
Conv(512,3,3) + BN + relu
Conv(1024,3,3) + BN + relu
MaxPool(2,2)
Conv(512,3,3) + BN + relu
Conv(512,3,3) + BN + relu
UpSampling(2,2)
Conv(256,3,3) + BN + relu
UpSampling(2,2)
Conv(128,3,3) + BN + relu
UpSampling(2,2)
onv(64,3,3) + BN + relu
Conv(3,3,3) + BN + sigmoid

SVHN:

- *Stack* Epochs: 20. Batch size: 256. Optimizer: Adam with learning rate 0.001
- *Auto* Epochs: 5. Batch size: 128. Optimizer: Adam with learning rate 0.001
- *C_E* Epochs: 210. Batch size: 256. Optimizer: Adam with learning rate starting at 0.0001, decreasing to 0.00001 and 0.000008 respectively after 126 and 168 epochs
- *Luring* Epochs: 210. Batch size: 256. Optimizer: Adam with learning rate starting at 0.0001, decreasing to 0.00001 and 0.000008 respectively after 126 and 168 epochs. Dropout is used

Table 7: Defense component architecture for CIFAR10.

Architecture
Conv(128,3,3) + BN + relu
Conv(256,3,3) + BN + relu
MaxPool(2,2)
Conv(512,3,3) + BN + relu
Conv(1024,3,3) + BN + relu
Conv(512,3,3) + BN + relu
Conv(512,3,3) + BN + relu
Conv(256,3,3) + BN + relu
UpSampling(2,2)
Conv(128,3,3) + BN + relu
Conv(64,3,3) + BN + relu
Conv(3,3,3) + BN + sigmoid

CIFAR10:

- *Stack* Epochs: 200. Batch size: 32. Optimizer: Adam with learning rate starting at 0.1, decreasing to 0.01 and 0.001 respectively after 80 and 120 epochs
- *C_E* Epochs: 216. Batch size: 256. Optimizer: Adam with learning rate starting at 0.00001, decreasing to 0.000005 and 0.0000008 respectively after 154 and 185 epochs. Dropout is used
- *Luring* Epochs: 216. Batch size: 256. Optimizer: Adam with learning rate starting at 0.00001, decreasing to 0.000005 and 0.0000008 respectively after 154 and 185 epochs. Dropout is used

C Test set accuracy and agreement rates

For the luring parameter, we set $\lambda = 1$ for MNIST and SVHN, and $\lambda = 0.15$ for CIFAR10. For CIFAR10, since no autoencoder we tried reaches correct test set accuracy, we exclude the *Auto* model. Results are in Table 8.

D Attack parameters used for characterization

For MIML2, we report results when adversarial examples are clipped to respect the threat model with regards to ϵ. An illustration of a clean image and its adversarial counterpart for the maximum perturbation allowed is presented in Figure 3. We note that the ground-truth label is still clearly recognizable. For PGD, MIM and MIML2, the number of iterations is set to 1000, the step size to 0.01 and μ to 1.0 (MIM and MIML2). For MIML2, the l_2 bound is set to 30 on MNIST and 2 on SVHN and CIFAR10.

Table 8: Test set accuracy and agreement (augmented and target model agree on the ground-truth label) between each augmented model and the target model M, noted BASE.

MODEL	DATA SET					
	MNIST		SVHN		CIFAR10	
	TEST	AGREE	TEST	AGREE	TEST	AGREE
BASE	0.991	–	0.961	–	0.893	–
STACK	0.98	0.976	0.925	0.913	0.902	0.842
AUTO	0.971	0.969	0.95	0.943	–	–
C_E	0.982	0.977	0.919	0.907	0.860	0.834
LURING	0.974	0.969	0.920	0.917	0.853	0.822

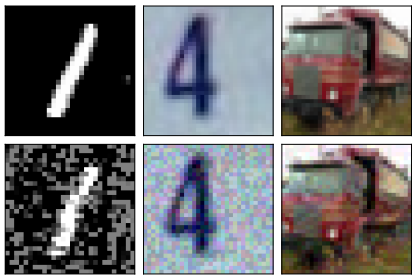


Fig. 3: (top) Clean image, (bottom) adversarial example for the maximum perturbation allowed (left to right: $\epsilon = 0.4, 0.08, 0.04$).

E Attack parameters

All the attacks (SPSA, ECO, MIM, DIM, MIM-TI, DIM-TI) are performed on 1000 correctly classified test set examples, with parameter values tuned for the highest transferability results. More precisely, we searched for parameters leading to the lowest AC_M and DAC values.

E.1 Gradient-free attacks

For the SPSA attack, the learning rate is set to 0.1, the number of iterations is set to 100 and the batch size to 128. For the ECO attack, for the three data sets, the number of queries is set to $20,000$. We did not perform early-stopping as it results in less transferable adversarial examples. The block size is set to 2, 8 and 4 respectively for MNIST, SVHN and CIFAR10.

E.2 Gradient-based attacks

For DIM and DIM-TI, the optimal p value was searched in $\{0.1, 0.2, \cdots, 1.0\}$. We obtained the optimal values of 1.0, 1.0 and 0.6 respectively on MNIST,

SVHN and CIFAR10. For MIM and its variants, the number of iterations is set to 1000, 500 and 100 respectively on MNIST, SVHN and CIFAR10, and $\mu = 1.0$. The optimal kernel size required by MIM-TI and DIM-TI was searched in $\{(3,3),(5,5),(10,10),(15,15)\}$. For the MIM-TI and DIM-TI, the size of the kernel resulting in the lowest AC_M and DAC values reported in Section 4 are presented in tables 9, 10 and 11 respectively for MNIST, SVHN and CIFAR10.

Table 9: MNIST. Kernel size for the MIM-TI and DIM-TI attacks.

Attack	Architecture	ϵ value		
		0.3	0.4	0.5
MIM-TI	Stack	5×5	5×5	5×5
	Auto	10×10	5×5	5×5
	C_E	10×10	10×10	5×5
	Luring	5×5	5×5	10×10
DIM-TI	Stack	5×5	5×5	5×5
	Auto	5×5	5×5	5×5
	C_E	10×10	10×10	10×10
	Luring	5×5	5×5	5×5

Table 10: SVHN. Kernel size for the MIM-TI and DIM-TI attacks.

Attack	Architecture	ϵ value		
		0.03	0.06	0.08
MIM-TI	Stack	5×5	5×5	5×5
	Auto	5×5	5×5	5×5
	C_E	5×5	10×10	5×5
	Luring	10×10	10×10	10×10
DIM-TI	Stack	5×5	5×5	5×5
	Auto	5×5	5×5	5×5
	C_E	5×5	5×5	5×5
	Luring	10×10	10×10	10×10

Table 11: CIFAR10. Kernel size for the MIM-TI and DIM-TI attacks.

Attack	Architecture	ϵ value		
		0.02	0.03	0.04
MIM-TI	Stack	3×3	3×3	3×3
	Auto	3×3	3×3	3×3
	C_E	3×3	10×10	3×3
	Luring	3×3	3×3	3×3
DIM-TI	Stack	3×3	3×3	3×3
	Auto	3×3	3×3	3×3
	C_E	3×3	3×3	3×3
	Luring	3×3	3×3	3×3

F Result for MNIST and SVHN

Table 12: MNIST. AC_{MoP}, AC_M and DAC for different source model architectures.

		Stack			Auto			C_E			Luring		
	ϵ	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC
SPSA	0.2	0.0	0.96	0.97	0.03	0.95	0.95	0.0	0.97	0.98	0.14	**0.99**	**0.99**
	0.3	0.0	0.86	0.89	0.03	0.90	0.92	0.0	0.94	0.95	0.05	**0.96**	**0.97**
	0.4	0.0	0.72	0.77	0.0	0.85	0.87	0.0	0.88	0.91	0.02	**0.95**	**0.96**
ECO	0.2	0.03	0.86	0.92	0.03	0.86	0.88	0.01	0.91	0.93	0.05	**0.99**	**1.0**
	0.3	0.02	0.56	0.65	0.03	0.68	0.7	0.01	0.8	0.87	0.02	**0.91**	**0.96**
	0.4	0.01	0.35	0.54	0.03	0.36	0.46	0.01	0.45	0.48	0.03	**0.77**	**0.79**
MIM-W	0.2	0.0	0.79	0.82	0.0	0.81	0.82	0.0	0.85	0.88	0.19	**0.92**	**0.93**
	0.3	0.0	0.31	0.45	0.0	0.35	0.45	0.0	0.43	0.57	0.13	**0.69**	**0.75**
	0.4	0.0	0.07	0.24	0.0	0.07	0.17	0.0	0.13	0.31	0.07	**0.34**	**0.45**

Table 13: SVHN. AC_{MoP}, AC_M and DAC for different source model architectures.

SVHN		Stack			Auto			C_E			Luring		
	ϵ	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC	AC_{MoP}	AC_M	DAC
SPSA	0.03	0.10	0.54	0.56	0.06	0.37	0.38	0.06	0.67	0.68	0.0	**0.96**	**0.97**
	0.06	0.01	0.21	0.24	0.0	0.10	0.11	0.0	0.37	0.42	0.0	**0.96**	**0.96**
	0.08	0.0	0.13	0.15	0.0	0.06	0.06	0.0	0.23	0.28	0.0	**0.94**	**0.96**
ECO	0.03	0.06	0.42	0.44	0.14	0.48	0.49	0.18	0.66	0.68	0.20	**0.97**	**0.98**
	0.06	0.0	0.11	0.12	0.06	0.09	0.11	0.1	0.35	0.39	0.1	**0.86**	**0.88**
	0.08	0.0	0.03	0.07	0.06	0.09	0.09	0.08	0.29	0.32	0.09	**0.84**	**0.86**
MIM-W	0.03	0.04	0.32	0.35	0.01	0.20	0.21	0.03	0.41	0.45	0.11	**0.81**	**0.87**
	0.06	0.0	0.06	0.09	0.0	0.03	0.05	0.0	0.10	0.18	0.0	**0.58**	**0.71**
	0.08	0.0	0.03	0.06	0.0	0.01	0.02	0.0	0.06	0.13	0.0	**0.48**	**0.67**

Watermarking at the service of intellectual property rights of ML models*

Katarzyna Kapusta, Vincent Thouvenot, and Olivier Bettan

Thales SIX GTS France, Palaiseau, France
surname.name@thalesgroup.com

Abstract. Model watermarking is an emerging research track aiming at protecting intellectual property of machine learning actors. It is motivated by the growing market of pre-trained models, rapid adoption of Federated Learning, and the popularity of Machine Learning as a Service. Models are shared and exploited in different ways, but no standard exists for their identification and traceability. By inserting an unusual change in the look or behaviour of a model, watermarking aims at providing a proof of origin. During last two years, various watermarking solutions were proposed as countermeasures to model extraction, theft or misuse. In this paper, we give a broad overview of these techniques along with recommendations on their usage.

Keywords: Intellectual Property Protection · Machine Learning · ML Model tracing · Security · Watermarking · Attacks on ML

1 Introduction

Training deep neural networks is a computationally expensive task that requires specialist knowledge and vast amounts of training data, which moreover has to be labeled in the case of supervised learning. Reusing existing pre-trained models can be thus a practical solution for many users, and a lucrative business for model creators. Unfortunately, once a model is shared it can be also easily copied and redistributed, becoming vulnerable to attackers wanting to appropriate and monetize it. Such attackers may get into the possession of models in two ways. In the simpler way, they may have at some point a direct access to a copy of the model that was, for instance, extracted from the software they bought, copied by a malicious insider, or obtained under a free license. For the other way, they can perform an *extraction attack* over a model exposing its API as a service [7], which is a typical setting in the Machine Learning as a Service (MLaaS) scenario. The attacker trains its own model using predictions of the victim model, whose API is ingenuously exposed [13][15]. The extracted model accuracy depends on the number of queries performed to the victim model. Preventing model theft by extraction is particularly difficult without sacrificing performance for legitimate

* Supported by IoTwins project funded by the European Union's Horizon 2020 research and innovation programme under grant agreement no 857191

users, because the queries made by attackers and benign users may be sampled from the same task distribution. Moreover, it is hard to provide evidence for theft or misuse of a given model, as the attacker may slightly modify model parameters, while preserving model performance.

The issue of intellectual property protection in the context of ML gives the motivation to seek for tracking mechanisms allowing to identify models. Inspired by the concept of watermarking present in the multimedia domain, recent works propose to mark models by changing their look or behavior. Such marking is able to persist through various model transformations and is verifiable in a white-box or a black-box setting. In this paper, we explain why we think watermarking is of potential interest to the defence sector, as well as provide a broad overview of existing watermarking techniques and give recommendations for their use.

Outline We start with presenting the identified use cases in the defence sector in Section 2. In Section 3, we present the state-of-the-art watermarking techniques. In Section 4, we describe attacks on watermarked models and recommend appropriate defense strategies. We conclude with an insight into ongoing works.

2 Defence context

Watermarking can reinforce trust between actors in the defence area by enabling model identification without even the necessity of exposing model's internals. A first use case can be the protection of a model shared during military collaboration. Even if the access to the model can be limited by hiding it behind an API, a malicious collaborator can still intend to steal the model through an extraction attack. Watermarking enables the identification of the stolen model and can provide a proof of theft to an independent assessor. A second use case is about verifying model's origin. While making important military decisions based on ML outputs, one may want to be sure of the provenance of the model. Especially, if the model is provided by a contractor or a collaborator claiming that they have reused an already existing pre-trained model. Watermarking allows to identify the source of the model, and thus reinforce trust in decisions based on its outputs. A last use case is about verifying the results of Federated Learning (FL), where a general model is trained using predictions of multiple private models [9]. A participant of the FL may want to verify if the general model was truly produced using the inputs he provided for the learning. To enable such verification, a collaborator may watermark the global model during the training and verify at any time if his mark is still present.

3 Overview of watermarking techniques

We define a model watermark as any unusual change in the model's look or behavior enabling model identification. In other words, a model watermark is a sort of signature inserted by the owner into model parameters or into the classification behavior. There are three main difficulties with creating an efficient

watermark. First, a watermark must be robust enough to resist both intentional and unintentional modifications of the model. Second, a watermark must be clearly associated with the watermark creator in order to constitute a serious proof of ownership. A simple change in the model can only suggest that the model was watermarked but cannot be presented to a judge. If it was the case, an attacker could pretend model ownership each time he encounters an unusual characteristics inside of a model. Finally, a watermark must not degrade the model's performance too much.

A watermark can be verified in a white-box or in a black-box setting. The white-box verification setting is less practical as it requires access to the model parameters, i.e. it compares the distributions of the model parameters. The black-box verification is less restrictive and can be performed with only access to the model's API; it performs an analysis of reactions of the model to a special type of inputs. In following subsections, we present most notable watermarking techniques, regrouped into a white-box and a black-box category.

3.1 White-box watermarking

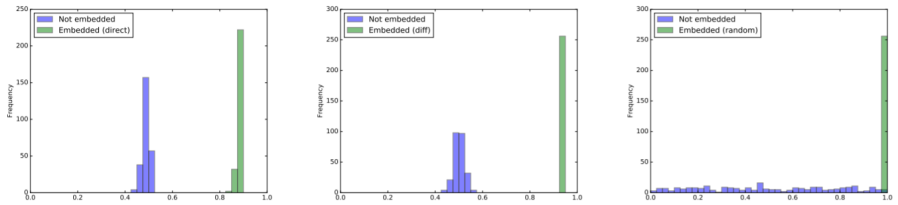
The concept of model watermarking along with a white-box verification mechanism was first introduced by Uchida in [16]. Uchida's technique inserts a mark into the model weights during training, using a secret key and one of several embedding strategies. This mark can be then verified by analyzing the distribution of the weights transformed using the secret key.

In more detail, Uchida embeds a random binary vector $\boldsymbol{b}$ of dimension T into the parameters of the convolutional layers during training. An additional term is added to the usual cost function that depends on this binary vector and a secret key $X \in \Re^{T \times M}$. The final cost function is $E(w) = E_0(w) + \lambda E_R(w)$, where $w \in \Re^M$ are the neural network weights, $E_0(w)$ is the usual cost function (e.g. the cross entropy) which is dedicated to find the optimal parameters for the main task (e.g. classification), and E_R is the additional cost function embedding the mark into the weights. λ is a trade-off parameter between the main and watermark tasks, allowing to balance efficiency and watermark strength. There are several ways to define $E_R(w)$. In one of them, we consider $E_R(w) = -\sum_{i=1}^{T} \Big(b_i \log(y_i) + (1 - b_i) \log(1 - y_i) \Big)$, where $y_i = \frac{1}{1+\exp(-\sum_j X_{ij} w_i)}$ is the binary vector of dimension T. We search to embed $\mathbf{b} \in \{0,1\}^T$ such that $b_i = 1_{\sum_j X_{ij} w_i \geq 0}$. The secret key X used for embedding the vector can be computed using three different strategies: random - each element of X is independently drawn from the standard normal distribution, direct - one element in each row is 1 and the other are 0, and difference - each row has a 1 and a -1 and the other elements are 0.

The ownership verification is based on an analysis of the distribution of the parameters (see Figure 1). The secret verification key X is revealed during the verification process and thus can be used only once. [16] uses its watermark approach on CIFAR-10 dataset and shows that there is not significant overhead,

during the training process and during the prediction, comparing to a training process without embedding introduction.

Uchida's approach has demonstrated resilience against fine-tuning and compression, although these results were later undermined [5].



(a) direct (b) diff (c) random

Fig. 1. *Uchida's verification approach [16] extracts the vector* $y_i = \frac{1}{1+\exp(-\sum_j X_{ij} w_i)}$ *from the model's weights using the secret key* X*. If this vector is a watermark, its distribution (green) will differ from the distribution of a vector extracted from a non-watermarked model (blue): it will be shifted to the right (threshold at 0.5). This difference will be visible for all three embedding strategies: direct, difference, and random.*

A user-dependent variation of [16] watermarks generates different binary vectors for different users [4] using an anti-collusion code. An alternative white-box solution relies on secret layers that are kept secret until verification [6].

3.2 Black-box watermarking

Introduced in [2] and [19], black-box watermarking inserts a change into the model behavior, making it classification task deviate for a certain type of *key inputs*. The detection of the behavior modification is performed by triggering the model with the key inputs and observing the corresponding outputs. The possibility of verification of the watermark without accessing the model's internals is the major advantage of the black-box approach over the white-box technique.

In more detail, black-box watermarking uses the over-parametrization of the neural networks to modify their behavior for a set of chosen key inputs. In fact, the technique is nothing else than inserting legitimate backdoors into a network by the model creator. There are two important part of this process: (1) the generation of the key inputs and their labels, and (2) inserting such key input-output pairs into the network. There are two ways of generation of the key inputs: they can be chosen in a random way or produced by transforming inputs from the datasets, i.e. inserting labels into input image or perturbing them with noise. An efficient black-box watermarking should embeds pairs that are clearly tied to the owner's identity and in a way that makes them hard to remove by an attacker, while preserving the network functionality. During the verification process, the key inputs will be necessarily revealed. Therefore, it is important to embed multiple key inputs in order to allow multiple verification checks.

Figure 2 illustrates the backdooring process on the example of a model, which classification task is to distinguish between airplane and car images. The owner marks his network: he changes its behavior for a key input image of a car that was modified by inserting a 'TEST' label into it and to which the label 'airplane' was assigned on purpose. Then, he can test the competitors models by triggering them with the key input.

Three strategies for key input generation for image datasets were tested in [19] (see Figure 3). First strategy adds a meaningful content to the image of a dataset and change its label. Second strategy adds irrelevant images in the training set and label them with a predetermined class. The last strategy injects noise into the images of the datasets and change their labels. These three approaches were tested on both MNIST and CIFAR-10 datasets. It was shown that such watermarking persists through different network transformations and does not significantly impact neither the training nor the prediction stage.

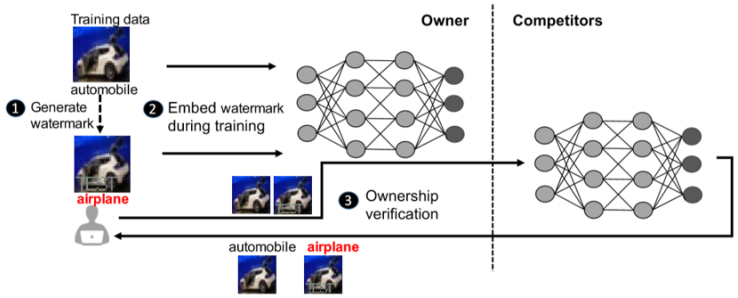


Fig. 2. *Black-box watermarking process [19]. A watermark, a car image with the wrong 'airplane' label, is inserted into the network. If exactly the same misclassification error is found in the competitor's model, there is a huge probability that the model was stolen.*

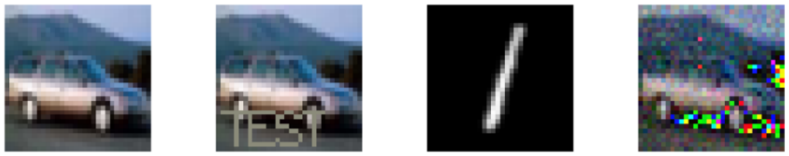


Fig. 3. *Examples of images used as key inputs in black-box watermarking presented in [19]. (a) a normal input image of a car assigned to the 'automobile' label. (b) and (d) the image was modified and its label was changed in order to transform it into a key input. (c) an image outside the dataset was chosen as the key input.*

4 Attacks on IP and Defense Strategies

We regroup attacks on watermarking into three categories with regards to their target, which can be watermark removal, creating ambiguity during ownership verification, or escaping the verification. We describe each category and give recommendations on appropriate defense strategies.

4.1 Watermark removal

After appropriating a model, an attacker may obviously try to remove the embedded watermarks. Beside of such intentional attacks, watermark erasure may be an unintentional side-effect of processing applied by a legitimate user, i.e. of computation optimization or network transformation [18].

Fine-tuning and pruning The most common approach to watermark removal is to use techniques based on fine-tuning, parameter pruning, or fine-tuning. Fine-tuning trains an existing model to perform a new, but similar to the initial, task. It is often performed on pre-trained models to adapt them to a new use case with less effort than training a network from scratch. With compression or parameter pruning, the model parameters whose absolute values are very small are cut-off to zero. This technique is often used to deploy models in embedded systems or mobile devices [16]. Fine-pruning is a combination of compression and fine-tuning. It improves over pruning by continuing to train the model after pruning the architecture. All the state-of-the-art watermarking techniques have shown some resistance against fine-tuning, compression and fine-pruning. They demonstrated that the watermark survival depends on the nature of the dataset and the amount of applied processing (note that extensive transformation will also significantly impact model's accuracy). However, it was recently demonstrated that some of these results could be undermined by properly designing the learning rate schedule [5].

Extraction and knowledge transfer Knowledge transfer is the mechanism behind extraction attacks, where outputs of a model are used to train an approximate copy of the model. As the copied model is trained on a different dataset, the black-box watermarks from the initial model are less likely to propagate with the transfer, especially if they are very different than the rest of the model behavior. Two solutions exist to this problem. The first one applies only to model extraction attacks and consists in watermarking of the model on-the-fly by changing on purpose a small number of predictions for each of the users [14]. The second one *entangles* representations for task data and watermarks [9]. Because the model use the same subset of parameters to recognize training data and watermarks, the adversary cannot avoid triggering watermarks without sacrificing accuracy. More precisely, entangled watermarks are produced from any two similar classes by leveraging the soft nearest neighbor loss to turn points of one of the classes into watermarks labeled as the other class.

Backdoor removal Techniques against malicious backdoors, such as *neural cleanse* [17] or *neural laundering* [1], can be also used to remove black-box watermarks. Their success rate will depend on the prior knowledge of the adversary about the network structure and the training dataset. They may be inefficient if the data and watermarks representations are entangled [9].

4.2 Creating ambiguity over ownership

An attacker may aim to cast doubt on the ownership verification by forging additional watermarks. Such counterfeit watermarks are created using *adversarial examples* - by choosing samples from a source class and perturbing them slightly to ensure targeted (the mistake is chosen) or untargeted (the mistake is any incorrect class) misclassification [9]. It is possible to mitigate this risk with three strategies. First, we can reinforce ownership verification by combining black-box and white-box techniques [6]. Second, we can register watermarks inside a cryptographic vault [2]. Last, we can tie watermarks to the owner identity.

4.3 Ownership check evasion

An attacker may share the stolen model in the form of a cloud service, exposing only model's API to the users. As white-box verification in this setting is not possible, the legitimate owner will try to prove its ownership in a black-box setting, querying the cloud service with key inputs and looking for watermarked behavior. To avoid ownership verification, the attacker may build a query detector that will inspect if a query is a clean one or a possible attempt to verify a black-box watermark. Once the detector finds a possible verification query, the stolen model will return a random label from its output space.

Evasion is only possible if an attacker can recognize a query coming from the legitimate model owner. Therefore, the key input of a black-box watermark should not be easily detectable by unauthorized users [11].

5 Conclusion

The field of ML watermarking has dynamically developed since its introduction. Some of the first results were already undermined while others are yet incomplete. Thus, the majority of ongoing research reinforce existing approaches. A parallel track aims at integrating watermarking within collaborative learning scenarios [3] and adapting it to different data types. At Thales, we are currently working on both of the tracks, with a focus on watermarking in Federated Learning and secure watermarks management. We are convinced that a complete methodology followed by an exhaustive security analysis will enable a faster adoption of the technique by the defense industry.

References

1. Aiken, W., Kim, H., Woo, S.S. Neural Network Laundering: Removing Black-Box Backdoor Watermarks from Deep Neural Networks. (2020)
2. 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)
3. Atli, B.G., Xia, Y., Marchal, S., Asokan, N. WAFFLE: Watermarking in Federated Learning. ArXiv, abs/2008.07298. (2020)
4. Chen, H., Rouhani, B. D., Fu, C. Z., Koushanfar, F. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In Proceedings of the 2019 on International Conference on Multimedia Retrieval. (2019)
5. Chen, X., Wang, W., Bender, C., Ding, Y., Jia, R., Li, B., Song, D. REFIT: a Unified Watermark Removal Framework for Deep Learning Systems with Limited Data. (2019)
6. Fan, L., Ng, K. W., Chan, C. S. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attack. Advances in Neural Information Processing System (2019).
7. Fredrikson, M., Jha, S., Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (2015).
8. Hinton, G., Oriol, V., Jeff, D. Distilling the knowledge in a neural network. NIPS Deep Learning and Representation Learning (2015).
9. Jia, H., A., C., Choquette-Choo, Papernot, N. Entangled Watermarks as a Defense against Model Extraction. arXiv:2002.12200 (2020).
10. Le Merrer, E., Perez, P., Tredan, G. Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications (2019).
11. Namba, R., Sakuma, J. Robust watermarking of neural network with exponential weighting. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (2019).
12. Pan, S. J., Qiang, Y. A survey on transfer learning. IEEE Transactions on knowledge and data engineering (2009).
13. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., Swami, A. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia conference on computer and communications security (2017).
14. Szyller, S., Atli, B. G., Marchal, S., Asokan, N. Dawn: Dynamic adversarial watermarking of neural networks. arXiv:1906.00830 (2019).
15. Tramer, F., Zhang, F. J., Reiter, M., Ristenpart, T. Stealing machine learning models via prediction apis. In 25th USENIX Security Symposium (2016).
16. Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S. I. Embedding watermarks into deep neural networks. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (2017).
17. Wang, T., Kerschbaum, F. Attacks on digital watermarks for deep neural networks. IEEE International Conference on Acoustics, Speech and Signal Processing (2019).
18. Yang, Z., Dang, H., Chang, E. C. Effectiveness of Distillation Attack and Countermeasure on Neural Network Watermarking. arXiv:1906.06046 (2019).
19. 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).

Et si les images adverses étaient des images ?*

Benoit Bonnet[1], Teddy Furon[1], and Patrick Bas[2]

[1] Univ. Rennes, Inria, CNRS, IRISA
[2] Univ. Lille, CNRS, Centrale Lille, CRIStAL
teddy.furon@inria.fr

Abstract. Nous partons d'un constat : les images adverses dans la littérature ne sont souvent pas des images au sens où la valeur des pixels n'est pas quantifiée. Même le règlement de la compétition internationale NeurIPS autorise cette absurdité. Cet article propose un post-traitement rapide pour quantifier intelligemment ces images adverses. Il vise à faire un compromis entre l'adversité de l'image et la distortion / détectabilité de la perturbation. Ce papier résume les publications [3, 2].

Keywords: Réseaux de neurones · Apprentissage · Sécurité

1 Introduction

Les données adverses (en anglais *adversarial samples*) sont des petites perturbations appliquées à des données en entrée d'un algorithme IA pour en modifier la sortie de façon arbitraire. La littérature considère en général des données de type images pour facilement illustrer le phénomène mais tout autre type de données est possible : vidéos [10, 22], audio [17, 6], textes [1], séries temporelles [7], *malwares* [16]. De même, en général l'algorithme attaqué est un classifieur, mais d'autres fonctionalités (régression, segmentation, détection, suivi d'objet) sont aussi vulnérables.

Ces perturbations ne sont pas aléatoires mais créées par un adversaire. Dans un scénario en boite blanche, leur création est facilitée car l'attaquant connait les entrailles de l'algorithme ciblé. De nombreuses attaques utilisent le gradient de l'algorithme de manière itérative, calculé par le mécanisme de propagation arrière (*backpropagation*) pour 'inverser' localement un réseau de neurones. Les attaques connues pour leurrer la classification d'images (de la plus simple à la plus évoluée) : FGSM [8], PGD [15], DDN [18], BP [24] et CW [5].

Le constat est le suivant : les perturbations sont de faible amplitude et ainsi à peine visibles à l'oeil nu. Cette extrême sensibilité des réseaux est bien sûr une vulnérabilité quand ceux-ci sont utilisés à des fins de sécurité. Plus largement encore, les données adverses remettent en cause le terme d' Intelligence Artificielle. La communauté vision par ordinateur a réussi à créer des algorithmes neuronaux s'acquittant de la tache difficile de classification mieux que l'humain (plus rapide et avec moins d'erreurs sur le benchmark ImageNet ILSVRC). Ces algorithmes

* Thèse financée par DGA / Inria.

sont donc a priori dignes du label 'Intelligence Artificielle'. Or, les images adverses sont des exemples où ces algorithmes se trompent quasi systématiquement alors qu'aucun humain n'aurait commis d'erreur.

Les données adverses sont un phénomène à la mode en recherche. La littérature foisonne de travaux proposant des attaques, des contre-attaques ou des explications théoriques de la vulnérabilité des réseaux. La communauté s'est aussi organisée en proposant la compétition [11] liée à la conférence annuelle NeurIPS. Les attaques y sont comparées en terme de distortion, probabilité de succès et temps de calcul.

L'idée de ce papier est simple : Dans la grande majorité de ces travaux, et y compris dans cette compétition NeurIPS, **les images adverses ne sont en fait pas des images**. La section 2 introduit quelques notations et défend le constat ci-dessus. Elle montre les impacts de cette brèche. Pallier ce problème n'est pas si simple, la section 3 propose un post-traitement à appliquer après une attaque pour s'assurer que le résultat est bien avant tout une image, qui soit adverse et dont la perturbation est invisible. Ces mécanismes sont inspirées de la dissimulation de l'information. Ils aident à rendre la perturbation invisible que ce soit à l'oeil nu (comme en tatouage numérique) ou statistiquement (comme en stéganographie). Cette publication est une synthèse des papiers [3, 2].

2 Le constat

2.1 Notations

Soit I une image composée de 3 canaux couleurs, de L lignes et C colonnes. Les valeurs des $3LC$ pixels sont codées par des entiers entre 0 et 255. Ainsi, l'image I est un objet discret qui vit dans l'ensemble $\{0, \dots, 255\}^{3LC}$. On considère un classifieur d'image : à une image I, il associe une classe $c(I)$ parmi C classes apprises lors de son entrainement. Ainsi, $c(I) \in \{1, 2, \dots, C\}$.

En général, ce classifieur est composé de trois briques. Un premier traitement normalise l'image : elle calcule une représentation $x = T(I)$. Souvent $x \in [0, 1]^{3LC}$, mais cela peut aussi être $[-1, 1]^{3LC}$. Parfois, $T(\cdot)$ est une simple division par 255 de la valeur des pixels, mais cela peut être $x_i = \alpha(I_i - \beta)$ avec (α, β) dépendant du canal couleur. Ce qui est sûr, c'est que ce pré-traitement est ad-hoc et qu'il est fixe. Il ne fait pas partie de l'apprentissage. Ce traitement d'image est déconsidéré par la communauté *machine learning* car il y a rien à apprendre.

La deuxième brique est le réseau de neurones qui prend en entrée x et donne en sortie les logits $y \in \mathbb{R}^C$. Plus y_i est grand plus l'image est probablement de classe i. La dernière brique est l'opérateur softmax qui normalise y en un vecteur de probabilités $p \in [0, 1]^C$ avec $\sum_{i=1}^{C} p_i = 1$. Ce dernier étage donne aussi la classe prédite comme étant celle qui a la plus grande probabilité associée : $c(I) = \arg\max_i p_i$.

2.2 La brèche

Partant d'une image originale I_o de la classe c_o que l'on suppose bien classée ($c(I_o) = c_o$), l'attaquant souhaite trouver une image adverse I_a proche de I_o mais mal classée : $c(I_a) \neq c_o$ (attaque non ciblée). La majorité des papiers définissent l'image adverse optimale par l'équation:

$$x_a^\star = \arg \min_{x \in [0,1]^{3LC},\, c(x) \neq c_o} \|x - x_o\|. \tag{1}$$

La vraie définition devrait être :

$$I_a^\star = \arg \min_{I \in \{0,1,\ldots,255\}^{3LC},\, c(I) \neq c_o} \|I - I_o\|. \tag{2}$$

Sur les 25 papiers traitant d'images adverses aux conférences CVPR 2019 et ECCV 2019, 88% utilisent (1) au lieu de (2). Pour eux, une donnée adverse est un tenseur x_a (une matrice 3D) contenant des réels codés sur 4 octets en virgule flottante, et non des entiers entre 0 et 255. A notre connaissance, un seul papier propose une attaque (DDN [18]) produisant directement des images quantifiées. Le reste des papiers ne commet pas cette erreur car ils étudient les images adverses dans le monde physique : celles-ci sont imprimées et donc quantifiées.

De la même manière, la règle [11] de la compétition internationale du challenge NeurIPS est sidérante : "*The adversary has direct access to the actual data fed into the model* [c'est-à-dire x]. *In other words, the adversary can choose specific float32 values as input for the model*". Il y a clairement une mauvaise analyse des menaces. Le scénario 'boite blanche' signifie que l'attaquant cible un classifieur (disponible sur un site web, dans un produit fermé *etc*) dont il possède une copie de l'algorithme qu'il est libre d'analyser dans son garage. Cependant, à l'extérieur du garage, c'est une image I_a qu'il doit fournir en entrée à ce classifieur cible ; il ne peut pas modifier ses variables internes. Or, le pré-traitement $T(\cdot)$ fait partie intégrante du classifieur et l'attaquant ne peut pas directement imposer un x_a.

Pour justifier ce choix étonnant, NeurIPS écrit "*In a real world, this might occur when an attacker uploads a PNG file to a web service, and intentionally designs the file to be read incorrectly.*" On imagine que les auteurs pensent à des attaques par dépassement de pile. Ce sont des attaques informatiques qui menacent le décodage du fichier (et non pas l'algorithme de classification) et on connait depuis longtemps des contre-mesures.

2.3 L'impact

L'impact de cette mauvaise définition est multiple.

Tout d'abord, il n'est pas trivial de créer de images adverses quantifiées. On pourrait croire qu'il suffit de trouver x_a, de le faire passer dans la fonction réciproque $T^{-1}(\cdot)$ (linéaire, souvent multiplication par 255), et enfin d'arrondir chaque pixel à l'entier le plus proche entre 0 et 255. Ce procédé est hasardeux. La

caractéristique des données adverses est leur faible distortion. Autrement dit, la perturbation $T^{-1}(x_a - x_o) = T^{-1}(x_a) - I_o$ est de très faible amplitude et l'arrondi à l'entier le plus proche la détruit. La perturbation est quantifiée à 0 sur de nombreux pixels et après quantification, l'image n'est plus adverse. L'article [3] donne une justification théorique. Les papiers de la littérature sont généralement illustrés par des images attaquées. Elles ont forcément été quantifiées, donc ces images présentées comme adverses ne le sont peut-être pas !

La quantification est une contrainte supplémentaire forte. Il est clair qu'en augmentant l'amplitude de la perturbation pour qu'elle résiste à la quantification, on augmente les chances d'obtenir une image quantifiée et toujours adverse. Mais est-ce le meilleur procédé ? La perturbation ne devient-elle pas visible ?

Cette brèche empêche de comparer les classifieurs. Comme dit auparavant, tous n'ont pas le même pré-traitement $T(\cdot)$. Ainsi, mesurer la vulnérabilité d'un classifieur par la distortion moyenne $\|x_a - x_o\|$ ne veut rien dire car ce n'est pas une mesure invariante. Par exemple, il est facile d'augmenter ou diminuer cette mesure arbitrairement: en substituant à $T(\cdot)$ le pré-traitement $T'(\cdot) = \alpha T(\cdot)$ et en multipliant tous les poids de la première couche du réseau par $1/\alpha$, alors on obtient un nouveau classifieur qui fait exactement les mêmes prédictions mais dont la 'vulnérabilité' est multipliée par α.

Cette littérature contient autant de papiers proposant des attaques que des contre-attaques. Ces auteurs montrent que i) leur défense ne dégrade pas les performances du classifieur sur des images originales (donc quantifiées), ii) qu'elle est efficace en soumettant des images attaquées (donc, non quantifiées). Il est amusant de voir que simplement détecter si les données d'entrée sont quantifiées bloquerait la plupart des attaques.

3 La quantification comme un post-traitement

Notre idée n'est pas de construire une nouvelle attaque mais un post-traitement qui quantifie intelligemment une image attaquée. Le schéma est le suivant : partant d'une image I_o, le pré-traitement calcule $x_o = T(I_o)$, une attaque de la littérature donne x_a, et notre post-traitement calcule $\tilde{I}_a = T^{-1}(x_a) \in \mathbb{R}^{3LC}$, puis quantifie intelligemment en $I_a = Q(\tilde{I}_a) \in \{0, 1, \ldots, 255\}^{3LC}$.

3.1 Compromis distortion - adversité

Supposons que l'attaque de la littérature ait réussi, $c(\tilde{I}_a) = c_a \neq c_o$, alors notre post-traitement doit trouver I_a quantifiée proche de I_o tout en restant adverse, c'est à dire de classe c_a. Idéalement, on veut résoudre le problème :

$$I_a = \arg \min_{I \in \{0,\ldots,255\}^{3LC},\, c(I)=c_a} \|I - I_o\|^2. \tag{3}$$

Une première étape propose une formulation Lagrangienne:

$$I_a^{(\lambda)} = \arg \min_{I \in \{0,\ldots,255\}^{3LC}} \|I - I_o\|^2 + \lambda L(I) \tag{4}$$

avec $L(I) = p_{c_0}(I) - p_{c_a}(I)$. En clair, $L(I)$ est la différence entre la probabilité prédite pour la classe originale c_o et celle de la classe c_a. L'image I est adverse si $L(I) < 0$ car alors $c(I) \neq c_o$. Plus $L(I)$ est négatif, plus le classifieur est confiant dans son erreur.

Pour $\lambda = 0$, seule la distortion compte dans le problème (4) et la solution évidente $I_a^{(0)} = I_o$ n'est pas adverse. Pour λ très grand, seule l'adversité compte, et $I_a^{(\infty)}$ est une image (trop ?) adverse mais très éloignée de I_o. Il faut faire un compromis par une recherche dichotomique sur λ. Pour un λ donné, calculer $I_a^{(\lambda)}$ et voir si cette image est adverse. Si oui, on peut baisser λ et voir si on obtient une nouvelle image adverse et plus proche de I_o, sinon on augmente λ.

3.2 Linéarisation

Supposons que l'on se donne q degrés de liberté par pixel (q entier pair). Le pixel $\tilde{I}_{a,i}$ n'est a priori pas un entier et on va le quantifier sur un des q entiers les plus proches : $\lfloor \tilde{I}_{a,i} \rfloor + \{-(q/2-1), \ldots, -1, 0, 1, ..., q/2\}$ (sauf si $\tilde{I}_{a,i}$ est trop proche de 0 ou 255). Pour λ donné, résoudre (4) demande de passer en revue les q^{3LC} combinaisons possibles, soit une complexité exponentielle avec le nombre de pixels.

La linéarisation $L(I) \approx L(\tilde{I}_a) + (I - \tilde{I}_a)^\top \nabla_I L(\tilde{I}_a)$ simplifie le problème en

$$\arg\min \sum_{i=1}^{3LC} (I_i - I_{o,i})^2 + \lambda (I_i - \tilde{I}_{a,i}) g_i + cte \tag{5}$$

où g_i est la i-ème composante du gradient $\nabla_I L(\tilde{I}_a)$. Ce gradient est facilement calculé par la propagation arrière. Cette approximation a cassé un problème NP en une suite de $3LC$ problèmes très simples puisqu'on peut minimiser chaque terme de la somme indépendamment.

3.3 Généralisation

La distortion de la perturbation est mesurée jusqu'à présent par la norme Euclidienne au carré, $\|I - I_o\|^2$. Pour de faible amplitude, cette mesure de la visibilité n'est pas si mal. On peut la remplacer par n'importe quelle autre distance du moment qu'elle reste séparable de la forme $d(I, I_o) = \sum_i w_i(I_i, I_{o,i})$. Nous pensons notamment à des coûts utilisés en stéganographie comme HILL [12], MiPod [19], ou GINA [13, 21]. Ils modélisent non pas la distortion visible mais la détectabilité statistique de la perturbation $I - I_o$.

L'algorithme est alors simple : i) calculer la fonctionnelle $w_i(I_i, I_{o,i}) + \lambda(I_i - \tilde{I}_{a,i})g_i$ pour les q valeurs possibles de I_i, ii) trouver pour quelle valeur la fonctionnelle est à son minimum, iii) itérer sur tous les pixels. D'où une complexité linéraire en nombre de pixels : $O(3LCq \log q)$. Quelques astuces sont possibles, notamment si $w_i(I_i, I_{o,i})$ a une forme quadratique comme dans (5), alors le minimum recherché a une expression simple [3, 2], ce qui évite un tri rapide en $O(q \log q)$ à chaque pixel.

4 Investigation expérimentale

4.1 Protocole

Nos expériences utilisent les images de la compétition NeurIPS [11]. C'est en fait un sous-ensemble d'ImageNet. Les images ont la taille 224×224. Nous testons différents réseaux: ResNet-18, ResNet-50 [9], ResNet-50R qui est une version robustifiée par entrainement adverse [15], mais aussi les tout nouveaux EfficientNet-b0 [20] et sa version robustifiée [23].

Nous mesurons la distortion par la norme Euclidienne normalisée au nombre de pixels $\overline{d} = \|I_a - I_o\|/\sqrt{3LC}$. Les attaques étant des processus à plusieurs paramètres, pour chaque image nous essayons plusieurs jeux de paramètres et retenons celui qui offre une image adverse avec la plus petite distortion. Nous introduisons le concept de caractéristique $\overline{d} \to P_{suc}(\overline{d})$, probabilité que l'attaque réussisse avec une distortion inférieure à $\overline{d}$.

4.2 Arrondir à l'entier le plus proche ne fonctionne pas

La première expérience compare les attaques classiques de la littérature avec et sans quantification. La figure 1 montre clairement que la quantification naïve par arrondi à l'entier le plus proche est une catastrophe : plus aucune image n'est adverse sauf si la distortion est supérieure à 1. Notre quantification est bien plus performante.

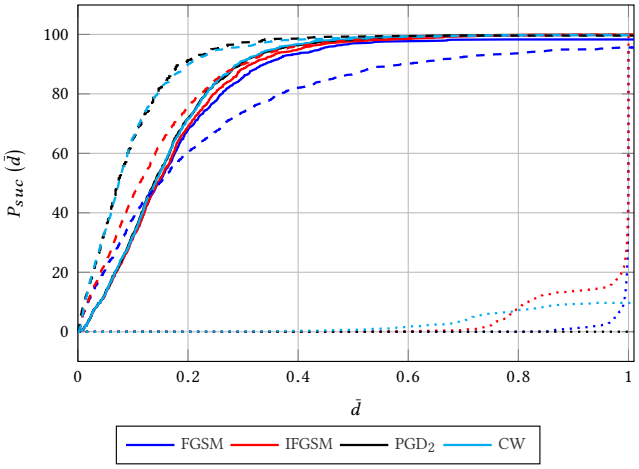


Fig. 1. Caractéristique des attaques FGSM, IFGSM, PGD et CW contre ResNet-18 : sans quantification (tiret), avec quantification par arrondi (pointillé), ou avec la quantification proposée (plein).

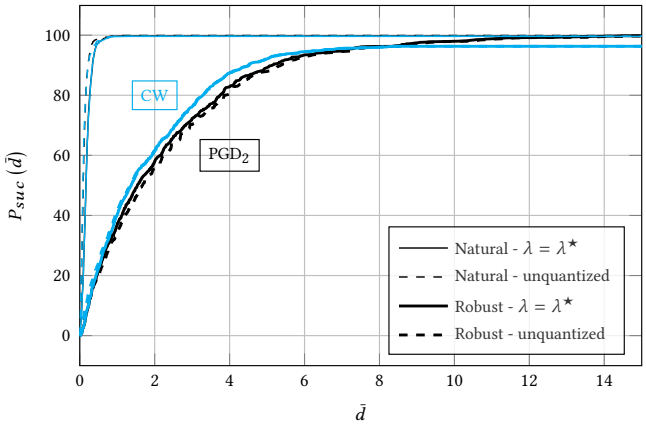


Fig. 2. Caractéristiques des attaques PGD et CW contre ResNet-50 version naturelle et version robustifiée, avec et sans notre quantification.

4.3 La quantification n'est pas une contrainte forte

La deuxième expérience montre que la quantification n'est pas une contrainte forte au sens où les caractéristiques sans ou avec *notre* quantification sont très proches. La figure 2 montre aussi que la version robuste ResNet-50R est effectivement moins vulnérable : l'attaquant doit consommer 10 fois plus de distortion pour leurrer le classifieur.

4.4 Des images adverses non détectables

Cette expérience utilise les nouveaux réseaux EfficientNet (en version naturelle et robuste) et l'attaque PGD$_2$ qui offre des performances similaires à CW tout en étant plus rapide. Le post-traitement quantifie les pixels pour faire un compromis entre adversité et distortion Euclidienne (1ère ligne du tableau) ou détectabilité d'un modèle stéganographique (HILL, MiPod, ou Gina). Pour voir si cette recherche d'indétectabilité est concluante, le défenseur utilise des détecteurs d'images adverses : SCRMQ1 [14] et SRNet [4]. Ces détecteurs sont réglés pour commettre des faux positifs avec une probabilité de 5%.

Le tableau 1 montre que la version robuste du réseau est effective : les attaques sont moins puissantes (probabilité de succès plus petite) tout en consommant plus de distortion. La quantification qui minimise la distortion est celle que l'on a vue jusqu'à présent (1ère ligne du tableau). Cependant, elle est très détectable (à $\approx 90\%$ avec SRNET). Le détecteur SCRMQ1 est moins puissant que SRNet. Les quantifications qui utilisent des couts stéganographiques consomment plus de distortion mais sont moins détectables (avec GINA, 4 images sur 5 ne sont pas détectées). Ceci s'explique par des coûts non stationnaires. L'image originale est analysée et les coûts pour les pixels dans des régions texturées de grande dynamique sont inférieurs à ceux des régions uniformes. Ainsi, la perturbation se concentre dans les régions texturées, qui cachent / masquent ce signal faible de manière perceptuelle et statistique.

Table 1. Détectabilité de l'attaque PGD_2 contre EfficientNet-b0 naturel ou robuste, en fonction de la quantification en post-traitement avec q degrés de liberté par pixel.

	q	P_{suc} (%)		$\overline{d}$		SCRMQ1(%)		SRNet(%)	
		Nat	Rob	Nat	Rob	Nat	Rob	Nat	Rob
$\overline{d}$	2	88.0	71.8	**0.22**	**0.29**	81.2	76.4	93.3	87.5
HILL	2	88.0	71.8	0.24	0.30	74.8	66.3	86.1	77.6
HILL	4	**88.8**	**72.6**	0.27	0.33	72.4	72.4	85.5	72.3
MiPod	2	87.9	71.8	0.26	0.32	74.9	64.3	84.0	76.1
MiPod	4	88.2	72.2	0.29	0.35	72	57.0	82.6	67.5
GINA	2	88.0	71.8	0.43	0.47	5.4	**3.0**	44.2	33.5
GINA	4	88.2	71.9	0.60	0.63	**3.8**	3.1	**20.7**	**14.2**

5 Conclusion

Ce papier a exploré le jeu entre l'attaquant et le défenseur lorsque ces acteurs utilisent les armes les plus récentes : classifieur EfficientNet, attaques CW, détecteur SRNET, et cout stéganographie GINA. La conclusion est sans appel : l'attaquant gagne le jeu. Il a 70% de chances de trouver une image qui leurre à la fois le classifieur et le détecteur en un temps raisonnable. Mais ce jeu du gendarme et du voleur n'est pas fini. Nos résultats sont donnés pour des détecteurs qui n'ont jamais vu d'images adverses a la 'GINA'. La prochaine étape est de les nourrir de ces images à l'apprentissage.

References

1. Behjati, M., Moosavi-Dezfooli, S., Baghshah, M.S., Frossard, P.: Universal adversarial attacks on text classifiers. In: ICASSP. pp. 7345–7349. IEEE (2019)
2. Bonnet, B., Furon, T., Bas, P.: Adversarial images through stega glasses. In: submitted to IEEE WIFS'20 (2020)
3. Bonnet, B., Furon, T., Bas, P.: What if adversarial samples were digital images? In: Proc. of ACM IH&MMSec '20. pp. 55–66 (2020). https://doi.org/10.1145/3369412.3395062
4. Boroumand, M., Chen, M., Fridrich, J.: Deep residual network for steganalysis of digital images. IEEE Transactions on Information Forensics and Security **14**(5), 1181–1193 (2018)
5. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE Symp. on Security and Privacy (2017)
6. Carlini, N., Wagner, D.A.: Audio adversarial examples: Targeted attacks on speech-to-text. In: IEEE Symposium on Security and Privacy Workshops. pp. 1–7. IEEE Computer Society (2018)
7. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.: Adversarial attacks on deep neural networks for time series classification. In: IJCNN. pp. 1–8. IEEE (2019)
8. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6572

9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (June 2016). https://doi.org/10.1109/CVPR.2016.90
10. Jiang, L., Ma, X., Chen, S., Bailey, J., Jiang, Y.: Black-box adversarial attacks on video recognition models. In: ACM Multimedia. pp. 864–872. ACM (2019)
11. Kurakin, A., Goodfellow, I., Bengio, S., Dong, Y., Liao, F., Liang, M., Pang, T., Zhu, J., Hu, X., Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A., Huang, S., Zhao, Y., Zhao, Y., Han, Z., Long, J., Berdibekov, Y., Akiba, T., Tokui, S., Abe, M.: Adversarial attacks and defences competition (2018)
12. Li, B., Wang, M., Huang, J., Li, X.: A new cost function for spatial image steganography. In: Image Processing (ICIP), 2014 IEEE International Conference on. pp. 4206–4210. IEEE (2014)
13. Li, B., Wang, M., Li, X., Tan, S., Huang, J.: A strategy of clustering modification directions in spatial image steganography. Information Forensics and Security, IEEE Trans. on **10**(9) (2015)
14. Liu, Y., Moosavi-Dezfooli, S.M., Frossard, P.: A geometry-inspired decision-based attack. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
15. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings (2018), https://openreview.net/forum?id=rJzIBfZAb
16. Martins, N., Cruz, J.M., Cruz, T., Abreu, P.H.: Adversarial machine learning applied to intrusion and malware scenarios: A systematic review. IEEE Access **8**, 35403–35419 (2020)
17. Qin, Y., Carlini, N., Cottrell, G.W., Goodfellow, I.J., Raffel, C.: Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In: ICML. Proceedings of Machine Learning Research, vol. 97, pp. 5231–5240. PMLR (2019)
18. Rony, J., Hafemann, L.G., Oliveira, L.S., Ayed, I.B., Sabourin, R., Granger, E.: Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
19. Sedighi, V., Cogranne, R., Fridrich, J.: Content-adaptive steganography by minimizing statistical detectability. Information Forensics and Security, IEEE Transactions on **11**(2), 221–234 (2016)
20. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv (2019)
21. Wang, Y., Zhang, W., Li, W., Yu, X., Yu, N.: Non-additive cost functions for color image steganography based on inter-channel correlations and differences. IEEE Trans. on Information Forensics and Security (2019)
22. Wei, X., Liang, S., Chen, N., Cao, X.: Transferable adversarial attacks for image and video object detection. In: IJCAI. pp. 954–960. ijcai.org (2019)
23. Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., Yuille, A.L.: Improving transferability of adversarial examples with input diversity. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
24. Zhang, H., Avrithis, Y., Furon, T., Amsaleg, L.: Walking on the edge: Fast, low-distortion adversarial examples. IEEE Trans. on Information Forensics and Security (2020)

DECOV et Météo COVID : l'IA pour la modélisation et l'aide au déconfinement COVID-19

Tristan BITARD-FEILDEL[1], Tristan CHARRIER[1], Benjamin FARCY[2] et Silvia GIL CASALS[2]

[1] DGA MI, tristan.charrier@def.gouv.fr, France
[2] Capgemini, benjamin.farcy@capgemini.com, France

Résumé. La crise du COVID-19 a nécessité l'application rapide de mesures de contrôle de la pandémie. Ces mesures ont eu un impact fort sur les capacités hospitalières et l'économie du pays. Ce projet présente différents outils permettant d'aider les autorités publiques dans leurs décisions pour contrôler au mieux l'évolution du COVID-19 sur leurs territoires.
Des modèles épidémiologiques et économiques associées à des outils de visualisation présentent l'impact de mesures de contrôles sur l'évolution du COVID-19 dans un territoire. Par ailleurs, des algorithmes d'intelligence artificielles (IA) combinés à ces modèles permettent de rechercher le meilleur ensemble de mesures afin de limiter la propagation du virus et les pertes économiques.
Ce projet montre l'intérêt des outils d'aide à la décision pour les autorités publiques par leurs capacités de projections de scénarii et leurs aptitudes à proposer des solutions non-triviales grâce à l'IA.

Mots-clefs: COVID-19, Déconfinement, Intelligence Artificielle.

1 Introduction

L'apparition en France du virus SARS-Cov-2 responsable du COVID-19 en début d'année 2020 a nécessité un suivi au plus près de l'évolution de la pandémie sur les différents territoires et la mise en place de mesures exceptionnelles pour contenir cette évolution. La mesure la plus significative, le confinement de la population française entre le 17/03 et le 11/05/2020, a eu plusieurs impacts dans notre société notamment au niveau sanitaire et économique. Il a permis de drastiquement réduire la propagation du virus et de soulager les systèmes hospitaliers mais a eu un fort impact économique par la réduction d'activité. Une solution de contrôle pharmaceutique n'étant pas disponible à un horizon proche, la sortie du confinement a suscité des interrogations concernant la capacité de suivi sur les territoires de l'état de la pandémie, ses évolutions possibles ainsi que les mesures sanitaires à adopter afin d'empêcher une reprise de propagation tout en relançant l'économie. Les travaux présentés ici combinent de la modélisation et des algorithmes d'intelligence artificielle (IA) pour aider à répondre à ces challenges.

Premièrement, des modèles épidémiologiques à compartiments issus des travaux du laboratoire des maladies infectieuses de Montpellier [1] et l'institut Pasteur [2] ont été

adaptés et un troisième a été développé comme expliqué en Section 2. Ceci permet d'évaluer la pandémie dans le passé et de projeter les propagations envisageables suivant différents scénarios.

Au second trimestre 2020 le PIB français a chuté de 13.8% [3] principalement dû au COVID-19. Deuxièmement, un modèle économique détaillé en Section 3 est développé afin d'évaluer la perte de points de PIB en fonction des différentes mesures sanitaires appliquées et de leurs durées.

La recherche d'équilibre entre la meilleure prise en charge médicale possible des patients et le maintien de l'activité économique du pays tout en cherchant à enrayer la propagation du virus est un exercice difficile. Pour y répondre, DGA-MI a développé DECOV, un outil IA d'aide à la prise de décision développé en Section 4. L'outil recherche des combinaisons de mesures offrant de bons compromis entre les objectifs sanitaires et économiques projetés à partir des modèles épidémiologiques et économiques et présente un front de Pareto des meilleures combinaisons.

A la sortie du confinement, ces travaux ont été repris et adaptés par Capgemini au profit de la région Grand Est afin d'aider la région dans le suivi de la pandémie sur ses territoires et sur ses politiques sociétales. Une suite d'outils de visualisation pour l'aide à la décision, Météo COVID, a été développée comme détaillée en Section 5.

Ce projet montre l'intérêt de l'IA dans la gestion d'une crise par la recherche de solutions non triviales parmi un ensemble de mesures possibles très grand. Les travaux apportés par Capgemini ont permis la mise en production de l'outil au sein de la région Grand Est. Le code associé aux travaux de DGA-MI est mis à disposition en *open source* sous licence GNU GPL-v3 à l'adresse https://gitlab.com/covid_dia/deconf [4].

2 Modèles épidémiologiques

2.1 Modèles épidémiologiques à compartiments

Deux modèles ont été retenus et un troisième a été développé. Ces modèles font partie de la classe des modèles épidémiologiques à compartiments [5] [6] [7], approche éprouvée sur de précédentes épidémies [8]. Les modèles choisis comme référence sont le modèle Alizon [1] du laboratoire des maladies infectieuses de Montpellier prenant en compte de différents degrés de sévérité de la maladie, et un modèle basé sur celui de Salje de l'institut Pasteur [2] introduisant une stratification en âge de la population (dénommé *Salje* par la suite). Il a en effet été observé que le COVID-19 atteint les individus et se transmet différemment selon l'âge. Un troisième modèle fusionnant les avantages de ces deux modèles de référence a été créé afin de tirer parti au maximum de toutes les données disponibles et de modéliser le plus finement possible l'épidémie.

Toute la population initiale, non immunisée, susceptible d'être infectée est représentée par un compartiment initial noté S. Les individus de cette population suivent un parcours dépendant du modèle. Une personne infectée termine inévitablement dans l'un des deux compartiments définitifs : guéri avec immunité (R) ou décédé (D). Les transitions de population entre ces compartiments sont régies par un système d'équations

différentielles ordinaires paramétrées. Les paramètres sont de deux familles : biologique reflétant l'évolution de la maladie (e.g. durée d'incubation) et transmission transcrivant les contacts entre individus et la probabilité d'infection. Ce sont ces paramètres de transmissions qui peuvent être mitigés par des mesures sanitaires.

Un schéma général des modèles est donné **Fig. 1**. Pour le modèle Alizon **(a)**, la population S exposée suit deux chemins distincts selon la sévérité de la maladie, *mild* et *severe* nécessitant une hospitalisation. Les populations sont d'abord exposées et non infectieuses (Em, Es), puis asymptomatiques et infectieuses (Am, As), puis symptomatiques et infectieuses (Im, Is). Le chemin non virulent aboutit à une guérison Rm. Le chemin virulent aboutit à une guérison Rs ou un décès D.

Le modèle de *Salje* **(b)** ne distingue pas directement les populations selon la sévérité dès l'exposition, mais à partir de l'état symptomatique et infectieux (I1). De cet état une partie de la population va guérir (R) ou voir son état s'aggraver, nécessitant une hospitalisation (I2) et menant à la guérison (R) ou au décès (D). L'apport majeur de ce modèle est la subdivision en sous compartiments, non représentés sur le schéma, pour chaque strate d'âges de la population afin de transcrire les différentes probabilités observées d'hospitalisation en service de réanimation et de décès [2] selon l'âge des patients. La transmission différenciée entre les strates d'âges est également prise en compte par une matrice de contact dérivée de Béraud et al [9].

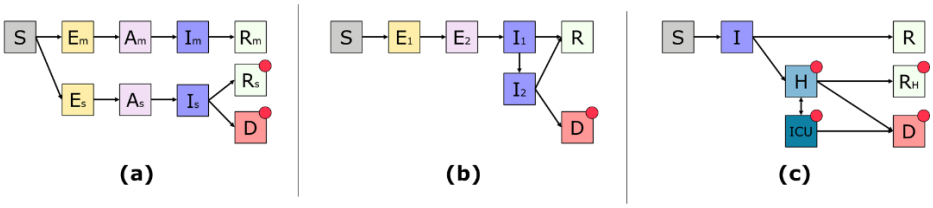


Fig. 1. Représentation graphique des trois modèles, les points rouges correspondent aux compartiments dont les valeurs sont mesurables avec les données ouvertes [10]

Un modèle complémentaire **(c),** noté SIHICURD**,** a été développé. Les compartiments sont stratifiés par classes d'âge comme celles de Salje et al [2]. Ce modèle reflète le parcours de soin pour utiliser au mieux les données provenant des hôpitaux afin de calibrer et vérifier le modèle. La population infectée (I) peut soit guérir directement (R) soit nécessiter une hospitalisation (H) ou encore s'aggraver rapidement et résulter en décès (D). L'état des patients peut soit s'améliorer et résulter en une guérison par retour à domicile (Rh), soit s'aggraver et nécessiter des soins intensifs (ICU). Les patients en soins intensifs peuvent soit retourner en hospitalisation classique (H), soit décéder (D).

2.2 Méthodes de calibration des modèles à compartiments

De nombreuses méthodes de calibration et d'initialisation des modèles épidémiologiques à compartiments existent [7]. L'approche retenue est d'optimiser numériquement l'ensemble des paramètres biologiques, de transmission et l'état initial du système d'équations différentielles ordinaire en les contraignant dans des plages de valeurs reflétant la réalité observée [2]. Ces paramètres sont définis constants par morceaux en

fonction du temps pour rendre compte des différents stades de la pandémie depuis son arrivée sur le territoire, comme suggéré par Alizon [1]. Les zones temporelles reflètent les différentes mesures : propagation libre puis pré-confinement le 28/02, confinement le 18/03, déconfinement phase 1 le 11/05, phase 2 à partir du 03/06.

Une fonction de coût est définie pour chaque modèle, afin de minimiser l'écart des compartiments ayant des valeurs mesurables (cf **Fig. 1**). La fonction de coût pour le modèle SIHICURD, de type MAPE pondérée, est décrite en (α). Les données utilisées pour la calibration [10] débutent le 18/03/2020, sont rafraichies quotidiennement et représentent les retours à domicile, les décès, l'occupation des hôpitaux et des soins intensifs. Des fonctions similaires sont utilisées pour les deux autres modèles.

$$e = \frac{1}{t_{end} - t_{start}} \sum_{t=t_{start}}^{t_{end}} \sum_{s=strat_{[0,20)}}^{strat_{[80,+)}} \left(\begin{array}{l} \omega_H \times \frac{\left|H_{ref}^{t,s} - H^{t,s}\right|}{H_{ref}^{t,s}} + \omega_{ICU} \times \frac{\left|ICU_{ref}^{t,s} - ICU^{t,s}\right|}{ICU_{ref}^{t,s}} \\ + \omega_D \times \frac{\left|D_{ref}^{t,s} - D^{t,s}\right|}{D_{ref}^{t,s}} + \omega_{Rh} \times \frac{\left|Rh_{ref}^{t,s} - Rh^{t,s}\right|}{Rh_{ref}^{t,s}} \end{array} \right) \quad (\alpha)$$

Cette fonction de coût est minimisée numériquement avec une approche en deux temps permettant de modéliser les évolutions de transmission selon les territoires tout en gardant une cohérence de l'évolution de la maladie chez les populations affectées. Premièrement, une calibration est faite à l'échelle de la France pour fixer les paramètres biologiques représentatifs de l'évolution médicale de l'épidémie. Ensuite, les paramètres de transmission sont calibrés sur les sous-territoires (départements).

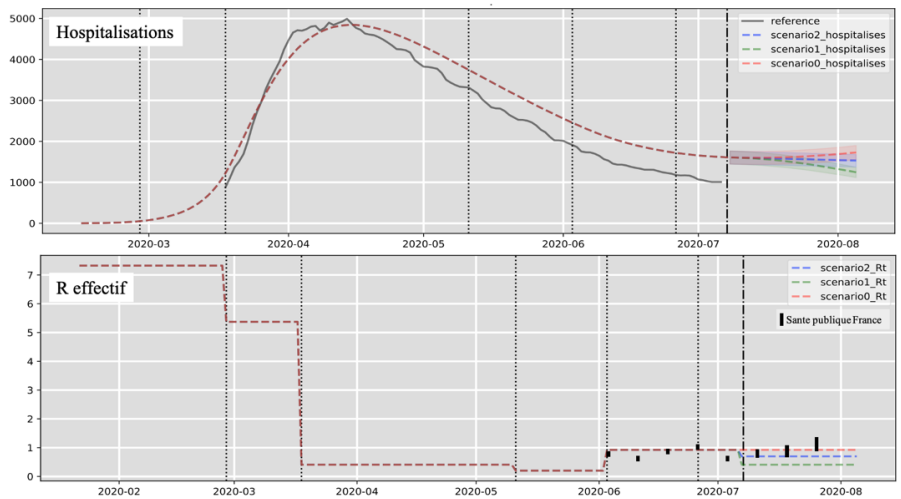


Fig. 2. Résultats de calibration sur la région Grand Est : hospitalisations et R effectif. Calibration faite sur données du 18/03/2020 au 20/06/2020, projections à partir du 07/07/2020. Scénarios de projection : 0 continuité, 1 confinement maximal, 2 confinement partiel.

Un exemple de résultats et de projection est donné **Fig. 2** pour la région Grand Est avec le modèle SIHICURD. La tendance du nombre d'hospitalisés est bien capturée par

le modèle. A partir des paramètres de transmission calibrés [2] est calculé le nombre de reproduction effectif (R), i.e. le nombre moyen d'infections secondaires provenant d'un seul infecté. Cet indicateur évolue de façon attendue pendant les différentes phases, maximal en début d'épidémie et décroissant avec l'application des mesures restrictives. Cependant la baisse en phase 1 de déconfinement est contre-intuitive. Ensuite, l'indicateur de transmission augmente jusqu'à des niveaux en accord avec l'estimation hebdomadaire de Santé publique France [10], faite avec une méthode différente.

3 Modèle économique

Ce modèle estime la perte de points de PIB par région. Pour cela, est utilisée une estimation **Tableau *1*** de ces pertes par secteur par rapport au PIB global [11], sous l'hypothèse que la part du PIB pour une branche dans une région est directement proportionnelle au ratio des employés de ce secteur.

Branches	Fermeture des écoles	Baisse de la demande	Autres chocs	Contributions à la croissance
Agriculture	-5,7	-2,9	-4,6	-0,2
Industrie	-4,4	-25,3	-3,5	-4,6
Construction	-1,3	-46,8	-2,6	-2,9
Services marchands	-3,6	-29,9	-5,2	-21,6
Services non marchands	-3,4	-1,9	-4,6	-2,3
Ensemble de l'économie	-3,6	-23,3	-4,7	-31,6

Tableau 1: Pertes du PIB par secteur et mesure *[11]*

Les données de répartition géographique des employés et des secteurs d'activité sont récupérées de l'INSEE. La part du PIB de chaque branche d'activité, $part(PIB_{branche})$, est calculée avec le tableau 1 de [12]. L'équation utilisée est donc :

$$PIB_{(région,branche)} = PIB_{national} \times part\ (PIB_{branche}) \times \frac{nb\ employé\ branche\ région}{nb\ employé\ branche\ France}$$

4 Calcul de politiques optimales

La crise COVID-19 a nécessité la mise en place de mesures sanitaires. Plusieurs niveaux de restrictions sont associés à chaque mesure (exemple : interdiction des rassemblements de plus de 10 personnes, de plus de 100 personnes etc.), cf Section 4.3. Il est alors possible d'agir sur ces mesures pour contrôler l'évolution de l'épidémie et du PIB. L'impact des mesures sur le modèle épidémiologique est effectué en agissant directement sur la matrice de contacts de la population [2]. En incitant par exemple au télétravail, la probabilité de contacts entre actifs est réduite. L'impact sur le modèle économique est direct par construction du modèle, cf Section 3. Une *politique* correspond à un ensemble de mesures et de niveaux au cours du temps. Les outils IA exacts et approchés proposés permettent de chercher les meilleures solutions.

4.1 Difficulté du problème

Mettre en œuvre un solveur d'optimisation est ici ardu pour plusieurs raisons. Les différents critères à optimiser évoluent différemment. Typiquement, le nombre de malades et la perte de PIB sont des critères antagonistes. Pour définir une politique optimale, on cherche alors un Front de Pareto, i.e. un ensemble de politiques qui ne soient pas strictement moins bonnes sur tous les critères que d'autres politiques. Par ailleurs, pour n mesures, le nombre de politiques possibles est d'au moins 2^n, ce qui induit une explosion combinatoire. Enfin, l'algorithme d'optimisation doit être capable de trouver une solution en ayant uniquement accès aux entrées et sorties du modèle. En effet, dans un contexte où le virus est encore en cours d'étude, il est important de développer un algorithme agnostique au modèle sous-jacent.

4.2 Méthodes implémentées

Les méthodes implémentées dans l'outil se divisent en trois catégories : exactes, approchées et renforcement. Les méthodes exactes se basent sur l'énumération car peu d'hypothèses sur les modèles sont disponibles. Le *bruteforce* consiste à énumérer toutes les politiques possibles, tandis que la méthode *séparation et évaluation* (*Branch & Bound*) arrête l'énumération si un seuil prédéfini est dépassé (e.g. capacité hospitalière). Les méthodes approchées par variations des politiques consistent à partir d'un ensemble aléatoire de politiques données et les faire varier pour trouver une approximation du Front de Pareto. Trois méthodes sont implémentées : l'aléatoire pour référence, *NSGA* [13] un algorithme génétique et *PSO* [14] qui utilise les lois de la physique des particules où chaque politique est considérée comme une particule. Concernant l'apprentissage par renforcement, *l'algorithme PPO* [13] est implémenté. L'algorithme apprend la meilleure politique en monocritère en s'adaptant aux retours du modèle.

4.3 Comparaison des approches

Pour comparer les approches, la zone considérée est la France métropolitaine divisée en départements rouges et verts. Les mesures disponibles sont le traçage des contacts (tests PCR et confinement des symptomatiques et positifs/désactivé), le télétravail (imposé/une semaine deux en alternance/non imposé), la fermeture des écoles (établissements du supérieur et demi-classes pour les écoles/établissements du supérieur/désactivé), interdiction des rassemblements (10 personnes/100 personnes/désactivé), injonction de confinement pour les personnes à risque (activé/désactivé). Les politiques s'étalent sur 15 semaines par blocs de 3 semaines consécutives. Les deux critères sont le pourcentage de décès et la perte de PIB.

A cause de l'explosion combinatoire seules les méthodes approchées sont comparées ici. Aléatoire/NSGA/PSO ont le droit à 500 appels au modèle, soit 20 générations de 25 individus pour NSGA et PSO. L'approche par renforcement a un temps d'apprentissage en amont sur différentes combinaisons de critères et effectue quelques appels

en exécution ensuite. Sur les Fronts de Pareto **Fig. 3** est constaté que NSGA est la meilleure méthode parmi les approchées, ayant une aire minimale sous la courbe. L'apprentissage par renforcement a un front de Pareto peu diversifié car il faut l'entrainer pour chaque combinaison de critères. PSO est globalement moins performant que l'aléatoire.

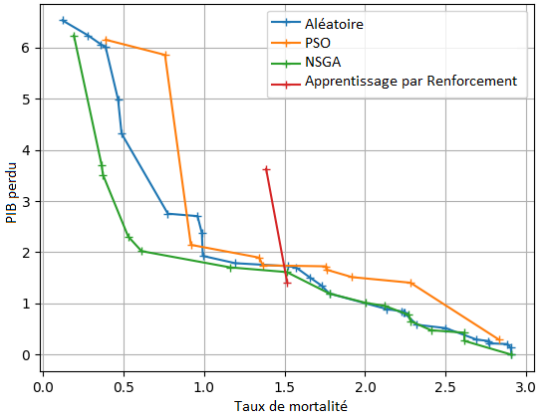


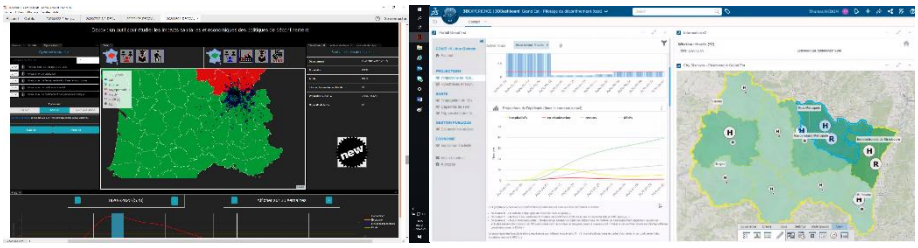
Fig. 3. Fronts de Pareto calculés pour chacune des 4 approches

5 Outils d'aide à la décision

Ces travaux se sont matérialisés dans deux outils d'aide à la décision. Dans un premier temps, orientée par la cellule de crise COVID du Ministère de la Santé, DGA-MI a réalisé DECOV [4], un outil permettant la visualisation de l'état sanitaire des départements français métropolitains, le lancement de simulations épidémiologiques, leurs projections et la recherche de mesures post-confinement optimales et la visualisation des fronts de Pareto associés (**Fig. 4.(a)**).

Ces travaux ont été repris et adaptés au profit de la région Grand Est par Capgemini, aboutissant en Météo COVID, un outil de visualisation d'indicateurs clés de suivi de l'épidémie sur le territoire de la région Grand Est et de projection de l'épidémie et de l'impact potentiel de différents ensembles de mesures. Parmi les scénarii définis par DGA-MI et explorés dans leurs combinaisons par les algorithmes d'optimisation en sont retenus trois principaux pour cet outil : continuité, confinement total et confinement partiel (**Fig. 4.(b)**).

Ces scénarii sont projetés et permettent d'estimer l'impact de ces mesures sur la tendance de l'évolution future de l'épidémie au sein de chaque territoire, comme aperçu **Fig. 4.(b)** pour la région Grand Est. L'impact des scénarios sur le nombre de reproduction effectif projeté et sur les tendances d'hospitalisation est directement appréciable, apportant des informations utiles pour l'aide à la décision.



(a) DECOV **(b) Météo COVID**

Fig. 4. Captures d'écran des outils d'aide à la décision.

6 Conclusions et perspectives

Les outils développés dans ces travaux permettent d'évaluer et de proposer des stratégies non triviales de contrôle de la propagation du virus.

Les modèles épidémiologiques nécessitent encore des améliorations pour rendre compte de la réalité des données observées. Les modèles à compartiments sont limités dans leur représentation de la propagation du virus. Ils peuvent être améliorés en adaptant les compartiments, en enrichissant les données de calibration et en utilisant des phases temporelles continues mais ils resteront des modèles populationnels homogènes. Des modèles multi-agents peuvent apporter dans ce contexte une vision plus précise des déplacements de populations et ainsi tester des mesures de confinement plus fines [15] [16] [17] et étudier des impacts plus locaux sur les territoires.

La mise en place de différentes solutions permet des prises de décisions éclairées à partir d'approches complémentaires. Ce type de projet d'aide à la décision se poursuit également dans d'autres pays, comme celui financé par l'université Princeton [18] permettant la prise en compte de la mutation du virus. Néanmoins, les résultats de ces outils nécessitent un regard critique d'expert afin de bien comprendre leurs limitations.

7 Remerciements

Nous tenons à remercier chaleureusement l'Agence d'Innovation de Défense et sa Cellule de Coordination de l'IA de Défense, plus particulièrement Michaël KRAJECKI et Amélie BARAZZUTTI, l'Université de Reims Champagne-Ardenne, le TGCC, la plate-forme MODCOV19 du CNRS et le Health Data Hub et sa Task Force « Data vs COVID » pour leur soutien au projet DECOV ainsi que la région Grand Est et l'agence d'innovation Grand E-nov pour leur soutien au projet Météo COVID. Le projet a pu se dérouler grâce à la participation pour la DGA d'une équipe de 11 personnes dont font partie Tristan BITARD-FEILDEL et Tristan CHARRIER, et pour Capgemini de Zakaria BOUAYYAD, Moez DRAIEF, Benjamin FARCY, Silvia GIL CASALS, Thomas HEDON et Barnabé LECOUTEUX.

Références

1. S. Alizon, «Rapport 3 sur un modèle de propagation de l'épidémie de COVID-19,» 24 Mars 2020. [En ligne]. Available: https://covid-ete.ouvaton.org/Rapport3_Modele.html.
2. H. Salje, C. T. Kiem, N. Lefrancq, N. Courtejoie, P. Bosetti, J. Paireau, A. Andronico, N. Hoze, J. Richet et C.-L. Dubost, «Estimating the burden of SARS-CoV-2 in France,» *Institut Pasteur, preprint,* 11 mai 2020.
3. «Point de conjoncture, Banque de France,» Banque de France, 31 Juillet 2020. [En ligne]. Available: www.banque-france.fr/statistiques/conjoncture/enquetes-de-conjoncture/point-de-conjoncture. [Accès le 27 Aout 2020].
4. Agence de l'innovation de défense, Ministère des Armées, «Project DECOV,» 30 juin 2020. [En ligne]. Available: https://gitlab.com/covid_dia/deconf. [Accès le juin 2020].
5. L. D. Domenico, G. Pullano, C. E. Sabbatini, P.-Y. Boëlle et V. Colizza1, «Impact of lockdown on COVID-19 epidemic in Île-de-France and possible exit strategies,» *INSERM, Sorbonne Université, Pierre Louis Institute of Epidemiology and Public Health, preprint,* 16 Juillet 2020.
6. D. Efimov et R. Ushirobira, «On an interval prediction of COVID-19 development based on a SEIR epidemic model,» *Inria, Univ. Lille, preprint,* 3 Juin 2020.
7. G. Giordano, F. Blanchini, R. Bruno, P. Colaneri, A. D. Filippo, A. D. Matteo et M. Colaneri, «Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy,» *Nature,* 22 avril 2020.
8. A. B. Gumel, S. Ruan, T. Day, J. Watmough, F. Brauer, P. v. d. Driessche, D. Gabrielson, C. Bowman, M. E. Alexander, S. Ardal, J. Wu et B. M. Sahai, «Modelling strategies for controlling SARS outbreaks,» chez *Proc. R. Soc. Lond. B.*, London, 2004.
9. G. Béraud, S. Kazmercziak, P. Beutels, D. Levy-Bruhl, X. Lenne et N. Mielcarek, «The French Connection: The First Large Population-Based Contact Survey in France Relevant for the Spread of Infectious Diseases.,» *PLoS One.*, p. https://doi.org/10.1371/journal.pone.0133203, July 15, 2015. DO.
10. «Santé publique France,» [En ligne]. Available: https://www.santepubliquefrance.fr.
11. «Evaluation au 30 mars 2020 de l'impact économique de la pandémie de COVID-19 et des mesures de confinement en France,» Département analyse et prévision de l'OFCE, 30 mars 2020. [En ligne]. Available: https://www.ofce.sciences-po.fr/pdf/pbrief/2020/OFCEpbrief65.pdf.

12. «Point de conjoncture du 26 mars 2020,» INSEE, 26 mars 2020. [En ligne]. Available: https://insee.fr/fr/statistiques/fichier/version-html/4471804/Point_de_conjoncture_INSEE_26mars2020_7h30.pdf.
13. J. Schulman, F. Wolski, P. Dhariwal, A. Radford et O. Klimov, «Proximal policy optimization algorithms.,» 2017.
14. R. &. K. J. Eberhart, «Particle swarm optimization,» *Proceedings of the IEEE international conference on neural networks*, pp. Vol. 4, pp. 1942-1948, 1995.
15. N. Hoertel, M. Blachier, C. Blanco, M. Olfson, M. Massetti, M. S. Rico, F. Limosin et H. Leleu, «A stochastic agent-based model of the SARS-CoV-2 epidemic in France,» *Nature Medicine*, 14 juillet 2020.
16. L. Perez et S. Dragicevic, «An agent-based approach for modeling dynamics of contagious disease spread,» *International Journal of Health Geographics*, 2009.
17. V. Srinivasan, L. Bryan, C. Jiangzhuo, H. Dave, V. Anil et M. Madhav, «Using data-driven agent-based models for forecasting emerging infectious diseases,» *Epidemics*, 2018.
18. «Modeling and Control of COVID-19 Propagation for Assessing and Optimizing Intervention Policies,,» [En ligne]. Available: https://ee.princeton.edu/news/ee-faculty-members-receive-grants-covid-19-research-c3ai-digital-transformation-institute.

Empowering Mission Preparation with AI*

Juliette MATTIOLI[1], Marc FIAMMANTE[2]

[1] Thales, France
[2] IBM, France
juliette.mattioli@thalesgroup.com - marc.fiammante@fr.ibm.com

Abstract. Today, mission management becomes complex due to uncertainty and dynamicity of asymmetric warfare. This paper underlines how symbolic AI (such as semantic information fusion, multi-criteria decision, AI planning, scenario planning, knowledge graph...) and data-driven AI (genetic algorithms, machine learning, neural networks...) could improve mission preparation life cycle, in particular mission intelligence, risk assessment, targeting and mission planning.

Keywords: Mission Preparation · Mission Intelligence · Risk Assessment · Mission Planning · Data-driven AI · Symbolic AI

1 Mission management

Modern military missions are becoming more and more complex. Not only are insurgents hard to identify, inscrutable and resourceful, but missions take place in very diverse, dynamic and uncertain operational environments, which are nothing like the traditional battlefield. Thus, in the mission process (see fig. 1.), the environment (physical and human terrain), all relevant actors and factors, as well as own means and assets, are assessed in order to prepare the mission for executing operations to achieve the intended effects. Then, mission preparation aims to identify the desired effects, based on the commander's intent and the available means to achieve these desired effects (which units will execute what activities). When formulating this plan, the commander with his staff needs to make assumptions and deductions, as not all information is always available on the environment or opponent. The plans are then translated into orders, which result in units executing mission operations.

Even though the strategic goals of asymmetric warfare remain constant, the increasingly destructive nature of hostile tactics requires that new approaches for dealing with asymmetric warfare be developed to neutralize potential threats and at the same time reduce collateral damage [1]. Therefore, mission management is highly complex involving **mission preparation**, **mission execution** and **mission debrief**. The most critical factors are uncertainty, dynamicity, and

* This article is the result of discussions held on the framework of the NATO working group: SG NIAG 252 on "Emerging Disruptive Technology in the Context of Emerging Powers".

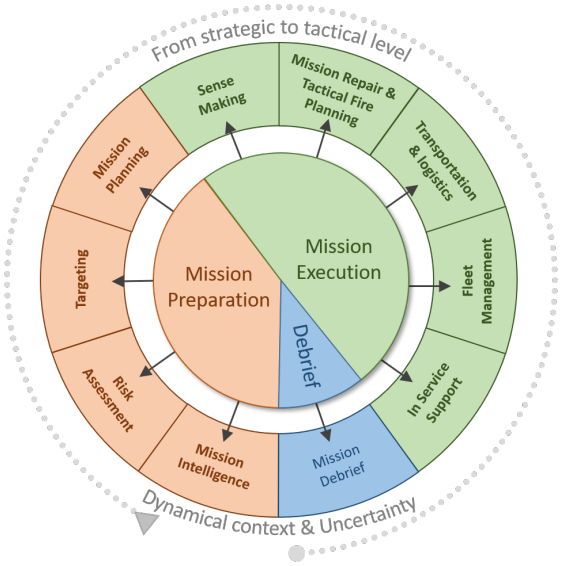


Fig. 1. Mission management involves preparation, execution and debriefing.

computational complexity. As mission management covers a huge area, this paper rather focuses on preparation by highlighting how some artificial intelligence (AI) techniques (data based AI and symbolic AI) could empower this capacity.

2 Data driven vs. symbolic AI

Introduced in 1956 [2], AI is a branch of computer science attempting to simulate human cognition capabilities such as perception, learning, abstraction, reasoning, planning, decision, dialogue and the ability to move and manipulate objects. AI includes a wide range of technologies which can be divided into two broad categories [3,4]: (1) **data driven AI** which includes machine learning such as neural networks and deep learning, statistical learning, evolutionary computing...; and (2) **symbolic AI** which focuses on ontology and semantics graphs, knowledge-based systems, reasonning... The aims of data-driven AI and symbolic AI are fundamentally different. The paradigm of data-driven AI is based on brain-style learning, whereas symbolic AI approaches employ model and knowledge reasoning. Thus, AI becomes critical for military mission management (see fig. 2) to extract value from data and knowledge by automating and optimizing processes, producing actionable insights and making a proactive decision under dynamical context and uncertainties by providing typical features such as learning, reasoning, and decision support.

3 Mission preparation

Mission preparation focuses on goal determination to support team decision making and actions (e.g. battlefield operations) over social, economic, and po-

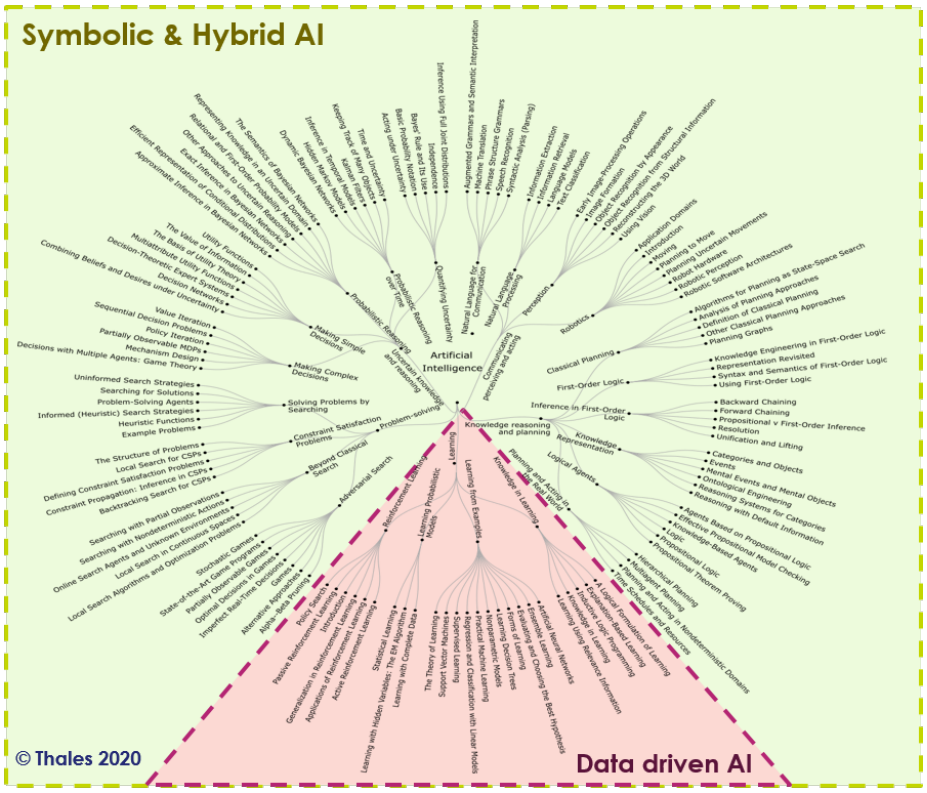


Fig. 2. An AI taxonomy attempt

litical constraints covering mission intelligence, risk assessment, targeting, and mission planning inclusing scenario planning and ressource allocation.

3.1 Mission Intelligence

Traditionally, military intelligence focused on threat and enemy, and the physical terrain of the area of operation. However with operations becoming increasingly complex (with more actors and non-combatant parties active on the 'battlefield'), and a focus shift to population centric operations the Human Terrain[3] [5] becomes increasingly important for achieving mission goals Human Terrain includes topics such as various population groups, their culture, local customs and traditions, political systems, tribal structures, and economic development. A synthetic view of all information domains for Mission Intelligence is represented by the Intelligence Pentagram.

Mission intelligence aims to build a cognitive situational awareness picture of the operational environment based on intelligence and counterintelligence

[3] Human terrain refers to the socio-cultural dynamics of an area including demographics as well as the cultural intricacies. The goal of better understanding local cultures and social structures is to improve the units' operational effectiveness.

information. To reach that goal, semantic information fusion combined sensor data with other data sources such as social networks. [6] proposed an approach and a framework for high-level information fusion based on conceptual graph, which was extended in order to manage uncertain information [7]. But before that fusion can occur in order to create knowledge an essential aspect is the analysis of raw data and semantic interpretation. That information can take many structured or unstructured formats as diverse as PDF, Word, PowerPoint, Excel, PNG, TIFF, JPG, JSON, HTML, etc. As an example doctrine documents are often made available as pdfs, targets in word documents, operations reports as powerpoint, and any variants of these. For that purpose, we initially created a machine learning platform to ingest documents at scale [8], platform that is now integrated in "Watson Discovery". For further details see Smart Document Understanding `https://cloud.ibm.com/docs/discovery?topic=discovery-sdu`.

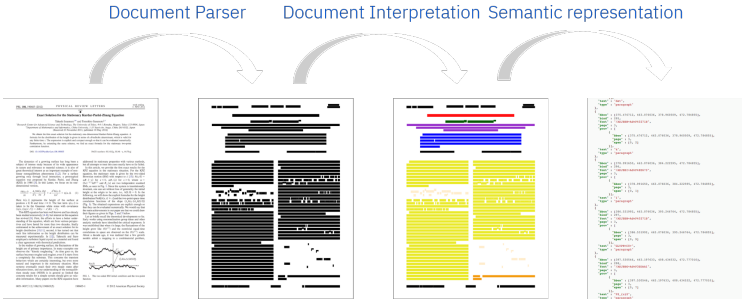


Fig. 3. From raw documents to semantic

Once the elements are extracted domain specific analysis is applied, which for Mission Intelligence can use information models such as JC3IEDM or NATO STANAG 5525 or any appropriate sets of dictionaries, taxonomies or ontologies. Semantic entities and relationships relevant to the domain creating a knowledge graph, which can contain millions of nodes such as the one created for COVID19. This knowledge graph enables natural language queries on the ingested information, questions such as: "What is the closest airport from that target location".

3.2 Risk evaluation and management

Risk management influence strategic and tactical military planning by identifying, assessing, and controlling risks/threats arising from operational factors that balance risk costs with mission benefits.

We propose to use Multi-Criteria Decision Making (MCDM) aggregation of the risk values for the hazards associated with each mission component, to form a partial ordering of those tasks that contribute most to overall risk. Efficient tools and methodologies are today available to establish a good MCDM model. By using the Thales Myriad© suite [9], the commander is able to built relevant metrics based on utility functions (capturing the risk/mission efficiency value) and to define coefficients according to his operational strategy.

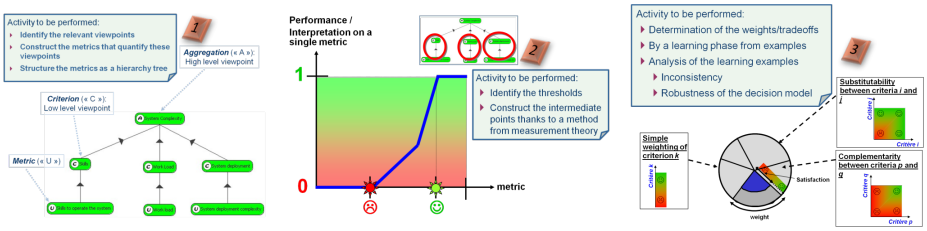


Fig. 4. Multi–criteria risk assessment of a mission based on MCDM Methodology

3.3 Targeting

Targeting is also a MCDM process which involves selection and prioritization of targets and the design of the appropriate response to them considering operational requirements and capabilities but also to Measure of Effectiveness, Measure of Performance and Measure of Suitability with respect to the specific desired effects. First, a target classification analyses sensor data and classifies observations to real-world objects (e.g., aircraft, armored vehicle...) through Machine learning, Bayesian classifier, Rule-Based Expert Systems, DempsterShafer Theory, or Fuzzy Logic [10]. Then, target aggregation identifies groups of related objects which can be represented as units comprising a number of subcategories (e.g., platoons, companies, and squadrons) or a specific situation (e.g., ambush and retreat). Therefore MCDM could be used for spatiotemporal targeting decision making in a dynamic operational environment. Commanders are not only interested in the evaluations, but also in the explanations of these scores [11]. So, above Myriad, e-Myriad is an application which provides insights into a particular evaluation outcome, to explain, in an intelligible way, the estimated evaluation score and on which attribute worked to increase the mission efficiency, even if the mechanisms of aggregation are based on notions or concepts which escape the understanding of a non-mathematician. This part is in line with the concept of "Explainable AI" (a.k.a. XAI) in which AI and how it comes to its decisions are made intelligible to the user by providing explanations [12].

3.4 Scenario Planning

To perform the threat/risk analysis, one identifies the vulnerabilities of each item listed during the inventory step and the potential attack means that may exploit them, and list all the threats that may plague the system. For each pair (vulnerability, threat), scenario planning identifies the consequences of a hypothetical exploitation of the vulnerability by the threat. AI Preemptive Scenario Planning can provide a significant "look-ahead" advantage with Automated Courses of Action where the Gray Zone is the new normal where there is a need to move from a posture of rapid response to a preemptive capability. The Scenario Planning Advisor (SPA) is a technology created by IBM Research that automatically projects many plausible high-impact future scenarios, to provide insights for strategic decision making[4]. It does not rely on traditional statistical

[4] `https://researcher.watson.ibm.com/researcher/view_group.php?id=9444`

forecasting techniques but uses AI Planning to generate scenarios, and uses Natural Language Understanding to extract knowledge from documents to craft the planning models needed to generate the scenarios together with current state awareness, to provide hints about where to focus what-if scenario analyses.

Fig. 5. IBM Scenario Planning Advisor

Using such AI techniques Strategic Planning becomes responsive to a rapidly evolving world allowing to focus on the new/unexpected, while reducing minimizing biases that are often encountered with small groups of humans generate scenarios. Additionally, it is fully transparent as SPA's causal models are accessible to anyone who cares to examine them.

3.5 Mission Planning

Military mission planning involves resource allocation and task scheduling which are combinatorial optimization problems [13]. Traditionally, such problems are solved to get a static solution (plan) with minimal makespan. Constraint-based Temporal Planning and Scheduling [14] is based on an explicit notion of time and a deep commitment to a constraint based formulation of planning problems.

However, there are lots of uncertainties in military operations, such as disruption of actions or unexpected increases of task duration, a static plan is prone to be invalid. The Divide-and-Evolve paradigm proposed by [15] can handle these issues based on the basic principle to execute a Divide-and-Conquer strategy driven by an evolutionary algorithm.

Mission planning issues can be mathematically modelled and solved with powerful algorithms from optimizers such as IBM CPLEX, which can produce precise and logical decisions[5]. The mathematical programming technology of CPLEX Optimizer enables decision optimization for improving efficiency, reducing costs and increasing usage of assets. It provides flexible, distributed, high-performance mathematical programming solvers addressing difficult problems for linear programming and constrained programming. problems with a simple Optimization Programming Language (OPL) to express rules and constraints. As an example it is used in the Mission Planning context to:

– Minimize the risk to mission, by reducing unmet deployment requirement assignments due to hard business rules.

[5] `https://www.ibm.com/analytics/cplex-optimizer`

– Minimize the risk to force, by maintaining troops at the desired dwell ratio, which is defined as the duration of rest, "dwell" at the time of a new deployment to the duration of the prior deployment.
– Minimize the risk to plan by reducing deployment changes made to the existing deployment plan, and
– Minimize the risk to compatibility by reducing the violation of desired compatibility of the deployment decisions.
– Allocate vehicles and other across the horizon to match vehicle demand and arrival forecast.

4 Conclusion

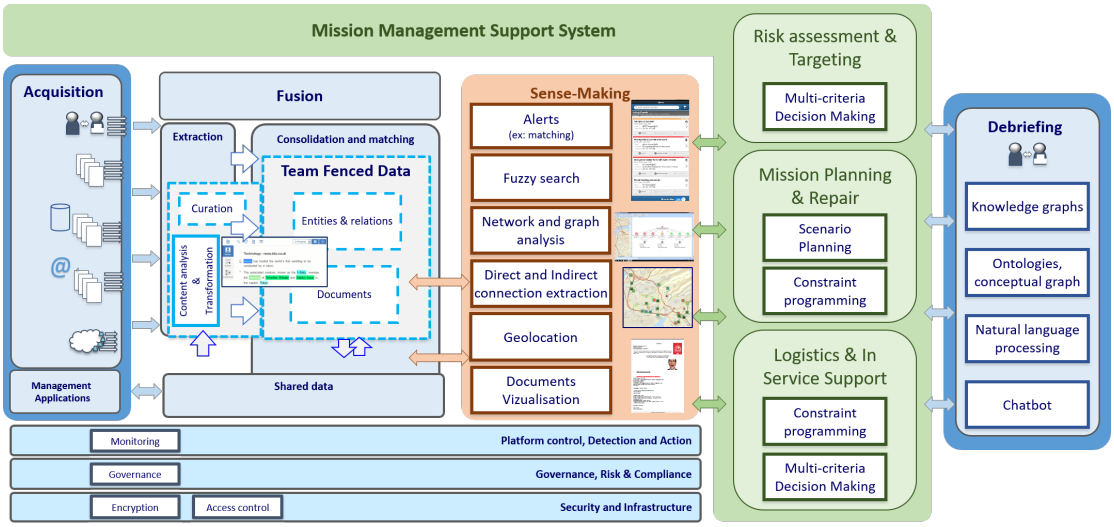


Fig. 6. An instance of a mission management support system

To manage efficiently an overall mission, one of the key issues is to build a cognitive situational awareness [16] based on mission intelligence. The understanding of the on-going situation gains from the total sum of relevant information provided to make a correct decision-making regarding the mission objectives and/or the desired state. But, the complexity of friendly and enemy organizations, unique combinations of terrain and weather, and the dynamic interaction among all participants create uncertainty. Thus, mission execution aims to make timely and effective decisions and to act faster than the enemy to place it under the pressures of uncertainty and time. These decisions include assigning tasks; prioritizing, allocating, and organizing forces and resources; and selecting the critical times and places to act. The speed accuracy of a decisionmaker's actions to address a changing situation is a key contributor to agility. Finally, mission debriefing occurs either after (post-event) an operational mission or during (within-event) a training simulation. The value of performance debriefing is that it allows operational to discuss to identify gaps and propose strategies for

improvement driven by the overall satisfaction of the mission. This paper underlines that all these capabilities especially mission preparation can be enhanced by AI (see Fig.6).

References

1. M. Duistermaat, NTA. Schmitz, LRMA. Jacobs, JP. Faye, et al. SIMS-smart information for mission success: Enhancing joint mission planning and training. In *Presentation at ITEC 2011 International Training & Education Conference*, 2011.
2. S. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.
3. J. Mattioli and Ch. Meyer. L'intelligence artificielle au service des systèmes critiques. *REE*, 4:112–122, 2018.
4. J. Mattioli, P. Perico, and PO. Robic. Artificial intelligence based asset management. In *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, pages 151–156. IEEE, 2020.
5. M. Arnold. Improving the coalition's understanding of 'the people'in afghanistan: Human terrain mapping in kapisa province. *Small Wars Journal*, pages 1–9, 2010.
6. C. Laudy. Semantic knowledge representations for soft data fusion. *Efficient Decision Support Systems-Practice and Challenges from Current to Future*, 2011.
7. S. Fossier, C.e Laudy, and F. Pichon. Managing uncertainty in conceptual graph-based soft information fusion. In *Proceedings of the 16th International Conference on Information Fusion*, pages 930–937. IEEE, 2013.
8. P. Staar, M. Dolfi, Ch. Auer, and C. Bekas. Corpus conversion service: A machine learning platform to ingest documents at scale. *Association for Computing Machinery*, July 2018:774–782, 2018.
9. Ch. Labreuche and F. Le Huédé. Myriad: a tool suite for mcda. In *EUSFLAT Conf.*, pages 204–209, 2005.
10. J. Ma. Constrained target clustering for military targeting process. *Defence Science Journal*, 67(5), 2017.
11. M. Grabisch, Ch. Labreuche, and M. Ridaoui. On importance indices in multicriteria decision making. *European Journal of Operational Research*, 277(1), 2019.
12. Ch. Labreuche and S. Fossier. Explaining multi-criteria decision aiding models with an extended shapley value. In *IJCAI*, pages 331–339, 2018.
13. L. Tang, C. Zhu, W. Zhang, and Z. Liu. Robust mission planning based on nested genetic algorithm. In *The Fourth International Workshop on Advanced Computational Intelligence*, pages 45–49. IEEE, 2011.
14. Ph. Laborie, J. Rogerie, P. Shaw, and P. Vilím. Ibm ilog cp optimizer for scheduling. *Constraints*, 23(2):210–250, 2018.
15. M. Schoenauer, P. Savéant, and V. Vidal. Divide-and-evolve: a new memetic scheme for domain-independent temporal planning. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 247–260, 2006.
16. C. Laudy, J. Mattioli, and N. Museux. Cognitive situation awareness for information superiority. In *IST Panel on Information Fusion for Command and Control*, pages 1–8. NATO, 2005.

Quantification de l'incertitude pour l'apprentissage automatique
revue de quelques méthodes

Marc Lambert

DGA/CATOD
marc-h.lambert@intradef.gouv.fr

Résumé La quantification de l'incertitude consiste à estimer l'erreur de prédiction d'un classifieur ou régresseur. En régression, on peut ainsi chercher à fournir un intervalle de confiance permettant d'encadrer l'erreur de prédiction. En classification, il s'agira d'estimer la probabilité d'appartenance à chaque classe. Une grande partie des algorithmes d'apprentissage automatique ne fournissent pas de telles erreurs ou fournissent des estimation d'erreurs erronées. Par exemple, les réseaux de neurones sont connus pour être trop confiants en leurs prédictions, typiquement la sortie de la fonction softmax d'un classifieur a tendance a saturer à 0 ou 1. La quantification de l'incertitude consiste à corriger ces erreurs ou à modifier ces algorithmes de sorte qu'ils fournissent des estimations consistantes avec les erreurs empiriques. Cet article présente quelques solutions actuelles, tel que le calibrage et les méthodes Bayésiennes et souligne les enjeux de la quantification d'incertitude pour fournir des IA robustes et fiables.

Keywords: Incertitude · Calibrage · Réseau Bayésien · Dropout.

1 Pourquoi quantifier l'incertitude ?

Un système autonome n'est jamais isolé et fait partie d'une architecture d'ensemble. Par exemple, une caméra infrarouge de reconnaissance de cibles sur drone fournira des sorties qui seront intégrées à une situation tactique, enrichie par des éléments de navigation, et transmise à un opérateur déporté via une liaison de données ; un radar de reconnaissance automatique pour voiture autonome sera intégré à un système superviseur qui pourra croiser les informations avec les informations de cartographies et les informations issues des autres capteurs comme les données laser et images. Ainsi, la quantification d'incertitude permet à un système autonome de fournir une information enrichie qui va faciliter son intégration dans l'architecture d'ensemble, voire augmenter les performances globales de l'architecture. Le système haut niveau pourra ainsi soit présenter des niveaux de confiance à l'opérateur, soit sélectionner la brique fournissant la prédiction la plus certaine ou même fusionner les prédictions de différentes briques pour réduire les incertitudes des briques élémentaires. Une revue des concepts possibles de fusion est décrite en [6].

2 Les quantifications possibles de l'incertitude

Les méthodes d'apprentissage sont basées historiquement sur le formalisme des probabilités. Par exemple la fonction de réponse d'un neurone en sigmoïde, σ, est interprétée comme la probabilité d'une loi binomiale. Les approches classiques d'entrainement par maximum de vraisemblance donnent des estimations de probabilité souvent trop confiantes (saturation à 1 ou 0) qui ne peuvent pas être exploitées telles quelles pour estimer l'erreur de prédiction. Pour palier à ce problème on distingue les approches suivantes :

- **L'approche fréquentiste :** l'approche fréquentiste considère les sorties d'un système comme la réalisation d'un tirage aléatoire suivant une loi de distribution des entrées. Si on peut estimer cette distribution, alors on peut effectuer des tirages pour calculer les statistiques des prédictions (approche Monte Carlo). Les courbes ROC (Receiver Operating characteristic) sont ainsi classiquement utilisées pour comparer différents classifieurs suivant les taux de vrais et de faux positifs. Malheureusement on ne connait que très partiellement la distribution des entrées et on doit souvent se limiter à une base de tests qui peut être fortement biaisée.
- **L'approche Bayésienne :** Dans l'approche Bayésienne, on considère les paramètres w à estimer comme des variables aléatoires, les données d'entrée x étant supposées fixes (pour les modèles discriminatifs). La distribution de ces paramètres est apprise puis utilisée pour prédire les sorties. Contrairement à l'approche fréquentiste, l'approche Bayésienne fournit ainsi une distribution complète des sorties qui permet une estimation de la variance de la prédiction. Elle permet aussi d'introduire plus naturellement des informations a priori sur le modèle.
- **L'approche générative :** Dans les approches génératives, on considère les entrées x comme des variables aléatoires. La loi de x est apprise puis la règle de Bayes est utilisée pour en déduire la loi des prédictions. Comme dans l'approche Bayésienne, on dispose de la loi complète $p(y|x)$ mais l'approche générative a des qualités supplémentaires : en calculant $p(x)$ on peut détecter des données aberrantes et en utilisant $p(y)$ on peut gérer les données mal équilibrées [18]. L'approche générative et l'approche Bayésienne peuvent être combinées, dans ce cas plutôt que de chercher à estimer la distribution des entrées on va plutôt estimer la distribution de variables latentes de plus faible dimension. Les auto-encodeurs variationels en sont un exemple [11].

Bien que cet article ne porte que sur les méthodes probabilistes, ils convient de rappeler que d'autres approches existent telle que l'approche ensembliste qui cherche à garantir que les sorties sont contenues dans un intervalle borné. Cette approche utilise des méthodes reposant sur l'évaluation de la constante de Lipschitz ou sur des algorithmes de preuves formelles, voir [2] pour une revue de ces méthodes.

3 Quantification d'incertitude en traitement statistique du signal

Nous commençons par considérer les méthodes de quantification qui ont déjà été éprouvées en traitements statistique du signal et qui sont embarquées sur de nombreux systèmes de défense.

3.1 Régression par estimation bayésienne

Les méthodes bayésiennes sont utilisées depuis les années 60 sur les systèmes autonomes à travers le filtre de Kalman. Le filtrage de Kalman permet de quantifier l'incertitude sur l'état θ du système. Un cas d'application typique est la poursuite de cibles par un radar où θ inclue la position et la vitesse de la cible, inconnues, que l'on cherche à estimer en fonction d'observations y composées de mesures angulaires et distance bruitées.

Le filtrage de Kalman linéaire est efficace : il fournit donc une matrice de covariance consistante des erreurs empiriques. Malheureusement dans le cas non linéaire, ce n'est plus vrai. Le filtre de Kalman étendu peut estimer une matrice de covariance erronée qui sous estime largement l'erreur empirique. Il existe des techniques pour rétablir la consistance de cet estimateur, par exemple le filtrage adaptatif permet d'adapter automatiquement le gain de Kalman en fonction des erreurs de prédiction observées [13]. Une approche par réseaux de neurones a récemment été proposée pour adapter dynamiquement les bruits d'observations pour une application du Kalman en navigation inertielle, voir figure 1.

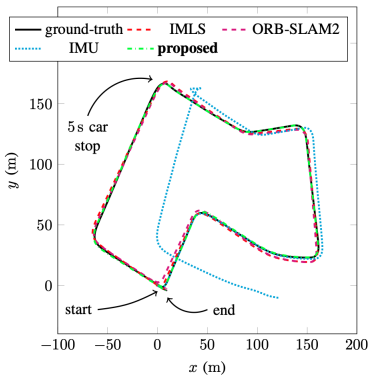


Figure 1. Filtre de Kalman adaptatif appliqué à la navigation avec centrale inertielle (IMU). En apprenant la covariance du bruit, l'IMU ne dérive plus et donne des résultats equivalents aux solutions utilisant laser ou recalage d'image, extrait de [4].

3.2 Classification par test statistique

Pour les problèmes discrets de décision, une application qui a été largement éprouvée est la détection de cibles où l'on doit décider entre deux "classes" : présence ou absence de cibles. Le test statistique de Neyman-Pearson est alors classiquement utilisé pour décider de l'hypothèse la plus vraisemblable en se basant sur une information à priori sur la distribution des signaux d'entrée y. Ce seuillage est utilisé dans la chaîne de réception (filtre adapté) de la plupart des radars, pour lesquels des modèles probabilistes de signatures peuvent être bien définis. Par exemple, le bruit thermique du récepteur peut être approché par une distribution Gaussienne f_{bruit} non structurée, alors que la cible va répondre avec une observation structurée, distribuée suivant une loi différente f_{cible} (de type χ^2 par exemple). Ces modèles probabilistes permettent de prédire les probabilités de détection et de fausses alarmes en fonction d'un seuil de détection. Pour réduire le taux de fausses alarmes, des tests sont opérés en cascade. On fixe alors un seuil de fausses alarmes très bas, quitte à ne pas détecter de cibles, et on fait N observations indépendantes pour cumuler la probabilité de détection élémentaire p_e par la formule :

$$p_d = 1 - (1 - p_e)^N. \tag{1}$$

On peut ainsi obtenir un très bon ratio de détection/fausse alarme.

3.3 Lien avec les méthodes d'apprentissage

Les exemples de prédicteurs que nous avons présentés appartiennent à la classe des méthodes génératives qui consistent à estimer d'abord la probabilité des données, ie les modèles de bruits capteurs, pour ensuite en déduire la probabilité de l'état ou de la sortie. Les paramètres des lois (biais, variances) doivent être estimés par un processus en amont de **calibrage du capteur**. La phase de calibrage nécessite des données "d'apprentissage", par exemple pour un radar on va effectuer des mesures sur des objets connus (sphère ou antenne de calibrage) dans des conditions maitrisées. Nous allons voir que le problème de calibrage se pose aussi dans le cadre des méthodes d'apprentissage.

4 Approches fréquentistes pour quantifier l'incertitude d'apprentissage

4.1 La base de test

Pour les estimateurs qui sont liés à une base d'apprentissage, la certification fonctionnelle passe par une base de données, appelée base de tests. On peut alors calculer les incertitudes d'un estimateur en calculant directement la statistique des erreurs de prédiction sur cette base : moyenne empirique et covariance empirique. Cette approche est cependant limitée car la certification n'est valide que pour des observations qui sont proches de celles de la base de test. Or un système autonome est amené a évoluer dans des environnements inconnus ou

changeants pour lesquels on ne peut plus garantir la performance. L'enjeu est alors de construire des bases d'apprentissage le moins biaisées possibles en explorant le mieux possible l'espaces des paramètres de la scène. Cela peut se faire par simulation des données, par augmentation des données réelles ou en utilisant des réseaux génératifs de type GAN appris sur les données réelles [19].

4.2 Le modèle fréquentiste

La certification par une base de tests permet d'évaluer l'incertitude a posteriori, une fois que l'algorithme a fait ses prédictions. Avec les modèles probabilistes, il est aussi possible d'évaluer l'incertitude de façon online, on parle d'auto-évaluation puisque c'est l'algorithme lui même qui prédit ses erreurs. En apprentissage supervisé probabiliste, les observations dépendent de données d'entrées X via un modèle de loi p supposé connu a priori mais dépendant de paramètres θ inconnus $p(Y|\theta, X)$. L'approche fréquentiste, dite classique, consiste a calculer les paramètres du modèle θ qui maximisent la vraisemblance des observations Y : $\theta^* = \arg\max p(Y|\theta, X)$. La prédiction d'une classe pour une donnée de test x est alors donnée par $p(y|x) = p(y|\theta^*, x)$. Le problème avec cette approche est que la probabilité résultante est souvent trop forte, le prédicteur surestime sa précision. Pour corriger ces erreurs de prédiction, il est possible de calibrer le modèle.

4.3 Le calibrage en apprentissage

Une approche simple de calibrage assure une mise en cohérence des statistiques globales. Par exemple si un classifieur fait 100 prédictions avec une probabilité de 0.8 pour chacune, on s'attend à ce que 80 de ces prédictions soient effectivement correctes. Cette exigence peut se traduire par la métrique "Expected Calibration Error" qui compare la précision empirique à la distribution prédite en échantillonnant les probabilités en M intervalles :

$$ECE = \sum_{i=1}^{M} \frac{n_b}{N} |acc(b_i) - conf(b_i)|, \tag{2}$$

où $acc(b_i)$ est le pourcentage de données bien classifiées pour les données dont la probabilité estimée est dans l'intervalle b_i et $conf(b_i)$ est la probabilité estimée moyenne sur cet intervalle. Les méthodes de re-calibrage consistent à ajouter une fonction correctrice en sortie de l'estimateur dont les paramètres peuvent être appris pour minimiser la métrique ECE. Le "temperature scaling" fait partie des méthodes récemment proposées [9]. Ces méthodes ne tiennent pas compte de la proximité des entrées et ne garantissent pas la robustesse du classifieur : deux entrées proches peuvent donner des prédictions différentes.

5 Approches bayésiennes pour quantifier l'incertitude d'apprentissage

Les méthodes fréquentistes par maximum de vraissemblance fournissent des estimations ponctuelle θ^* qui ne contiennent qu'une information parcellaire sur les données. Par conséquent les probabilités de prédiction qui en résultent sont souvent trop confiantes et non consistante des erreurs empiriques. Les méthodes bayésiennes vont conserver une information plus riche qui va rétablir, en partie, la consistance des probabilités en sortie.

5.1 Marginalisation de la vraissemblance : calcul de l'évidence

Dans les méthodes bayésiennes, les paramètres θ du modèle sont vus comme des variables aléatoires et sont inférés en fonction d'un modèle aléatoire sur les observations Y. L'inférence repose sur la formule de Bayes :

$$p(\theta|Y,X) = \frac{p(Y|\theta,X)p(\theta)}{p(Y|X)}, \tag{3}$$

où $p(\theta)$ représente notre hypothèse à priori sur θ et $p(Y|X) = \int p(Y|\theta,X)p(\theta)d\theta$ est appelé la fonction de partition.

Une fois que la distribution $p(\theta|Y,X)$ a été estimée, la prédiction d'une classe pour une donnée de test x peut se faire en intégrant sur cette distribution pour obtenir la loi marginale appelée **évidence** :

$$p(y|x) = \int p(y|\theta,x)p(\theta|Y,X)d\theta, \tag{4}$$

C'est cette évidence qui nous donne la quantité recherchée, à savoir l'incertitude de prédiction. Cette incertitude se réduit lorsque l'entrée évaluée x est proche d'un élément de la base d'apprentissage comme montré figure 2 dans le cas Gaussien. Il est possible de calculer $p(y/x)$ sans passer par l'estimation du paramètre θ mais directement depuis les données. Cet apprentissage non paramétrique s'appel processus Gaussien et est lié aux méthodes d'interpolation à noyau. Pour le lien entre évidence et processus Gaussien voir [12].

Pour évaluer l'influence de l'approche bayésienne sur la quantification d'incertitude, on peut comparer les sorties d'un prédicteur classique, par exemple la fonction softmax d'un réseau, avec les sorties d'un prédicteur bayésien calculant l'évidence. Un prédicteur bayésien a tendance a moins surestimer ses prédictions comme montré figure 3.

Les intégrales qui apparaissent dans la phase d'apprentissage et la phase de prédiction sont souvent difficiles à calculer lorsque $p(y/\theta)$ est une fonction complexe, typiquement la sortie d'un réseau de neurone profond. Les méthodes d'inférence bayésienne se différencient par leur manière d'approcher ces intégrales. On distingue deux grandes familles que nous présentons plus en détail : les méthodes Monte Carlo et l'inférence variationnelle.

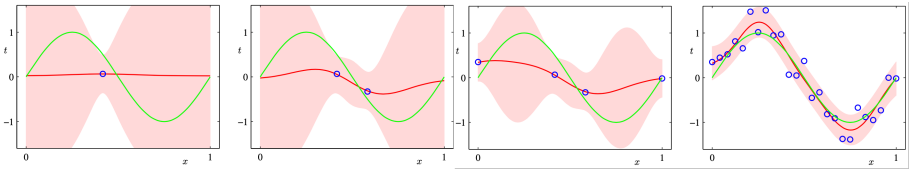


Figure 2. Evolution de l'évidence en fonction du nombre de données. Les données, représentés par des ronds bleus, sont générées suivant une sinusoïde $t = sin(2\pi x)$ avec du bruit et un modèle de régression linéaire est employé. Pour pouvoir approcher la fonction sinusoïde, la régression linéaire n'est pas appliquée directement sur les entrées mais sur des attributs construits avec 9 fonctions de bases Gaussiennes. On observe que l'écart type de la distribution $p(y|x)$, représenté par la région rose, se réduit autour des données d'apprentissage. Figure extraite de [3].

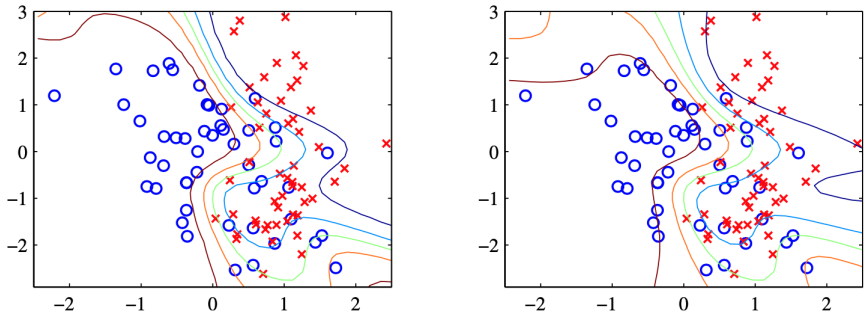


Figure 3. Courbes d'iso-probabilités pour un problème de classification binaire avec un réseau de neurone à huit couches utilisant une sortie classique obtenue par maximum de vraisemblance (figure de gauche) et une sortie bayésienne calculé avec l'évidence (figure de droite). La courbe verte correspond a une probabilité de 0.5, les autres courbes aux probabilités 0.1, 0.3, 0.7 et 0.9. L'estimation bayésienne fournit une estimation de confiance plus robuste que l'estimation classique. Figure extraite de [3].

5.2 Les méthodes Monte Carlo

Les méthodes Monte Carlo utilisent une approximation intégrale en somme discrète sur un nombre fini d'échantillons tirés sur la loi à intégrer, ces méthodes peuvent converger lentement lorsque la distribution à estimer est principalement concentrée sur un support de faible taille. Les techniques d"importance sampling [1] peuvent éviter ces écueils. En introduisant un modèle dynamique bien choisi il est aussi possible de mieux s'adapter à ce genre de distribution : c'est le principe des méthodes à chaîne de Markov (MCMC) dont sont dérivés les algorithmes de Metropolis-Hasting [1]. Les méthodes Hamiltoniennes en sont les versions les plus sophistiqués et font partie de l'état de l'art [15]. Des versions stochastiques ont été développées pour pouvoir traiter des données en grand nombre (Big Data) telles que le gradient stochastiques de Langevin SGLD [20], cependant lorsque

la dimension des données est grande, leurs bonne convergence peut nécessiter de nombreuses passes sur la base de données.

La technique de Monte Carlo Dropout [7] permettrait d'estimer la moyenne et la covariance de la distribution postérieure de façon efficace en grande dimension. Le Dropout est utilisé normalement lors de la phase d'apprentissage, mais si on effectue N tirages de Dropout lors de la phase de prédiction, il a été constaté que l'on pouvait calculer les moments empiriques avec de bonnes précisions pour des valeurs de N pas trop grandes. Le Monte Carlo Dropout fournit cependant des estimations non consistantes des erreurs empiriques et nécessite une étape de calibrage [8].

5.3 L'inférence variationnelle

Plutôt que d'optimiser une fonction de coût basée sur la log-vraissemblance $\log p(y|\theta)$ comme c'est fait classiquement en apprentissage, en inférence variationelle on optimise une fonction de coût basé sur la divergence de Kullback-Leibler entre le vrai posterior $p(\theta|y)$ que l'on ne sait pas calculer et un posterior approximé plus simple à intégrer $q_\lambda(\theta)$ qui est entièrement définit par un jeu de paramètres λ :

$$\mathbf{KL}(q_\lambda(\theta)||p(\theta|y)) = \int q_\lambda(\theta) \ln \frac{q_\lambda(\theta)}{p(\theta|y)} d\theta, \tag{5}$$

On peut alors calculer la dérivée de cette métrique par rapport aux paramètres λ et chercher les paramètres optimaux via une descente de gradient afin de déterminer q. Pour peu que les distributions soient assez régulières, l'intérêt de cette approche est que l'on peut dériver sous l'intégrale et aboutir à des équations qui ne dépendent plus de la fonction de partition difficile à calculer. Ces méthodes nécessitent cependant de définir un modèle pour la distribution q qui soit adapté à la distribution vrai, l'erreur d'approximation étant donnée par la métrique de Kullback-Leibler. Le modèle pour la distribution approximé peut éventuellement reposer sur une fonction de changement de variable dont les paramètres sont appris sur les données.

L'inférence variationelle a été utilisée récemment pour l'apprentissage profond en considérant pour q une distribution Gaussienne avec une matrice de covariance factorisé pour limiter le nombre de paramètres à calculer [16], [14]. Une distribution Gaussienne permet de calculer facilement la prédiction du réseau. L'inférence variationelle a été utilisée aussi dans le cas des approches génératives pour estimer la distribution de variables latentes sous jacente aux données via des auto-encodeur bayésien appelés "Variational Auto Encoder" ou VAE [11], [5].

Les VAE utilisent aussi une distribution Gaussienne pour q mais ont été étendus à des distributions multimodales construites à l'aide de fonctions de flots, appelées "normalizing flow" [17]. Ces techniques font l'objet de recherches actives et d'un workshop dédié aux conférences "International Machine Learning Society" (ICML) de 2019 et 2020.

6 Conclusion

La quantification de l'incertitude est une discipline qui a été utilisée très tôt dans les systèmes automatiques tels que les radars pour le pistage ou la détection de cibles. C'est un domaine actuellement en plein essor pour l'apprentissage profond. Les approches bayésiennes offrent un cadre rigoureux qui pourrait permettre de compléter les validations statistiques sur base de tests. L'enjeu de ces approches est de pouvoir calculer la distribution des prédictions de réseaux de neurones qui sont des fonctions non linéaires difficiles à intégrer et ceci en un temps raisonnable. Toutefois, ces méthodes ne sont pas exemptes de biais et doivent être associées à des méthodes de calibrage pour fournir des incertitudes consistantes des erreurs empiriques et rendre possible la sélection ou fusion d'information au niveau supérieur.

Références

1. Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1) :5–43, 2003.
2. Hugo Bazille, Eric Fabre, and Blaise Genest. Certification des réseaux neuronaux profonds : un état de l'art en 2019.
3. Christopher Bishop. *Pattern Recognition and Machine Learning*, volume 16, pages 140–155. 01 2006.
4. M. Brossard, A. Barrau, and S. Bonnabel. Ai-imu dead-reckoning. *IEEE Transactions on Intelligent Vehicles*, pages 1–1, 2020.
5. Carl Doersch. Tutorial on variational autoencoders, 2016.
6. Bloch et al. Fusion : General concepts and characteristics. *International Journal of Intelligent Systems*, 16 :1107 – 1134, 10 2001.
7. Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation : Representing model uncertainty in deep learning, 2015.
8. Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout, 2017.
9. Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017.
10. Danijar Hafner. Building variational auto-encoders in tensorflow. Blog post, 2018.
11. Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.
12. D. MacKay. Introduction to gaussian processes. 1998.
13. R. Mehra. On the identification of variances and adaptive kalman filtering. *IEEE Transactions on Automatic Control*, 15(2) :175–184, April 1970.
14. Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. Slang : Fast structured covariance approximations for bayesian deep learning with natural gradient, 2018.
15. Radford Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 06 2012.
16. Victor M.-H. Ong, David J. Nott, and Michael S. Smith. Gaussian variational approximation with a factor covariance structure. *Journal of Computational and Graphical Statistics*, 27(3) :465–478, 2018.

17. Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv :1505.05770*, 2015.
18. Jebara S. Discriminative, generative, and imitative learning. 08 2005.
19. Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1) :60, 2019.
20. Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 681–688, Madison, WI, USA, 2011. Omnipress.

Evaluating the stability of the Neur-HCI framework

Roman Bresson[1,2], Johanne Cohen[2], Eyke Hüllermeier[3], Christophe Labreuche[1], and Michèle Sebag[2]

[1] Thales Research and Technology, 91767 Palaiseau, France
{roman.bresson, christophe.labreuche}@thalesgroup.com
[2] LRI, CNRS - INRIA, Université Paris-Saclay, 91400 Orsay, France
{johanne.cohen, michele.sebag}@lri.fr
[3] Department of Computer Science, Paderborn University, 33098 Paderborn, Germany eyke@upb.de

Abstract. Artificial Neural Networks (ANNs) allow to exploit the information contained in data in order to build highly-performant predictive models, at the cost of interpretability and transparency. In this work, we describe the Neur-HCI framework, a class of ANNs which learns highly constrained and naturally interpretable models called Hierarchical Choquet Integrals along with monotonic rescaling of the features. We study empirically the stability of the learned model, and its robustness to small changes in the training set, a property necessary for building models which are to-be-used in safety-critical settings, such as defense applications.

Keywords: Artificial Neural Networks · Robustness · Multicriteria Decision Aiding

1 Introduction

Artificial neural networks (ANNs) are machine learning models whose parameters are learned from available data through the use of the algorithm called *gradient backpropagation*. The thus learned function is usually highly overparameterized, meaning that there are many more degrees of freedom than necessary. This allows ANNs to learn any function, given a sufficient size, but comes at the cost of several drawbacks.

First of all, such models have a high risk of *overfitting*, a phenomenon where the model is able to fit the training data so well that it also learns random noise, thus limiting its ability to generalize beyond the training dataset; this is particularly true with small datasets, and is usually adressed through methods such as regularizations [7] and dropout [8]. Secondly, the learned model will be very hard to directly interpret and characterize. Indeed, as all of the parameters are simple weights in a very large model, one weight doesn't hold any understandable meaning, and one must look at how groups of parameters work together in specific situations in order to try and interpret a given result, making a global

understanding of the model intractable in practice. Finally, as the network is allowed to adopt almost any behaviour, it is hard to ensure that some formal guarantees are enforced everywhere on the input space.

ANNs are, thus, often used as black box models; even though some methods are being developed in order to analyze and interpret the outputs of some models [2], they are highly probabilistic, and offer no formal explanation or guarantee. These are not issues in many application cases of ANNs, such as image recognition or online translation services, where mistakes hold no real consequences, especially as these systems perform very well on average. Nonetheless, in *safety-critical* settings, a user might wish for the model to be fully interpretable, meaning that an expert can immediately, for a given prediction, understand the output simply from the parameters or other relevant deterministic indicators. Finally, they might want some formal constraints, such as the monotonicity of the output w.r.t. the input, or the type of certain interactions, to be strictly enforced; such constraints usually come from domain knowledge, and their non-violation is necessary for the model to be trustable.

The idea is then to restrict the class of model in order to make sure that the domain knowledge is fully satisfied. We consider here monotonicity as it is an important property in many applications in Defense. Models coming from Multi-Criteria Decision Aiding (MCDA) – aiming at representing the preference of some decision maker regarding how to select or rank alternative on the bases of several conflicting *criteria* – are then good candidates. We focus on a general class of MCDA models based on the Hierarchical Choquet integral (HCI) as (1) it is a versatile model able to capture a wide range of decision strategies and in particular interacting variables, (2) each level is interpretable and can be explained to a user [6], (3) it is described in a hierarchical way through simple concepts understandable by a user.

Traditional approaches to construct the HCI require lengthy interactions with an expert user where they need to provide local information on subparts of the model. Methods to learn such models from data have been developed, allowing to limit the need for costly expert time [3] but still cannot learn the full model at once and cannot handle a hierarchy. Leveraging the power of ANNs, the Neur-HCI framework has been recently developed in order to learn all parameters of the general 2-additive HCI [1]. Another advantage of this approach is that the ANN is only used as a tool for learning; once the model is trained, it can easily be re-written as its closed-form mathematical formula, which is lightweight to both store and compute w.r.t. a complex ANN, allowing it to be run in embedded or resources-scarce environments.

In this paper, we further develop the Neur-HCI framework by providing elements towards its qualification. Notably, the small size and Lipschitz properties of the model allow it to naturally be safe from adversarial attacks [4]. We describe the HCI model in Section 2. The Neur-HCI framework, which allows to learn such models, is presented in Section 3. We then introduce in Section 4 the *identifiability* of the model – showing that there are no two HCI models which have the same output. This property has important theoretical and practical

consequences. The model parameters have a true meaning as they are uniquely specified. This allows to explain the model [6], which is important in Defense applications. Identifiability is a theoretical result as it considers the limit case where there is a continuum of training examples. Then, we experimentally show in Section 5 the stability of the learned model w.r.t. the training dataset, both on artificial and real-life data. Changing a little bit the training set only marginally affects the values of the parameters of the HCI model. This is important for the user to trust the learned model and its ability to generalize to new examples.

2 Description of the HCI model

2.1 Scoring Models and Monotonicity condition

We consider a set $N = \{1, ..., n\}$ of criteria, where the domain of definition of criterion i is the set X_i. Alternatives are characterized by a value on each criterion and are thus elements of $X = X_1 \times \cdots \times X_n$. We consider a general setting in which alternatives are evaluated by a scoring function $\mathcal{S} : X \to \mathbb{R}$.

Example 1. A surveillance aircraft must choose an itinerary in order to take pictures of some areas of interest. The proposed itineraries, or alternatives, are rated on 6 criteria: (1) duration of the mission (in seconds), (2) distance between the trajectory and the areas of interest (in km) [efficiency of the mission], (3) probability of detection by enemy radars, (4) probability of engagement by enemy forces, (5) distance to allied forces able to assist, (6) fuel consumption (in liters). $\mathcal{S}(a)$ represents the level of satisfaction associated to itinerary a. ■

Monotonicity plays a central role in many safety-critical problems. By monotonicity, we mean that there exists a natural preference denoted by $\succeq_i$ on each criterion i, such that $x_i \succsim_i y_i$ ($x_i, y_i \in X_i$) indicates that x_i is at least as *good* as y_i, and scoring function $\mathcal{S}$ is monotonic in $\succsim_1, \ldots, \succsim_n$:

$$\forall x, y \in X \qquad [\forall i \in N \;\; x_i \succsim_i y_i] \Rightarrow \mathcal{S}(x) \geq \mathcal{S}(y). \tag{1}$$

Example 2 (Ex. 1 cont'd). All criteria shall be minimized, i.e. $\succeq_i$ corresponds to $\leq$ for all i. For instance, the smaller the distance to areas of interest, the better. Relation (1) simply says that an itinerary which ends sooner, gets closer to the areas of interest, has smaller probability of detection and engagement by enemy, is closer to allies, and consumes less, is preferred. ■

2.2 Hierarchical decomposition of the model

In order to ease the interpretation and explanation of the scoring function $\mathcal{S}$, the n criteria are organized in a tree, where the leaves are the criteria N, the root is the overall score, and the other nodes are intermediate scores aggregating a subset of criteria that make sense to the user.

Example 3 (Ex. 2 cont'd). Thanks to a domain expert, the six criteria are organized as in Fig. 1. There are two intermediate nodes: node 7 related to the risk of being detected/engaged by an enemy and node 8 related to the safety of the mission. Node 9 represents the overall score. ■

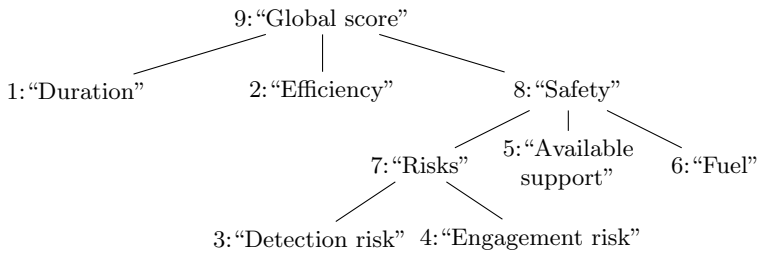


Fig. 1. Hierarchy of criteria for the itinerary problem

In this example, the intermediate nodes completely make sense to the user. The model allows for several levels of interpretation, for different users, as described in [6].

Thanks to the hierarchy, the scoring model $\mathcal{S}$ is decomposed in the tree. If the nodes are numbered $1, \ldots, m$ (with $m > n$), where the criteria are labeled $1, \ldots, n$ and the top node is m, then a score u_i is computed at each node given the score of its children, based on an aggregation function $F_i : \mathbb{R}^{c_i} \to \mathbb{R}$, where c_i is the number of children of node i. The overall score $\mathcal{S}$ is then the score of node m.

Example 4 (Ex. 3 cont'd). Functions $u_1, \ldots, u_6$ transform the values of metrics duration, distance, ... onto satisfaction degrees. For the other nodes, $u_7(x_3, x_4) = F_7(u_3(x_3), u_4(x_4))$; $u_8(x_3, x_4, x_5, x_6) = F_8(u_7(x_3, x_4), u_5(x_5), u_6(x_6))$ and $\mathcal{S}(x) = u_9(x) = F_9(u_1(x_1), u_2(x_2), u_8(x_3, x_4, x_5, x_6))$. Functions F_7, F_8, F_9 are the aggregation functions. u_1 to u_6 are called *marginal utilities* (detailed in the following sections). ■

We describe marginal utility functions and aggregators in the next sections.

2.3 Marginal Utilities

Marginal utility function u_i (for $i \in N$) transforms the value on metric X_i onto a level of satisfaction. For criterion 2 in Ex. 3, u_2 is large (resp. small) if the distance between the trajectory and the areas of interest is small enough (resp. large enough) to be able (resp. unable) to perform pictures of good quality. In order to satisfy (1), u_i shall be monotonic: for any two values $x_i, y_i \in X_i$, we have $x_i \succeq_i y_i \iff u_i(x_i) \geq u_i(y_i)$.

2.4 Aggregation functions: the Choquet Integral

There exist many families of aggregation functions that could be used for $F_i : \mathbb{R}^{c_i} \to \mathbb{R}$. The most well-known is the class of weighted sums. However they have limited representation power as they assume independence among criteria. We consider in this paper a very general class of aggregation functions called the Choquet Integral (CI) [5]. It can represent many varieties of interaction among criteria. In order to get an easily interpretable model, we limit to interaction between pairs of inputs. This yields the 2-additive CI which takes the following form (parameterized with a weight vector w):

$$C_w(a) = \sum_{k=1}^{c_i} w_k \, a_k + \sum_{1 \leq k < l \leq c_i} w_{kl}^{\wedge} \, \min(a_k, a_l) + \sum_{1 \leq k < l \leq c_i} w_{kl}^{\vee} \, \max(a_k, a_l). \quad (2)$$

In order to satisfy the monotonicity conditions (1), it is enough that all the weights $w_k, w_{kl}^{\wedge}, w_{kl}^{\vee}$ be non-negative. For a given set of weights, we call its *canonical form* the vector where, $\forall (k,l) \in \{1, ..., c_i\}^2, w_{kl}^{\wedge} = 0$ or $w_{kl}^{\vee} = 0$, which yields the same 2-additive CI. Note that this form always exist, and is equivalent to the previous one in terms of interpretation, as they are a re-writing of the same measure on the set of criteria.

3 Neur-HCI

Neur-HCI is a framework, developed in [1], which implements specific *neural modules* for 2-additive CIs and marginal utilities. Using expert knowledge (in particular, the monotonicity of the global score w.r.t. each criterion and the hierarchy of criteria need to be provided), it builds an ANN representing the given model. Once the model is built, the ANN is trained using data, in order to learn both the weights of the CIs and the parameters of the marginal utilities.

Neur-HCI builds marginal utilities as a convex combination of logistic sigmoids:

$$u_i(x_i) = \sum_{j=1}^{p_i} \frac{r_i^j}{1 - \exp\left(-\lambda_i^j x_i + \beta_i^j\right)} \quad (3)$$

where the weights r_i^j, the steepness λ_i^j and the biases β_i^j need to be tuned. All of the steepness parameters have the sense of $\succsim_i$: non-negative λ_i^j will yield a non-decreasing function, non-positive λ_i^j will yield a non-increasing u_i. p_i is a hyperparameter, selected beforehand, and determines the number of sigmoids used.

The output is thus a fully-trained model, specifically tailored by the expert knowledge. Through diverse techniques, such as clipping, batch-renormalization and the use of non-negative mappings, the learned parameters will respect all of the given constraints by-design. As a consequence, the final model will be guaranteed to be a valid HCI with monotonic utilities, and thus to have the properties given in section 2. This allows to analyse the trained model, not

as an ANN with extraneous parameters, but as a proper HCI with monotonic utilities, which is easily interpretable by an expert. Such an expert can thus, at conception time, easily validate the model based on the parameters, or point out discrepancies with their own expertise leading to diverse possible corrections.

Neur-HCI allows to learn in three settings (from three types of data). Firstly, regression, where the examples are alternatives along with their expected scores. Secondly, classification, where the examples are alternatives labeled by a satisfaction class (e.g. *Bad*, *Average*, *Good* for a 3-classes setting). Finally, pairwise comparisons, where the examples are pairs of alternatives, and the information about which of both alternatives is preferred to the other. The detailed learning processes can be found in the original paper [1].

4 Identifiability of HCIs

The next result (the proof of which is not provided due to space limitation) presents the identifiability of HCI model:

Theorem 1. *Let $X \in \mathbb{R}^n$. Let $\mathcal{F}$ and $\mathcal{F}'$ two HCIs with monotonic marginal utilities $u_1, ..., u_n$ (resp, $u'_1, ..., u'_n$), such that, $\forall x \in X, \mathcal{F}(x) = \mathcal{F}'(x)$, and the same hierarchy H of CIs. Then, for all $i \in \{1, ..., n\}, u_i = u'_i$. Moreover, let C_w (resp $C_{w'}$) one of the CIs of $\mathcal{F}$ (resp $\mathcal{F}'$) such that C_w and C'_w occupy the same position in H, then both CIs have the same parameterization (i.e. w and w' have the same canonical form).*

This means that two HCIs which are equal for all possible alternatives will have equivalent parameterizations. Thus, if it exists, there is a unique model satisfying a fully-determined problem. This is crucial for interpretability, as the parameters hold the meaning of a HCI; this unicity ensures that a model holds a single interpretation, rather than two, possibly contradictory ones.

Nonetheless, this does not ensure that two models that are close in terms of outputs have close parameterizations. While formal work will be needed in order to demonstrate this property, we show in section 5 that, in practice, training several Neur-HCI models on datasets that differ slightly yield HCIs that have very close parameters. This is necessary in real-life applications, as any dataset holds inherent randomness, due e.g. to noisy data collection or labeling errors.

5 Stability of Neur-HCI

The performance of Neur-HCI in terms of classic error metrics (classification loss, mean squared error) has been evaluated on diverse multicriteria datasets, against several other models, including a similarly sized ANN. The results, which can be found in [1], show that Neur-HCI performs globally well, and can beat the unconstrained ANN on several datasets. We will here evaluate its robustness and stability; that is, check whether the parameters of several learned models are similar when trained on the same dataset, or slightly altered versions of the same

dataset. Such alterations involve adding noise, or training on different parts of the dataset.

In the artificial case, we used an HCI with marginal utilities to generate examples, then trained our models on such data. In practice, every trained model converged to the real one in the large sample limit (fewer than 100 samples were enough in a 6-dimensional case). Nonetheless, due to limited space, we will focus on a real-life case.

We illustrate here the stability of Neur-HCI on the auto-MPG dataset[4]. It holds 292 alternatives described by 7 criteria. Fig. 2 shows the parameters values learned by 100 Neur-HCI networks in the regression setting, with a single CI, each being trained on a different split of the dataset (80% of the dataset was randomly chosen and used to train each model). We use a more compact version of the parameters called their Möbius representation. One can see that the parameters values remain similar for all models, indicating a strong influence of criterion 4, followed by criteria 3 and 5. The marginal utilities are also very similar for all models, as shown in Fig 3. This offers a first empirical confirmation of the stability of Neur-HCI, as we obtain close models for close training sets.

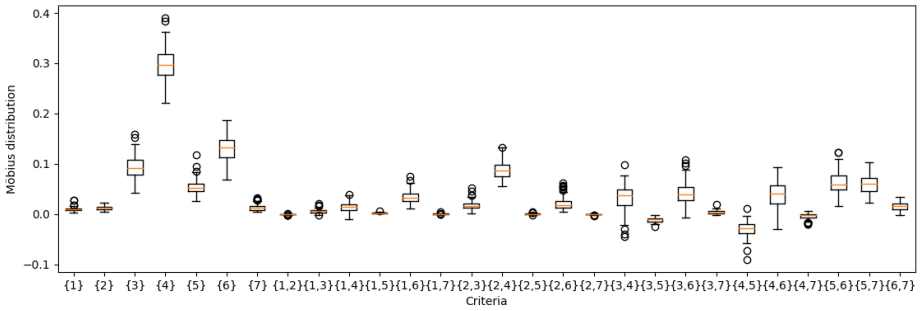


Fig. 2. Stability of the CI parameters for 100 splits of the MPG dataset

6 Conclusion

We have presented in this paper the Neur-HCI framework for learning highly-constrained interpretable models. We have first shown the identifiability of the HCI model: the HCI parameters are uniquely specified given its output. It thus makes sense to interpret the values of the HCI parameters.

Moreover, we have shown the empirical stability of the learned model: two networks trained on two similar datasets converge to the same final parameterization. This is encouraging, as it figures that the interpretation of a trained model will not be highly influenced by the inevitable noise present in real-life

[4] collected from http://archive.ics.uci.edu/ml/index.php

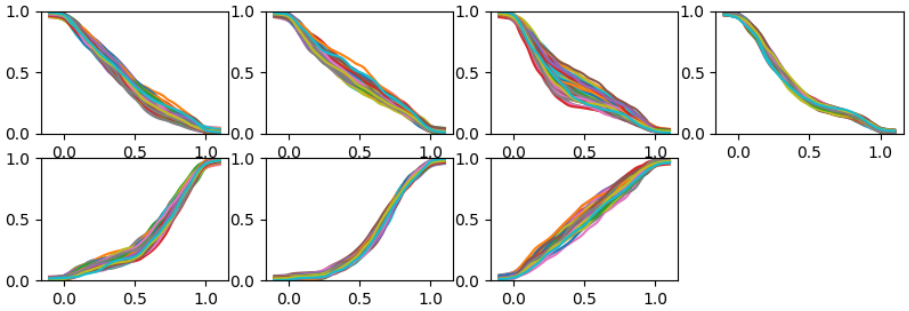


Fig. 3. Graphs of the learned marginal utilities for 100 splits of the MPG dataset; note that the input values were normalized linearly in the unit interval

datasets. The extent of such noise-tolerance and stability will be investigated thouroughly in future work, along with theoretical analyses of such robustness.

In terms of applications, this allows to use Neur-HCI to learn models which can easily be checked by a field-expert and used as-is. It can also be used as an analysis tool for extracting formal rules and properties present in a given dataset, in order to configure a constraints-based formal model. In particular, this stability makes it relevant for uses in safety-critical applications, where it is necessary for the model to be trustable. This involves decision-aiding settings in varied fields such as Defense, where failures must be avoided at all cost.

References

1. Bresson, R., Cohen, J., Hüllermeier, E., Labreuche, C., Sebag, M.: Neural representation and learning of hierarchical 2-additive choquet integrals. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. pp. 1984–1991 (2020)
2. Došilović, F.K., Brčić, M., Hlupić, N.: Explainable artificial intelligence: A survey. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). pp. 0210–0215 (2018)
3. Fallah Tehrani, A., Labreuche, C., Hüllermeier, E.: Choquistic Utilitaristic Regression. In: DA2PL-14 (2014)
4. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. ICLR 2015 (12 2014)
5. Grabisch, M., Roubens, M.: Application of the Choquet integral in multicriteria decision making. In: Fuzzy Measures and Integrals - Theory and Applications, pp. 348–374. Physica Verlag (2000)
6. Labreuche, C., Fossier, S.: Explaining Multi-Criteria Decision Aiding Models with an Extended Shapley Value. In: IJCAI. pp. 331–339 (2018)
7. Ng, A.Y.: Feature selection, regularization, and rotational invariance. In: Proceedings of the Twenty-First International Conference on Machine Learning. p. 78. ICML '04, Association for Computing Machinery, New York, NY, USA (2004)
8. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014)

Étiquetage non supervisé de représentations de chiffres manuscrits volées

Thomas Thebaud[1,2][0000−0001−8953−7872], Gaël Le Lan[2][0000−0002−1493−5777], and Anthony Larcher[1][0000−0003−4398−0224]

[1] LIUM - Le Mans University, 72085 Le Mans, France
{firstname}.{lastname}@univ-lemans.fr https://lium.univ-lemans.fr/
[2] Orange Labs, 35510 Cesson-Sevigne, France
{firstname}.{lastname}@orange.com

Abstract. L'omniprésence des systèmes d'authentification basés sur la biométrie nécessite de bien protéger les données biométriques. Nous considérons l'attaque d'un système biométrique basé sur le tracé de chiffres sur écran tactile : dans le cas d'un piratage des données d'enrôlement, c'est-à-dire des représentations (*embeddings*) des tracés, non annotées, issues d'un réseau LSTM de poids inconnus (*Long Short Term Memory*), nous proposons une méthode d'inférence de la valeur des chiffres inspirée par des algorithmes de traduction non supervisée.
Cette méthode implique un regroupement non supervisé en 10 classes des ces représentations, et l'identification du label de chaque classe en comparant leur distribution à celle d'autres représentations de données ouvertes comparables, issues d'un autre réseau LSTM de poids connus. L'annotation est un processus en deux étapes, comprenant l'utilisation d'un algorithme génétique pour calculer les N meilleures annotations candidates, suivi d'un affinage de ces N candidates afin de déterminer la meilleure. Expérimentalement, la méthode proposée permet d'inférer les bonnes valeurs pour tous les groupes, sur 100 essais aléatoires effectués sur des séparations différentes des données.

Keywords: Label inference · Handwritten digits · Density matching · Privacy · Long Short Term Memory · Recurrent Neural Network · Genetic search .

1 Introduction

L'utilisation de la biométrie pour l'authentification [5] replace les données personnelles au centre des systèmes de sécurité. La plupart des systèmes de biométrie compressent des relevés biométriques, comme la démarche [8], la voix [11], le visage [9] ou les chiffres manuscrits [14] [13] [7], en représentations de grande dimension via des réseaux de neurones profonds. Ces représentations sont sujettes à stockage et transfert pour l'enrôlement et l'authentification d'un utilisateur. Le stockage et le transfert présentent des brèches potentielles dans la sécurité de ce système de biométrie, même si pour l'instant, des représentations non annotées

seules ne sont pas suffisantes pour les attaques existantes [9] visant à reconstruire la donnée biométrique originale.

Dans ce papier, nous étudions la menace potentielle d'un piratage d'un ensemble de représentations non annotées, toutes extraites de tracés de chiffres manuscrits, dans le cadre d'un système d'authentification biométrique [14]. Ces représentations étant issues de chiffres tracés sur smart-phone, le nombre de classes est donc connu et fixé à 10, et nous faisons l'hypothèse que le système utilisé pour les extraire est un réseau de neurones récurrent LSTM (une architecture standard pour ce type de séquences [7]) dont les poids ne sont pas connus.

Grâce au petit nombre de classes et à la nature simple des données, nous supposons qu'un attaquant peut se procurer ou fabriquer une autre base de données de tracés de chiffres manuscrits, entraîner son propre extracteur et produire son propre jeu de représentations annoté. Inspirés par plusieurs méthodes de traduction non supervisée [4] [3] [15] [1], nous cherchons si il est possible d'appairer l'ensemble fabriqué des représentations annotées et l'ensemble volé des représentations non annotées avec des méthodes statistiques. Cet appariement permettrait d'annoter ces représentations volées en les projetant dans un espace connu, ce qui présente un problème de sécurité. En effet, Mai et al. [9] montrent qu'il est possible reconstruire des visages à partir de leurs représentations, en possédant en boite noire l'extracteur qui a servi à les produire. Ici, nous utilisons uniquement les représentations non annotées pour calculer la transformation de leur espace vers l'espace de sortie d'un extracteur connu, et ainsi deviner la valeur de leurs chiffres. Cet article est une version résumée et traduite depuis l'anglais de nos travaux [12].

2 Scénario d'attaque proposé

Nous voulons annoter un ensemble $\mathbf{U}$ de représentations d'enrôlement volées, supposé produit par un système d'authentification biométrique [14], à partir de chiffres manuscrits tracés sur un écran tactile, comme illustré figure 1. L'ensemble $\mathbf{U}$ est composé de N représentations en dimension D, extraites de l'avant dernière couche d'un réseau LSTM entraîné à reconnaître le chiffre (0 à 9) à partir d'une séquence de points en 2D de longueur variable.

On suppose l'attaquant capable de produire ou trouver son propre jeu de données, utilisable pour produire un ensemble annoté statistiquement semblable à celui volé. Pour simuler ce scénario d'attaque, on entraîne un réseau LSTM avec un ensemble disjoint de séquences 2D, produites par des utilisateurs différents de ceux de l'ensemble $\mathbf{U}$. Cet ensemble de représentations annoté sera nommé l'ensemble $\mathbf{L}$. Pour exploiter les représentations volées, nous proposons une méthode pour trouver la fonction d'alignement entre l'espace de $\mathbf{U}$ et de $\mathbf{L}$. Une fois projetées dans l'espace de $\mathbf{L}$, les représentations de $\mathbf{U}$ pourront donc être annotées. Les notions de permutation et de rotation optimale sont liées dans la plupart des travaux présentés [3] [4] [1], et notre méthode utilise les deux.

La méthode proposée se décompose en 4 étapes, illustrées figure 1 :

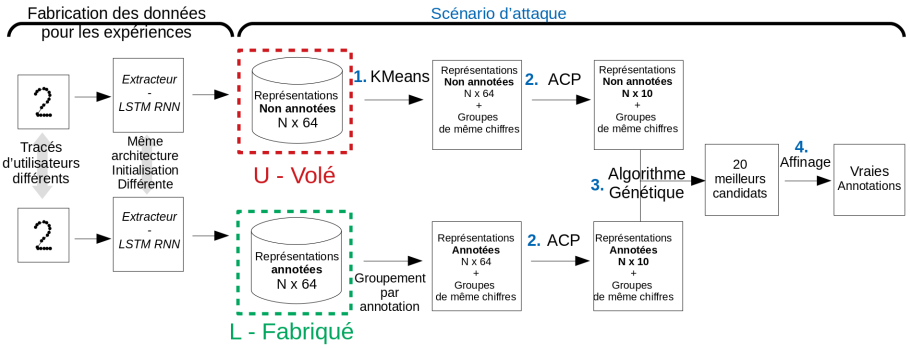


Fig. 1. Illustration de la fabrication des représentations, et de la méthode pour trouver leurs labels.

1. Regrouper les représentations de **U** en 10 groupes, en espérant que chaque groupe représente un chiffre.
2. Appliquer une ACP globale sur les ensembles **U** et **L** pour les projeter sur un espace en 10 dimensions.
3. Trouver les permutations candidates les plus probables entre les 2 ensembles, en utilisant un score de similarité.
4. Affiner les rotations associées aux meilleurs candidats pour trouver la rotation optimale, et annoter les groupes de **U**.

Nos principales contributions, relatives à l'étape 3 (Annotation) sont présentées dans les sous-sections 3.2 et 3.3.

3 Annotation

Pour considérer **U** annoté, nous cherchons à annoter chacun de ses 10 groupes, en l'appairant avec un groupe de **L** annoté. Pour représenter le lien entre les groupes de **U** et **L**, on utilisera la matrice de permutation $\mathbf{P} = (p_{ij})_{i,j\in[\![0,9]\!]^2}$, une matrice bi-stochastique composée de 0 et de 1.

Cette section se décompose en 4 parties :

1. L'utilisation de Procrustes [2] entre les centres des groupes **U** et de **L**, pour approximer la meilleure rotation possible pour une permutation donnée.
2. Le score utilisé pour mesurer la pertinence d'une rotation.
3. La recherche de la meilleure permutation dans l'espace des permutations possibles à l'aide d'un algorithme génétique.
4. L'affinage des meilleures rotations candidates pour trouver la rotation optimale.

3.1 Rotation optimale pour une permutation donnée

Pourquoi considérer que la fonction de transfert est une rotation? Mikolov et al. (2013) [10] a montré que la transformation entre deux espaces

de représentations de mots peut être bien représentée par une transformation linéaire (i.e. une matrice $\mathbf{W}$). Les ensembles $\mathbf{U}$ et $\mathbf{L}$ étant centrés et chaque représentation préalablement normalisée (de norme unitaire), les vecteurs colonnes de $\mathbf{W}$ sont donc de norme 1, donc, et après avoir appliqué une ACP, les représentations seront projetées dans des espaces orthogonaux. **Nous considérons donc que la fonction de transfert est une rotation**.

Analyse de Procrustes Procrustes [2] calcule la rotation optimale pour lier 1 à 1 deux ensembles de points, ici $\mathbf{U} \in \mathbb{R}^{N \times D}$ et $\mathbf{L} \in \mathbb{R}^{N \times D}$. Ici nous utiliserons les 10 centres $\mathbf{C_U}$ et $\mathbf{C_L}$ des groupes de $\mathbf{U}$ et $\mathbf{L}$. Dans le cas où la permutation $\mathbf{P}$ entre les deux ensembles de points $\mathbf{C_U}$ et $\mathbf{C_L}$ est connue, la matrice de rotation $\mathbf{W}$ peut être trouvée en résolvant le problème des moindres carrés suivant :

$$\min_{W \in \mathbb{R}^{D \times D}} \|\mathbf{C_U W} - \mathbf{P C_L}\|_2^2 \tag{1}$$

Évaluation d'une rotation donnée Pour sélectionner la permutation la plus pertinente, nous choisissons d'évaluer sa rotation correspondante. Nous faisons l'hypothèse que les distributions statistiques des représentations de chaque chiffre sont suffisamment différentes pour pouvoir apparier les groupes représentant le même chiffre entre les deux espaces. Pour prendre en compte cette distribution, nous modélisons chaque ensemble de représentations par un modèle à mélange de gaussiennes, un GMM. Le nombre de gaussiennes n'est pas égal au nombre de groupes, mais déterminé par le minimum du Critère d'information Bayesien.

Pour évaluer la pertinence d'une rotation donnée $\mathbf{W}$, nous mesurons la distance entre un ensemble de représentations $\mathbf{U}$, après application de cette rotation ($\mathbf{U_W}$), et un GMM entraîné sur l'autre ensemble de représentations $\mathbf{L}$ (GMM_L). Cette distance est le score global de log-vraisemblance $Score(U_W, GMM_L)$.

Pour prendre en compte la réversibilité de la rotation (i.e. l'inversibilité de $\mathbf{W}$), le score utilisé sera le maximum des 2 scores $Score(U_W, GMM_L)$ et $Score(L_{W^t}, GMM_U)$, avec $\mathbf{W^t}$ la rotation inverse de $\mathbf{W}$ et GMM_U le GMM entraîné sur l'ensemble $\mathbf{U}$. Un score plus haut signifie une plus grande similarité, en prenant le maximum des 2, nous utilisons le meilleur cas à chaque fois. Pour le reste de l'article, nous utiliserons l'opposé de ce score ($-Score(U, L, W)$) comme la fonction à minimiser pour trouver la meilleure rotation.

3.2 Recherche Génétique

Pour chaque permutation possible, nous voulons calculer sa rotation associée, et mesurer la pertinence de cette rotation pour aligner les deux ensembles $\mathbf{U}$ et $\mathbf{L}$. Pour minimiser le score décris ci-dessus, nous explorons l'espace des permutations $\mathbf{P} \in [\![0, 1]\!]^{10 \times 10}$. Pour trouver la meilleure permutation, nous devrions tester les $10! = 3628800$ possibles. Pour limiter le nombre de permutations testées, nous utilisons un algorithme génétique. Cet algorithme considère chaque permutation comme un chromosome, et parcours l'espace des possibles à travers des mutations et combinaisons, permettant de trouver les meilleurs candidats sans faire le test de chaque combinaison.

3.3 Affinage

Les rotations associées à chaque permutation ont été calculées en utilisant les centres des groupes de **U** et **L**, et pas toutes leurs représentations. Par conséquent, le meilleur candidat choisi par l'algorithme peut ne pas être la bonne permutation. Pour affiner et reclasser les meilleurs candidats proposés par l'algorithme génétique, nous proposons d'utiliser une optimisation stochastique des matrices de rotations associées à ces candidats. La rotation candidate avec le meilleur score après affinage est supposée être la rotation recherchée.

Notre affinage est inspiré par [15], qui utilise une descente de gradient sur une matrice de rotation, en utilisant deux faibles contraintes : orthogonalité et unicité du déterminant. Pour chacun des K meilleurs candidats, nous affinons la matrice de rotation **W** associée avec la méthode d'optimisation Adam [6] pour minimiser la somme non pondérée des fonctions de coût suivantes :

1. Le score global de log-vraisemblance $-Score(\mathbf{U}, \mathbf{L}, \mathbf{W})$
2. La valeur absolue du déterminant de **W** : $|\log(\det \mathbf{W})|$
3. La différence $u_i - (\mathbf{W^t} \times \mathbf{W} \times u_i), \forall u_i \in \mathbf{U}$

La première contrainte affine la matrice **W** vers la transformation optimale entre les ensembles **U** et **L**. Les deux autres imposent des conditions pour que **W** reste une rotation (orthogonale et déterminant unitaire). Après quelques douzaines d'époques, les scores se stabilisent et on obtient $\mathbf{W}^*$, la version affinée de **W**. Chaque instance de $\mathbf{W}^*$ est évaluée avec le score global de log-vraisemblance et celle avec le score minimum donne le candidat recherché, la permutation qui associe correctement les annotations les deux ensembles.

4 Données et pré-traitement

4.1 Données

Les données sont issues de deux jeux produits par l'université de Madrid, décris dans [14] et [13]. Ils contiennent 16350 tracés de chiffres de longueur variable, produits par 310 utilisateurs, un tracé étant une séquence de points en 2D. Chaque chiffre est représenté par un dixième des séquences. Cette base est séparée en 4 jeux disjoints, contenant chacun les séquences d'un quart des utilisateurs, la séparation étant déterminée aléatoirement. Ces 4 jeux seront nommés : **Train U**, **Test U**, **Train L** et **Test L**.

Pour multiplier les expériences avec une quantité limitée de données, nous avons utilisé 100 séparations différentes du jeu original pour mener 100 expériences différentes, avec 100 paires différentes de jeux **U** et **L**.

4.2 Architecture des réseaux

L'architecture de l'extracteur, connue par l'attaquant, est un réseau de neurones récurent LSTM. On entraîne deux réseaux avec les séquences en 2 dimensions

en entrée, et un vecteur de sortie de dimension $D = 64$. Pour l'entraînement, la sortie du LSTM est passée par une couche connectée en 10 dimensions et un softmax, pour prédire la valeur du chiffre. Les réseaux sont entraînés avec les jeux de *Train* en utilisant l'entropie croisée comme fonction de coût. L'entraînement s'arrête quand chaque réseau arrive à au moins 96% de précision sur les jeux de *Test*. Les représentations sont extraites en prenant les sorties du LSTM pour chacun des jeux de *Test*. Les jeux *Train U* et *Test U* seront utilisés pour le premier réseau, et les jeux *Train L* et *Test L* pour le deuxième.

4.3 Pré-traitement

Les représentations des 2 sets ont été normalisées et centrées (voir [3]). Chaque jeu a été projeté en $D = 10$ dimensions avec une ACP (voir [12]). Les représentations de l'ensemble non annoté **U** ont été regroupées en 10 groupes avec l'algorithme K-moyennes, et celles de l'ensemble annoté **L** ont été regroupées par classe en utilisant leurs annotations. Les ensembles **U** et **L** sont donc maintenant composés de 3450 à 4560 vecteurs (en fonction de la séparation des utilisateurs) en 10 dimensions.

5 Expériences

5.1 Pertinence du score de log-vraisemblance

Pour vérifier la pertinence de notre score, nous avons calculé sur un exemple les scores associés aux 10! permutations possibles entre les groupes de **U** et de **L**. La permutation recherchée a eu le troisième meilleur score sur 10!, la classant bien parmi les meilleures, mais pas la meilleure. Nous avons choisi de considérer les 20 meilleures permutations à affiner, pour ne pas passer à côté de la permutation voulue.

5.2 Recherche génétique

Nous cherchons les permutations les plus probables dans l'espace des permutations possibles. Pour faire cette recherche, nous utilisons un algorithme génétique, en considérant chaque matrice de permutation (représentant le lien entre un groupe non annoté et un groupe annoté) comme un chromosome. L'algorithme génétique utilisé est détaillé dans [12]. Celui-ci permet de trouver les 20 meilleures permutations selon le score de log-vraisemblance. Un exemple de résultat est présenté dans le tableau 1. Dans cet exemple, le candidat recherché, **[0 1 2 3 4 5 6 7 8 9]**, n'a été classé que troisième avant affinage, justifiant l'intérêt de l'affinage dans la section ci-dessous. Pour avoir des résultats plus précis, nous avons fait tourner cet algorithme sur nos 100 paires de jeux de données, et le candidat recherché à fini 1^{er} 62 fois sur 100, et dans le pire des cas a été classé 20^{eme} une fois.

5.3 Affinage

La recherche génétique nous a permis de trouver rapidement un ensemble de candidats pertinents, contenant le candidat recherché, maintenant nous souhaitons retrouver ce bon candidat, grâce à l'algorithme d'affinage présenté section 3.3.

Pour chaque permutation sélectionnée, sa matrice de rotation associée est calculée, puis affinée pendant 200 époques en suivant la configuration décrite section 3.3, et son score est recalculé. Les résultats sont présentés dans le tableau 1. Le candidat recherché obtient effectivement le meilleur score dans cet exemple, c'est à dire que chaque groupe est correctement annoté.

Table 1. Tableau des scores et rangs des 10 premiers candidats sur les 20 sélectionnés, avant et après affinage.

Rangs Après Affinage	Rangs Avant Affinage	Permutations candidates										Scores Après Affinage	Scores Avant Affinage
1*	3	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**-5.914**	-1.310
2	9	**0**	**1**	**2**	**3**	**4**	8	5	**7**	6	**9**	-5.827	-0.559
3	6	8	**1**	**2**	**3**	7	6	5	4	0	**9**	-5.770	-0.937
4	2	8	**1**	**2**	**3**	**4**	6	5	**7**	0	**9**	-5.603	-1.360
5	1	**0**	4	**2**	**3**	1	**5**	**6**	**7**	**8**	**9**	-5.602	**-1.465**
6	10	**0**	2	4	**3**	1	**5**	**6**	**7**	**8**	**9**	-5.420	-0.553
7	13	**0**	7	**2**	**3**	**4**	**5**	**6**	1	**8**	**9**	-5.335	-0.457
8	20	8	**1**	**2**	**3**	**4**	0	5	**7**	6	**9**	-5.334	-0.305
9	17	**0**	**1**	**2**	**3**	**4**	**5**	8	**7**	6	**9**	-5.322	-0.361
10	7	7	**1**	**2**	**3**	**4**	6	5	0	**8**	**9**	-5.269	-0.867

Pour avoir des résultats plus précis, nous avons fait tourner cet algorithme sur 100 paires de jeux de données, et le candidat recherché à fini 1^{er} à chaque fois, permettant d'annoter correctement les groupes de **U** à chaque fois.

6 Conclusion et futurs travaux

Ce papier présente une méthode d'alignement statistique de représentations à haut niveau de chiffres tracés sur écran tactile, dans le cadre d'un vol de données. Notre méthode, inspirée d'attaques existantes et de techniques traduction non supervisées, vise à retrouver les annotations des représentations volées. A partir de ces représentations que nous savons avoir été produites par un réseau LSTM, nous produisons un jeu de données similaire à partir de données différentes et d'un réseau de même architecture, pour inférer les labels des données volées par comparaison. Un algorithme génétique nous permet de trouver les meilleurs candidats pour l'annotation du jeu volé. Un affinage des premiers candidats permet de retrouver l'annotation correcte. L'application de cette méthode sur 2x100 jeux de données, avec 2x100 réseaux différents entraînés a montré que l'annotation

correcte arrivait en première position à chaque fois. Nos futurs travaux seront focalisés sur l'allègement des contraintes imposées (grand nombre de classes, architecture inconnue, nouvelles biométries), sur la reconstruction des signaux originaux et leur utilisation pour la fraude du réseau attaqué. Ce travail rappelle l'importance de la protection des données personnelles, en particulier pour les systèmes de biométrie, et ouvre des perspectives pour de futurs modèles d'attaque et de défense.

References

1. Goran Glavas, Robert Litschko, Sebastian Ruder, and Ivan Vulic. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. *arXiv preprint arXiv:1902.00508*, 2019.
2. John C Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.
3. Edouard Grave, Armand Joulin, and Quentin Berthet. Unsupervised alignment of embeddings with wasserstein procrustes. *arXiv preprint arXiv:1805.11222*, 2018.
4. Yedid Hoshen and Lior Wolf. Non-adversarial unsupervised word translation. *arXiv preprint arXiv:1801.06126*, 2018.
5. Anil K Jain, Karthik Nandakumar, and Abhishek Nagar. Biometric template security. *EURASIP Journal on advances in signal processing*, 2008:1–17, 2008.
6. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
7. Gaël Le Lan and Vincent Frey. Securing smartphone handwritten pin codes with recurrent neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2612–2616. IEEE, 2019.
8. Lily Lee and W Eric L Grimson. Gait analysis for recognition and classification. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 155–162. IEEE, 2002.
9. Guangcan Mai, Kai Cao, Pong C Yuen, and Anil K Jain. On the reconstruction of face images from deep face templates. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1188–1202, 2018.
10. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
11. David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333. IEEE, 2018.
12. Thomas Thebaud, Gaël Le Lan, and Anthony Larcher. Unsupervised labelling of stolen handwritten digit embeddings with density matching. In *International Workshop on Security in Machine Learning and its Applications (SiMLA)*, 2020.
13. Ruben Tolosana, Ruben Vera-Rodriguez, and Julian Fierrez. Biotouchpass: Handwritten passwords for touchscreen biometrics. *IEEE Transactions on Mobile Computing*, 2019.
14. Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, and Javier Ortega-Garcia. Incorporating touch biometrics to mobile one-time passwords: Exploration of digits. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.
15. Chunting Zhou, Xuezhe Ma, Di Wang, and Graham Neubig. Density matching for bilingual word embedding. *arXiv preprint arXiv:1904.02343*, 2019.

Multi-Radar Tracking Optimization for Collaborative Combat

Nouredine Nour[1], Reda Belhaj-Soullami[1], Cédric L.R. Buron[2], Alain Peres[3], and Frédéric Barbaresco[3]

[1] NukkAI, Paris, France (www.nukk.ai), {nnour, rbelhaj-soullami}@nukk.ai
[2] Thales Research & Technology, 1 av. Augustin Fresnel, Palaiseau France
cedric.buron@thalesgroup.com
[3] Thales Land & Air Systems, 3 Avenue Charles Lindbergh, 94150 Rungis, France
{alain.peres, frederic.barbaresco}@thalesgroup.com

Abstract. Smart Grids of collaborative netted radars accelerate kill chains through more efficient cross-cueing over centralized command and control. In this paper, we propose two novel reward-based learning approaches to decentralized netted radar coordination based on black-box optimization and Reinforcement Learning (RL). To make the RL approach tractable, we use a simplification of the problem that we proved to be equivalent to the initial formulation. We apply these techniques on a simulation where radars can follow multiple targets at the same time and show they can learn implicit cooperation by comparing them to a greedy baseline.
Keywords: Netted sensors, Reinforcement Learning, Actor Critic, Evolutionary Algorithms, Multi-Agent Systems

1 Introduction

Despite great interest in recent research, in particular in China [2, 14] micromanagement of sensors by centralized command and control drives possible inefficiencies and risk into operations. Tactical decision making and execution by headquarters usually fail to achieve the speed necessary to meet rapid changes. Collaborative radars with C2 must provide decision superiority despite the attempts of an adversary to disrupt OODA cycles at all level of operations. Artificial intelligence can make a contribution for the purposes of coordinated conduct of the action, by improving the response time to threats and optimizing the allocation and the distribution of tasks within elementary smart radars.

In order to address this problem, Thales and the private research lab NukkAI have been collaborating to introduce novel approaches for netted radars. Thales provided the simulation modeling the multi-radar target allocation problem and NukkAI proposed two novel reward-based learning approaches for the problem.

In this paper, we present these two approaches: Evolutionary Single-Target Ordering (ESTO), which is based on evolution strategies and an RL approach based on Actor-Critic methods. To make the RL method tractable in practice, we introduce a simplification of the problem that we prove to be equivalent to

solving the initial formulation. We evaluate our solutions on diverse scenarios of the aforementioned simulation. By comparing them to a greedy baseline, we show that our algorithms can learn implicit collaboration. The paper is organized as follows: section 2 introduces the related works. The problem is formalized in section 3. In section 4, we describe the proposed approaches. Section 5 presents the results of our simulations. section 6 concludes the paper.

2 Related works

Decentralized target allocation in a radar network has gained a lot of interest recently [4]. For this problem, resolution through non-cooperative game formalism [15] reaches good performance, but only considers mono-target allocation. Bundle auctions algorithm [7] overcomes this limitation; still, none of these approaches are able to model the improvement provided by multiple radars tracking the same target. Another suitable method is reward-based machine learning, that can either take the form of evolutionary computation [12] or reinforcement learning (RL). Recent successes in multi-agent RL were obtained by adapting mono-agent deep RL methods to the multi-agent case, most of them based on policy gradient approaches [16] with a centralized learning and decentralized execution framework [6, 8, 1]. In the policy gradient algorithm family, actor-critic methods [10] relying on a centralized critic have empirically proven to be effective in the cooperative multi-agent partially observable case [8]. However, the size of the action space requires to adapt these approaches for our problem.

3 Problem statement

In this paper, we consider that each radar can track a set of targets that move in a 2D space (for sake of simplicity, elevation is ignored). The targets are tracked with an uncertainty ellipse, computed using a linear Kalman model. We assume that the radars have a constant Signal-to-Noise Ratio (SNR) on target (adaptive waveform to preserve constant SNR on target), can communicate without limitations, have a limited field of view (FOV) and a budget representing the energy they can spend for tracking capabilities. Their Search mode are not simulated but taken into accoun the the constrained time budget for active track modes.

Let n be the number of radars and m the number of targets. In our model, an action of a radar is the choice of the set of targets to track. If multiple radars track the same target, the **uncertainty area** is the superposition of the uncertainty ellipses. We define a utility function $\mathcal{U}$ measuring the global performance of the system. Let l_i^j be the elementary radar i budget needed to track target j, L_i the budget of radar i, $\mathcal{E}_i^j$ the uncertainty ellipse on target j for radar i and $S(\mathcal{E}_j^i)$ the area of $\mathcal{E}_j^i$ (improvement is possible by considering covariance matrix of trackers). The problem can be expressed as a constraint optimization problem:

$$\text{maximize } \mathcal{U} = \frac{1}{m}\sum_{j=1}^{m} \exp\left[-S\left(\bigcap_{i=1}^{n} \mathcal{E}_i^j\right)\right] \text{ such that: } \forall i \leq n, \sum_{j=1}^{m} l_i^j \leq L_i \quad (1)$$

4 Task allocation in a radar network

4.1 Evolutionary Single-Target Ordering (ESTO)

ESTO is a centralized training with decentralized execution black-box optimization method. Based on contextual elements, agents define a preference score for each target. They then choose the targets to track greedily based on this score, until their budget is met. The preference score is computed by a parametrized function optimized to maximize the utility using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [5]. CMA-ES optimizes the parameters by sampling them from a normal distribution and updating both the mean and the covariance matrix based on the value of the utility function. We modelized ESTO's preference score function with a linear model with 9 input features including information on the target, the radar, and the position of the other radars. We also propose a variant of ESTO, called ESTO-M which takes into account 2 additional features based on inter-radar communication: the estimated target utility and the evolution of the estimated utility from the previous step.

4.2 The Reinforcement Learning approach

Dec-POMDP formulation. Collaborative multi-agent reinforcement learning typically relies on the formalization of the problem as a Dec-POMDP [11]. It is defined as a tuple $\langle D, S, A, P, R, \Omega, O\rangle$, where D is the set of agents ($|D| = n$), S is the state space, $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ is the joint action set, P is the transition function, R is the reward function, Ω is the set of observations and O is the observation function. A state can be described as a tuple containing the true position and velocity of each target, the position of the other radars, and the Kalman filter parameters for each radar-target pair. The transition dynamics include the update of the Kalman filters. The problem is not fully observable: the radars approximate the position and speed of the targets and can't access to the Kalman filters of the others. Radars rely on a $m \cdot n_f$ real-valued vector, with n_f the number of features: estimated position, speed, etc.

Our approach is based on centralized learning and decentralized execution. Although this approach may lead to stationarity issues, it is widely used in practice and yields good results in multi-agent RL [3, 8]. When applying policy, the probability of tracking targets beyond FOV is set to 0, the probabilities of remaining targets are updated accordingly. Agents share the same network but its input values depend on the radar and the targets.

In this setting, the size of the action space corresponds to all the possible allocations of sets of targets to each agent $\forall i \leq n, \mathcal{A}_i = \mathcal{P}(\llbracket 1, m \rrbracket)$ and $|\mathcal{A}_i| = 2^m$, *i.e.* the powerset of all targets. To tackle this issue, we propose a new formalization of the problem where the radars choose the target sequentially, and prove the equivalence between the solutions of the two formalisms.

Definition 1 (Sequential choice Dec-POMDP).
Let $M = \langle D, S, A, P, R, \Omega, O\rangle$ be a Dec-POMDP for our problem. Let $M' = \langle D, S', A', P', R', \Omega, O'\rangle$ with $S' = S \cup \{s \otimes \varepsilon | s \in S, \varepsilon \in \mathcal{P}(\llbracket 1, m \rrbracket \cup \dagger)^n\}$ where

ε is the set of targets tracked by each agent and the symbol † means that its allocation is finished; $s \otimes \varepsilon$ is a notation for the new (couple) state in S'. In a state $s \otimes \varepsilon \in S'$, the set of allowed actions of agent i is $(\llbracket 1, m \rrbracket \cup \{\dagger\}) \setminus \varepsilon_i$. The observation, state-transition and reward functions are defined as:

$$\forall a \in A', \forall (\varepsilon, \varepsilon') \in (\mathcal{P}(\llbracket 1, m \rrbracket))^n)^2, \forall (s, s') \in S^2, O'(s \otimes \varepsilon, s' \otimes \varepsilon) = O(s, s')$$

$$P'(s \otimes \varepsilon, a, s' \otimes \varepsilon') = \begin{cases} P(s, \varepsilon, s') \text{ if } a = (\dagger, \dots, \dagger), \varepsilon' = \varnothing \\ 1 \text{ if } s = s', \varepsilon'_j = \varepsilon_j \cup \{a_j\} \forall j \leq n \\ 0 \text{ else} \end{cases}$$

$$R'(s \otimes \varepsilon, a, s' \otimes \varepsilon') = \begin{cases} R(s, \varepsilon, s') \text{ if } a = (\dagger, \dots, \dagger), \varepsilon' = \varnothing \\ 0 \text{ else} \end{cases}$$

This new Dec-POMDP can be solved much more easily than the initial one. We now look for a solution of the initial Dec-POMDP from the sequential choice Dec-POMDP. In the rest of the article, we denote by V (resp. V') the averaged state value function in M, (resp. M'): $V_\pi(\rho) = \mathbb{E}_{a_t \sim \pi, s_0 \sim \rho}\left(\sum_{t=1}^{T} R(s_t, a_t, s'_t)\right)$. For space reasons, only sketches of the lemma proofs are provided.

Definition 2 (Policy transposition). *We define the policy transition function ϕ from the set of policies in M' to the set of policies in M as*

$$\forall j \in \llbracket 1, n \rrbracket, \phi_j(\pi')(\varepsilon|\omega) = \sum_{\{i_k\}=\varepsilon} \pi'_j(i_1|\omega)\pi'_j(i_2|\omega \otimes \{i_1\}) \dots \pi'_j(\dagger|s \otimes \{i_1, \dots, i_p\})$$

Lemma 1 (Value equivalence). *Let $\pi = \phi(\pi')$. Let ρ be a probability distribution on S and π' a policy on M'. Then $V'_{\pi'}(\rho) = V_{\phi(\pi)}(\rho)$.*

Proof (sketch). By definition 1, the result holds iff $\forall (s, \varepsilon, s') \in S \times \mathcal{P}(\llbracket 1, m \rrbracket)^n \times S$,

$$\rho(s)\pi(\varepsilon|\omega)P(s, \varepsilon, s') = \sum_{\{i_k\}=\varepsilon} \rho(s)P'(s, i_1, s \otimes i_1)\pi'(i_1|\omega) \dots P'(s \otimes \varepsilon, \dagger, s')\pi'(\dagger|\omega \otimes \varepsilon)$$

By using definition 1 the equation simplifies exactly to the one of definition 2.

Lemma 2 (Surjectivity). *The mapping ϕ is surjective. Let π be a policy on M. Then $\pi = \phi(\pi')$ with π' defined the following way (ω is omitted):*

$$\forall k \in \llbracket 1, n \rrbracket, \forall j \in \llbracket 1, m \rrbracket \setminus \varepsilon_k, \pi'_k(\dagger|\varepsilon_k) = \frac{1}{N_{\varepsilon_k}} \pi_k(\varepsilon_k) \frac{\pi_k(A)|\varepsilon_k|!}{|A| \dots (|A| - |\varepsilon_k|)}$$

$$\text{and } \pi'_k(j|\varepsilon_k) = \frac{1}{N_{\varepsilon_k}} \sum_{\substack{A \subset \llbracket 1, m \rrbracket \\ \varepsilon \subset A \\ j \in A}} \text{ with } N_{\varepsilon_k} = \sum_{\substack{A \subset \llbracket 1, m \rrbracket \\ \varepsilon_k \subset A}} \frac{\pi_k(A)|\varepsilon_k|!}{|A| \dots (|A| - |\varepsilon_k| + 1)} \quad (2)$$

Proof (sketch). First, we verify that $\sum_{j\in[\![1,m]\!]\setminus\varepsilon_k}\pi'_k(j|\varepsilon_k) = 1-\pi'_k(\dagger|\varepsilon_k)$. Then we show that $\pi_k = \phi(\pi'_k)$. Let $\varepsilon = (i_1,\dots,i_p)$, and $(j_1,\dots,j_p)$ an arbitrary permutation of ε. Let $\varepsilon_l = (j_1,\dots,j_l)$. Then, for all $l \in [\![0,p-1]\!], \pi'_k(j_{l+1}|\varepsilon_l) = \frac{N_{\varepsilon_{l+1}}}{N_{\varepsilon_l}(l+1)}$ with $\varepsilon_0 = \varnothing$. The product then simplifies to $\pi'_k(j_1)\pi'_k(j_2|j_1)\dots\pi'_k(\dagger|\{j_1,\dots,j_p\}) = \dfrac{\pi_k(\varepsilon_k)}{p!}$. Summing among all $p!$ permutations of $[\![1,p]\!]$, we verify that $\pi_k = \phi(\pi'_k)$.

Theorem 1. *Let π'_* be an optimal policy in M', then $\pi_* = \phi(\pi'_*)$ is an optimal policy in M.*

Proof. This follows directly from the surjectivity and value equivalence lemmas.

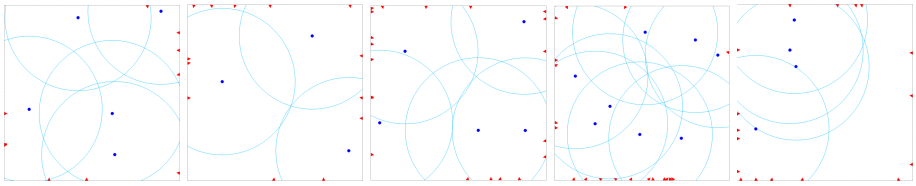
Actor-Critic methods. To find a policy that maximizes the expected average reward, we used Proximal Policy Optimization (PPO) [13], a variant of the actor-critic algorithm. Although the algorithm is only proved for MDPs, the use of a centralized critic has proven to be efficient in simple partially observable multi-agent settings [8]. PPO relies on an actor that plays episodes according to a parametrized policy and a critic that learns the state value function. After each batch of played episodes, the parameters of the two networks are updated according to the surrogate loss as in [13].

Neural network architecture. The critic neural network architecture is a standard multi-layer perceptron. Regarding the actor, the first layer consists of n_f neurons : an input tensor of size (m, n_f) is passed to the network instead of a first layer of $n_f \cdot m$ neurons. The network consists of a **feature extractor** of two layers reducing the number of features from 23 to 6 and a **feature aggregator** consisting of two linear models T and O, that represent respectively the contribution of the target itself, and the interest of the other targets. Intuitively, training at individual target level allows better feature extraction and generalization than a dense, fully connected architecture. Moreover, it allows to ensure full symmetry of the weights. However, this comes at the cost of expressiveness, as we use a special form of architecture for our actor. Let f be the extracted feature matrix : f_i is the extracted features for target i. We compute the score w_i of target i as $w_i = T(f_i) + \frac{1}{m-1}\sum_{j=1,i\neq j}^{m} O(f_j)$. The process is converted to a probability using a softmax activation function and can be represented as:

3@23 feat. → feature extractor → 3@6 feat. → feature aggregator → 3 scores

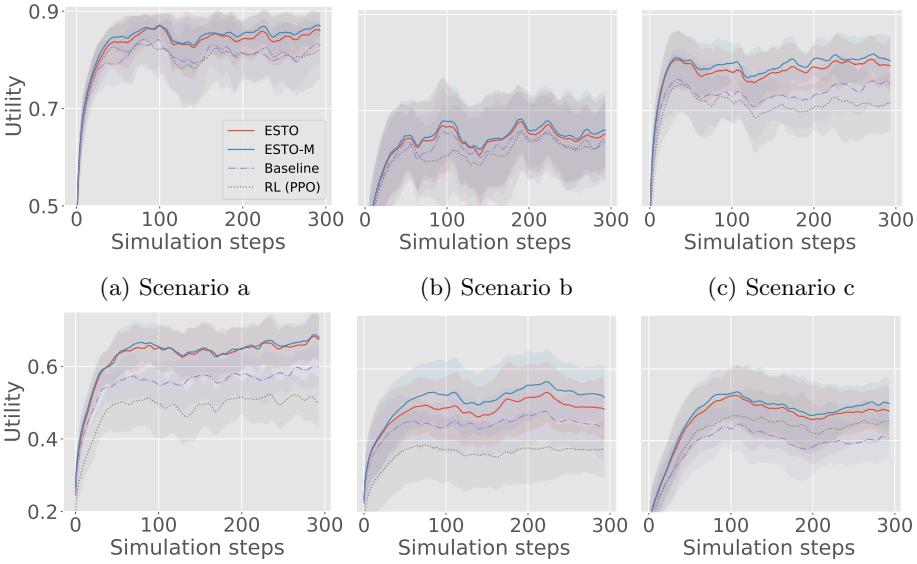
5 Evaluation

The evaluation is performed on a multi-agent simulator built with the mesa framework [9]. It consists of an environment of fixed size without obstacles with two kinds of agents: **the radars** are implemented according to the model presented in section 3. In order to simplify the model, we make the following assumptions: the radars rotate at the rate of 1 round/step; **the targets** have a



(a) Non saturated well positioned radars (b) Few saturated well positioned radars (c) Several saturated well positioned radars (d) Many saturated well positioned radars (e) Saturated ill-positioned radars

Fig. 1: The 5 validation scenarios (radars & FOV in blue, targets in red)

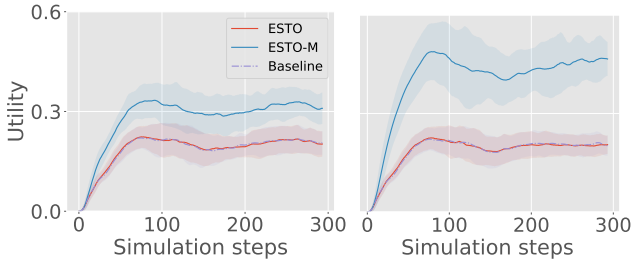


(a) Scenario a (b) Scenario b (c) Scenario c

(d) Scenario d (e) Scenario e (f) RL training scenario

Fig. 2: Utility of the radars over 300 steps with 16 random seeds

fixed speed and direction. They can turn by up to 15° and are replaced when they leave the simulation space. The simulator uses a random scheduler *i.e.* the agents act in a random order. The information they use may therefore be outdated, which allows to check the system resilience when the agents don't have up-to-date information. The ESTO approach is optimized on a scenario with 3 fixed radars and 20 targets with random trajectories over 10 runs to enhance generalization. The RL agent trains on the same settings, but over one run only due to time constraints. Our approaches are compared to a simple "baseline" approach (the radars greedily select the closest targets) on the 5 scenarios provided in fig. 1, representing interesting configurations of the radar network.



(a) 2 stacked radars (b) 8 stacked radars

Fig. 3: Performance for stacked radars scenarios

As fig. 2 shows, both ESTO and ESTO-M significantly outperform the baseline on all scenarios. The performance gain seems to be correlated with the overlap of the agents field of view (FOV). When the FOV overlap is minimal, there is less need for cooperation between agents and the baseline is close to being optimal. Conversely, when the overlap is maximal, cooperation is needed to achieve a good performance. Indeed, when radars are stacked (fig. 3), ESTO-M performs significantly better than the baseline, even more so as more radars are stacked unlike ESTO. This indicates that the distance to the closest radar feature plays an important role in ESTO's collaboration. This is confirmed by the fact that we do not observe a significant difference between ESTO and ESTO-M in scenarios (a) to (e) when ESTO can use the feature. The RL approach relies on the reformulation of the problem (definition 1). It outperforms the baseline on the training scenario but seems to have poor generalization.

6 Conclusion

In this paper, we presented two novel reward-based learning algorithms for multi-radar multi-target allocation based on centralized learning and decentralized execution. The first one, ESTO, relies on CMA-ES to optimize a linear preference function that is used to order targets for a greedy selection. The second is an actor-critic based reinforcement learning approach relying on a specific Dec-POMDP formalization. While ESTO significantly outperforms our greedy baseline by learning cooperative behaviors, the RL approach still lacks generality to do so systematically. Training it longer on more diverse scenarios (target trajectory, radar positions, number of steps) may help to prevent overfitting. Moreover, future improvements may include: The development of a neural version of ESTO that would rely on a large scale CMA-ES implementation [17] to handle the increase in the size of the parameter space. Another improvement would be the development of a more realistic radar simulation taking into account *e.g.* changes in SNR and rotation speed, and include obstacles and targets of different classes and priorities. More importantly, other than simply tuning our models for better numerical performance, we would like to interface them

with symbolic AI methods allowing them to leverage expert domain knowledge and opening the way for explainable AI (XAI) developments.

References

1. Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., Mordatch, I.: Emergent tool use from multi-agent autocurricula (2019)
2. Dai, J., Yan, J., Zhou, S., Wang, P., Jiu, B., Liu, H.: Sensor selection for multi-target tracking in phased array radar network under hostile environment. In: 2020 IEEE Radar Conference (RadarConf20) (2020)
3. Foerster, J.N., Assael, Y.M., de Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 2137–2145 (2016)
4. Geng, Z.: Evolution of netted radar systems. IEEE Access **8**, 124961–124977 (2020)
5. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation **9**(2), 159–195 (2001)
6. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning. Auton. Agents Multi Agent Syst. **33**(6), 750–797 (2019)
7. Jiang, H., Li, S., Lin, C., Wang, C., Zhong, K., He, G., Zhang, Q., Zhao, Y., Liu, J.: Research on distributed target assignment based on dynamic allocation auction algorithm. In: Journal of Physics: Conference Series. vol. 1419, p. 012001 (2019)
8. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments (2017)
9. Masad, D., Kazil, J.: Mesa: an agent-based modeling framework. In: 14th PYTHON in Science Conference. pp. 53–60 (2015)
10. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning (2016)
11. Oliehoek, F., Amato, C.: A Concise Introduction to Decentralized POMDPs. Springer (01 2016)
12. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning (2017)
13. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017)
14. Shi, C., Ding, L., Qiu, W., Wang, F., Zhou, J.: Joint optimization of target assignment and resource allocation for multi-target tracking in phased array radar network. In: 2020 IEEE Radar Conference (RadarConf20) (2020)
15. Shi, C., Wang, F., Sellathurai, M., Zhou, J.: Non-cooperative game theoretic power allocation strategy for distributed multiple-radar architecture in a spectrum sharing environment. IEEE Access **6**, 17787–17800 (2018)
16. Srinivasan, S., Lanctot, M., Zambaldi, V.F., Pérolat, J., Tuyls, K., Munos, R., Bowling, M.: Actor-critic policy optimization in partially observable multiagent environments. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada. pp. 3426–3439 (2018)
17. Varelas, K., Auger, A., Brockhoff, D., Hansen, N., ElHara, O.A., Semet, Y., Kassab, R., Barbaresco, F.: A comparative study of large-scale variants of cma-es. In: International Conference on Parallel Problem Solving from Nature. pp. 3–15 (2018)

Detection of target classification outliers for deep neural networks trained from IR synthetic data

Antoine d'Acremont[1,2], Guillaume Quin[1], Alexandre Baussard[2], and Ronan Fablet[3]

[1] MBDA France, 92350 Le Plessis-Robinson, France
[2] ENSTA-Bretagne, UMR 6285 labSTICC, 29806 Brest, France
[3] Institut Mines-Télécom, UMR 6285 labSTICC, 29238 Brest, France;

Abstract. Target classification in a military context with infra-red images is a challenging task for state-of-the-art deep learning schemes. Moreover, due to real data scarcity, one may have to train the considered frameworks from synthetic data. Besides, in operational context, the ability to detect outliers, which may comprise disturbed inputs or new classes, is critical. In this contribution, we address anomaly detection in military target classification using deep learning models trained from synthetic data. We propose a novel scheme based on a cascade of Local Outlier Factor (LOF) detectors. From inference experiments on real data, we show that the proposed scheme clearly outperforms a single LOF detector applied to the last layer of the neural network, which is among the state-of-the-art approaches.

Keywords: Anomaly Detection · Deep-learning · Infra-red imaging · Domain transfer.

1 Introduction

Convolutionnal Neural Networks (CNN) have shown great performance in a large variety of computer vision applications. Their good performance on classification tasks makes them a suitable choice for tackling automatic target recognition problems in a military context. However, their performance is conditioned by the size and the quality of the training dataset [12, 1]. For military applications, collecting a large annotated dataset may be more difficult than for civilian tasks. This can be especially challenging for training a CNN on radar [11], sonar [3] or infra-red images [10].

Moreover, deep neural networks are sensitive to the inputs quality which could induce errors. This is particularly true for military applications where the targets are usually non cooperative and may be partially masked or using camouflage. A classification error could have dangerous consequences on a battlefield.

In this work, we wish to address the problem of building an anomaly detection module using only synthetic data as training set. These anomalies, sometimes called outliers, includes images from unknown classes or background that could

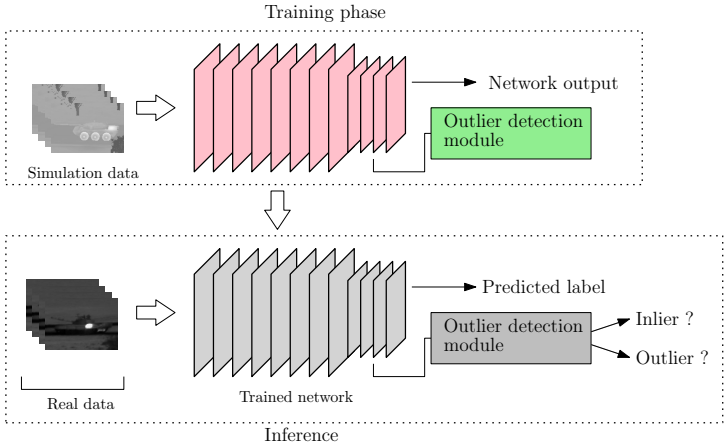


Fig. 1. Detection of anomalies or outliers from synthetic data.

be presented to a convolutional neural network during inference. This module is expected to be used on real images as shown in figure 1.

In this contribution we focus on detecting anomalies from a CNN used for the classification of military vehicles in infra-red imagery. We first train a compact neural net, the cfCNN [4], and an anomaly detector using the Local Outlier Factor (LOF) algorithm [2] on synthetic data to evaluate the performance of this approach and show its limits on real images. We then present an improved detector using a cascade of LOF-based anomaly detector to better identify anomalies when trained on synthetic data, compared to the original use of the LOF. We show that using a cascade of LOF detectors allows us to train both a neural network and an anomaly detection module only on synthetic data. This paper is divided into 4 parts. First, in section 2, we present an overview of existing anomaly detection techniques and the simple LOF detector with the cfCNN architecture. Then in section 3 we present our contribution, the cascade of LOF or CLOF. In section 4, we compare its performance to the single LOF detector on real images after having been trained on synthetic data. Finally, we discuss our results and possible improvements in section 5.

2 State of the art

2.1 Compact backbone for target classification

In previous works we introduced and used a new CNN, the cfCNN [4]. This network proved to be efficient for the classification of targets in real infra-red images, when trained from synthetic data.

The cfCNN model is presented in figure 2. It is an all convolutionnal neural network with six layers and a global average pooling layer (GAP) at its end. Every unit in the network, except for the final softmax layer, uses the Leaky

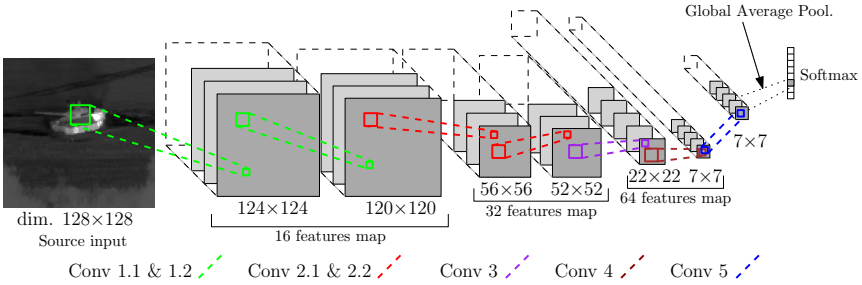


Fig. 2. Presentation of the cfCNN from [4].

ReLU non linearity as activation function. In [4] we showed that the use of a global average pooling layer increased the resilience of the cfCNN to translated or scaled inputs. However this vulnerability to disrupted inputs still exists. Being able to identify such anomalies or outliers during inference is an active field of studies.

2.2 Anomaly detection and LOF detector overview

Hendricks et al., have recently presented a detection framework for anomaly detection[6]. This specific framework requires a supervised training which can be difficult for our application due to the large variety of perturbations we may encounter during inference. Moreover, their method uses the softmax scores to help separate inliers from outliers. With specific examples such as adversarial examples, softmax score can appear artificially normal. As a result, new outlier detection methods were presented by Liang et al. [8] or Lee et al. [7] that either uses a modified softmax inputs with a temperature parameter or the outputs of the penultimate layer of the network to improve the detection of outliers. Both methods have shown excellent performance for the detection of outliers and adversarial examples on visible images.

However, their performance relies on preprocessing a modified version of the input that requires the computation of the gradient of the detector during the inference. In an effort to simplify the process of detecting outliers during inference, we proposed a novel anomaly detection module to identify classification errors [5] and tested it in combination with our cfCNN on infra-red images from the SENSIAC dataset [9]. This technique, based on the Local Outlier Factor algorithm (LOF)[2] was only tested when the training and validation data came from the same domain. The obtained performance was equivalent to the use of a more complex detection techniques such as ODIN (Out-of-DIstribution detector for Neural networks [8].

The anomaly detector based on the LOF algorithm proposed in [5], shown in figure 3, requires the computation of a covariance matrix $\mathbf{K}$ from the penultimate layer of the network f_{n-1}, n being the total number of layers and f_n the softmax layer. This covariance matrix is needed as we are using the Mahalanobis distance

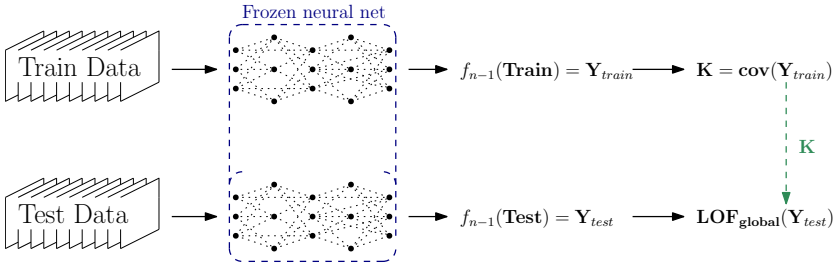


Fig. 3. Description of the usage of the LOF for anomaly detection in [5].

as the metric for the LOF algorithm. The output of LOF_{global} is a binary classification between outliers and inliers. By default, if $LOF_{global}(f_{n-1}(x)) > 1 + \epsilon$ then x is classified as an outlier and if $LOF_{global}(f_{n-1}(x)) \leq 1 + \epsilon$, x is classified as an outlier. ϵ is a user-defined parameter to tweak the sensitivity of the algorithm.

3 Proposed cascade anomaly detector

To improve on the performance of our single LOF detector linked to the penultimate layer of a cfCNN, we propose to combine the information from multiple layers within the cfCNN. A dedicated LOF detector is assigned to each layer. This setup will be called Cascade-LOF or CLOF. The complete setup is presented in figure 4.

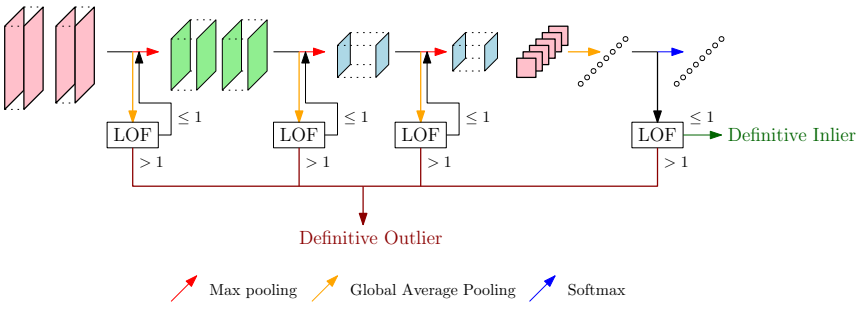


Fig. 4. Presentation of the CLOF anomaly detection module.

As shown in figure 4, since every layer in the cfCNN is a convolutionnal layer, we use a global average pooling layer on each intermediate output to reduce the size of the vector used as an input for the LOF of each layer. Without this step, the size of the input vector for the LOF would drastically reduce the speed of the detector and require too much additional computing power to be a viable solution for anomaly detection. A sample is labelled as an outlier if only one of

the detectors identifies it as an outlier. As a result, a sample has to pass through all the successive detectors without being detected to be labelled as an inlier. By doing so, we wish to monitor the change of internal representation of the data distribution at every layers.

In [5], we limited our study to a situation where our training and testing dataset came from the same source *i.e.* SENSIAC [9]. In the next section we evaluate our anomaly detection modules based on the LOF, trained on synthetic data and tested on real data. We will be using simulated and real infra-red images from a MBDA dataset described in section 4.1.

4 Comparison of the LOF and CLOF

4.1 Datasets

For our experiments we use in total six different datasets, described in table 1.

Category	Name	Type	Size(images)	Usage
Inliers	TS	Simulated Inliers	18340	Training
	VS	Simulated Inliers	4096	Validation
	R	Real Inliers	43810	Testing
Outliers	OR	Real Outliers	1862	Testing
	BR	Real Backgrounds	4096	Testing
	C	Simulated Canonical	4096	Testing

Table 1. Description of datasets used in experiments.

The datasets TS and VS are used to respectively train and validate the cfCNN, the LOF and the CLOF. The datasets R, OR, C and BR contain inliers and outliers and will be used to measure the LOF and CLOF performances. The dataset C is comprised of what we describe as canonical examples. Each image is made of geometric shapes with random grey values, shapes and positions.

4.2 Experimental setup

As explained in section 4.1, we use the cfCNN trained on the complete TS dataset with 18340 images. Once fully trained, we save the outputs of each layer of the network and use them to prepare both a LOF based detector linked to f_{n-1} and a CLOF. Additionally, we define:

- True positives, or TP, every outliers correctly detected.
- False positives, or FP, every inlier falsely detected.
- True negatives, or TN, each inlier not detected.
- False negatives, or FN, each outlier not detected.

To evaluate the performance we will use three metrics: the precision $PRE = \frac{TP}{TP+FP}$, the false positive rate $FPR = \frac{FP}{FP+TN}$, and the F-score $F1 = \frac{2 \times PRE \times REC}{PRE+REC}$ with $REC = \frac{TP}{TP+FN}$.

To get a representative performance of each detector, we repeat the training and testing cycles several times. Consequently, we use fixed parameters for the LOF and the CLOF. We will not use ROC curves to show the performances of each detector but average performances to avoid any bias by changing the parameters.

4.3 Results

We compiled the results in table 2 with details for each dataset in table 3.

Detector	PRE	FPR	F1
	Min/Mean/Max	Min/Mean/Max	Min/Mean/Max
LOF	0.14/0.24/0.48	0.52/0.57/0.61	0.23/0.31/0.51
CLOF	0.65/0.66/0.70	0.19/0.25/0.28	0.67/0.68/0.69

Table 2. Results on MBDA dataset.

As visible in table 2 the overall detection performance of the LOF is very low and with a strong variance. This variance can partly be explained as we are comparing the detection results on multiple training iterations of the cfCNN and the LOF detector. Since each network has its own initialisation and uses drop-out we expect that they will have different weights after training. However, while the false positive rate varies within less than 0.10, both the PRE and F1 scores for the LOF cover a wider range of values between their respective maximum and minimum values. This shows the poor performance of the detector.

When looking at the CLOF scores, there is an improvement in the overall performance. This is highlighted by the FPR which has significantly decreased compared to the single LOF to 25.1% on average.

Dataset	Detector	Detected as inlier Min/Mean/Max	Detected as outliers Min/Mean/Max
Inliers (real and simulation)			
VS	LOF	99.1/99.4/99.7%	0.3/0.5/0.9%
	CLOF	99.4/99.5/99.6%	0.4/0.5/0.6%
R	LOF	17.1/43.2/86.3%	13.7/56.8/82.9%
	CLOF	60.9/64.8/72.6%	27.4/35.2/39.1%
Outliers			
BR	LOF	64.8/74.5/84%	16/25.5/35.2%
	CLOF	35.2/42.9/48.8%	51.2/57.1/64.8%
OR	LOF	28.4/37.8/48.9%	51.1/62.2/71.6%
	CLOF	30.8/34.8/59.9%	40.1/65.2/69.2%
C	LOF	11.5/14.8/18.6%	81.4/85.2/88.5%
	CLOF	9.4/10.9/12.3%	87.7/89.1/90.6%

Table 3. Per-anomaly classification results for each detector.

For a more comprehensive understanding of how each detector performs, one must look at the results from table 3. Here we can see that both the LOF and

CLOF manage to correctly label on the samples from the VS dataset. However, the LOF detector is unable to generalize to real examples as highlighted by the performance on the R and OR dataset with on average only 43.225% of R data being labelled as inliers. Outliers from the OR dataset are also more frequently and incorrectly labelled as inliers with 37.8%. Moreover, BR samples are also not detected by the LOF with up to 84% of them being classified as inliers.

In table 3, we can see that the use of the CLOF has improved the detection rate on OR while decreasing the number of false positives on R. Now on average less than 40% of real images are detected as outlier, from up to 82.9% for the LOF. The performance on BR has increased significantly with now 42.9% false positives only on average. However, it remains far from the performance obtained on cases without domain transfer. Further improvements are needed for this systems to reach parity with results from [5] for example.

5 Conclusion

In this paper we have introduced a cascade detector to identify outliers during inference. We showed that we have improved the performance of outlier detection compared to a single LOF detector linked to the penultimate layer of a neural network. However, our results showed that even with those improvements, there still a high probability of misclassification of outliers especially on input images with only backgrounds.

More analysis is needed to find the best combination of parameters for the detectors in the cascade. One other possible suggestion would be to replace the GAP operation at each stage with different metrics that may better represent the information contained in each convolution layer.

References

[1] A. Nguyen, J. Yosinski, and J. Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 427–436. ISBN: 1063-6919. DOI: 10.1109/CVPR.2015.7298640.

[2] Markus M. Breunig et al. "LOF: Identifying Density-based Local Outliers". In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388. URL: http://doi.acm.org/10.1145/335191.335388.

[3] D. P. Williams. "Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. Dec. 2016, pp. 2497–2502. DOI: 10.1109/ICPR.2016.7900011.

[4] Antoine d'Acremont et al. "CNN-Based Target Recognition and Identification for Infrared Imaging in Defense Systems". In: *Sensors* 19.9 (2019). ISSN: 1424-8220. DOI: 10.3390/s19092040. URL: https://www.mdpi.com/1424-8220/19/9/2040.

[5] Antoine d'Acremont et al. "Detecting unseen targets during inference in infrared imaging". In: *Artificial Intelligence and Machine Learning in Defense Applications*. Ed. by Judith Dijk. Vol. 11169. International Society for Optics and Photonics. SPIE, 2019, pp. 66 –76. DOI: 10.1117/12.2533178. URL: https://doi.org/10.1117/12.2533178.

[6] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017. URL: https://openreview.net/forum?id=Hkg4TI9xl.

[7] Kimin Lee et al. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 7167–7177. URL: http://papers.nips.cc/paper/7947-a-simple-unified-framework-for-detecting-out-of-distribution-samples-and-adversarial-attacks.pdf.

[8] Shiyu Liang, Yixuan Li, and R. Srikant. "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks". In: *arXiv e-prints* (June 2017), arXiv:1706.02690.

[9] *Military Sensing Information Analysis Center (SENSIAC): Dataset for Automatic Target Recognition in Infrared Imagery*. 2008. URL: www.dsiac.org/services/store/atr-algorithm-development-imagedatabase.

[10] Iain Rodger, Barry Connor, and Neil M. Robertson. "Classifying objects in LWIR imagery via CNNs". In: vol. 9987. Oct. 2016, 99870H. ISBN: 0277-786X. DOI: 10.1117/12.2241858. URL: https://ui.adsabs.harvard.edu/abs/2016SPIE.9987E..0HR.

[11] S. Chen et al. "Target Classification Using the Deep Convolutional Networks for SAR Images". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.8 (Aug. 2016), pp. 4806–4817. ISSN: 1558-0644. DOI: 10.1109/TGRS.2016.2551720.

[12] S. Moosavi-Dezfooli et al. "Universal Adversarial Perturbations". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 86–94. ISBN: 1063-6919. DOI: 10.1109/CVPR.2017.17.

Siamese Network on I/Q Signal for RF Fingerprinting*

Louis Morge-Rollet[1], Frédéric Le Roy[1], Denis Le Jeune[1], and Roland Gautier[2][0000−0003−3570−1061]

[1] ENSTA Bretagne, Lab-STICC, CNRS, UMR 6285, F-29200, Brest, France
{firstname.last.name}@ensta-bretagne.org
[2] Univ Brest, Lab-STICC, CNRS, UMR 6285, F-29200, Brest, France
{firstname.last.name}@univ-brest.fr

Abstract. RF Fingerprinting techniques aim to authenticate a wireless emitter by the imperfections due to these components. It can be useful for authentication and network management for the future IoT networks. Various methods has been proposed using hand-crafted features and classic machine learning but nowadays many researchers try to apply deep learning architectures for RF Fingeprinting. Our contribution is based on Siamese Network, a deep learning architecture widely used by the face recognition community. We use the deep learning architectures proposed by the RF Fingeprinring community which processes the I/Q (In-phase and Quadrature) signal and the siamese network learning paradigms developed for the facial recognition to propose siamese architectures for RF Fingerprinting. One of the main advantage of the siamese network is the possibility to use one-shot learning and its ability to require a few data for the final implementation of the network. In this paper, we explain our implementation, our results and discuss about the potential benefits of our approach for final implementation in a wireless network.

Keywords: RF Fingerprinting, Siamese Network, Deep Learning

1 Introduction and state of the art

Cybersecurity is a major concern of our epoch. Devices become more and more connected and cyberattacks are increasingly frequent and massive. The wireless technologies, such as WiFi, Bluetooth and Mobile networks are massively used. With the incomming of new technologies such as autonomous vehicules, smart grid, smart cities among others, the demand for connectivity will explose and require the use of new protocols such as 5G and IoT Networks (Zigbee, LoRa, ...). Many IoT protocols are based on low energy constraints but these technologies need to be secured. However the security measures such as cryptography

* This work was funded by ENSTA Bretagne of Brest and also supported by the IBNM (Brest Institute of Computer Science and Mathematics) CyberIoT Chair of Excellence of the University of Brest.

are difficult to implement for IoT protocols due to the complexity of keys management and energy consumption of standard cryptography [1].
The RF Fingerprinting is a part of *physical layer securities* [2] aiming to protect communications based on physical-layer properties. This technique consist in authenticate a wireless emitter using the specific impairments of these components. The manufacturing process has some uncertainties and two devices that seems to be similar have their own physical impairments. The impairments such as I/Q offset, I/Q imbalance, clock offset among others can be used to authenticate an emitter. The RF Fingerprinting is considered as a *Non-Cryptographic authentication* technique [3], however there is a debate in the community to know if it can replace the cryptographic authentication protocol (RSA, ...) or be used as a second factor for authentication [4]. RF Fingerprinting can also be used for intrusion detection [5] or to secure network layer against attacks [6]. Our approach based on deep learning architecture and siamese network will focus on authentication but it is possible to generalize it to other applications.

1.1 Deep Learning architecture

The machine learning (ML) is a part of artificial intelligence, based on algorithms (SVM, neural network, ...) able to learn how to solve a problem from data. Neural networks are bio-inspired mathematical models, they are composed of stacked layers (i.e parallel set) of basic unit called *neuron*, generally many layers (called *hidden layers*) are stacked to mimic the way that brain processes informations. The deep learning (DL) generally refers to neural network with two or more hidden layers. Many architectures, inspired from brain specific parts, have appeared over time like Convolutive Neural Networks (CNN) or Recurrent Neural Networks (RNN) to solve specific problem like, respecively, image recognition or time-series prediction.
Many methods has been described in the literature for RF Fingerprinting. Some methods focus on the transcient aspect of the signal [7], others on the steady-state aspects (also called *Modulation-based*) like [8] or the both aspects [9]. With the increasing popularity of deep learning, RF Fingerprinting community begins to use deep learning architecture on raw I/Q signals, specifically the CNN [1], [10], [11] and [12]. Futhermore, the DARPA has lauched in 2017 the program RFMLS[3] (Radio Frequency Machine Learning Systems) which aims to develop the use of machine learning for radio frequency. One argument of the RFMLS project is to develop the use of deep learning architecture to replace classic machine learning techniques based on expert hand-crafted features which are dependent on a priori asumptions [13].

1.2 Siamese network

A siamese network consists of two neural networks which have identical weights and their inputs are projected on a latent space where similarity measure are

[3] https://www.darpa.mil/attachments/RFMLSIndustryDaypublicreleaseapproved.pdf

applied (L_2 distance, ...) to know the similarity between the two inputs. The first application of siamese networks was for signature recognition [14] but that type of architecture is widely used by the facial recognition community [15], [16] and [17]. Other applications using siamese networks have also appeared like dimensionality reduction [18] or voice casting [19] among others. The siamese networks can be really useful in several cases: when a few data are available for the final implementation, when there is a lot of classes and for detection intrusion. G. Koch et al. [20] show the possibility of using siamese network for *one-shot learning* (i.e when there is only one learning example per class) for hand drawn characters recognition. Futhermore, Langford et al. [21] show the possibility of using siamese network on compressed spectrogram for specific emitter identification (a task similar to RF Fingerprinting), the authors also show the performance gains of siamese network compared to classic CNN for low SNR.
Several learning paradigms has been proposed to train a siamese network. The first approach, developed by LeCun et al. in [14], was based on cosine similarity. The most popular approach is based on *contrastive loss* [15], [18] and [19], which uses a specific loss which constraints the latent representation to respect some properties (see further explanation in section 2.2). G. Koch et al. proposed in [20] a siamese network learning paradigm as a *logistic regression* problem using *weighted* L_1 *norm* (see further explanation in section 2.2) which seems yielding better results than previous methods [21]. The previous approaches were considered as *end-to-end problems* but other approaches differ from it like DeepFace [16] which consists to use *transfer learning* (i.e transfer some knowledge learned from similar task to a new one) or the *triplet loss* [17] which consists to a specific *end-to-end problem* using three inputs instead of two.

1.3 Proposed approach

Our approach consist in using the deep learning architecture coming from the RF Fingerprinting community [1], [10] and [11] directly on I/Q signals collected over real-world measurements and the siamese network paradigm for RF Fingerprinting. This paper is composed as follow: Proposed method (section II), Experimental data analysis and results (section III), Potential benefits and further work (section IV) and Conclusion (section V).

2 Proposed Method

2.1 Dataset

Original dataset: The dataset [4] on which this study is based come from real-world measurements and was used by [11] and [1] to explore deep learning architectures for RF Fingerprinting. It was composed of 2 types of datasets: over-the-air and over-the-cable configurations. These datasets are composed of 16 identical USRP X310 SDR (Software-Defined Radio) platforms. Each emitter is

[4] http://www.genesys-lab.org/oracle (last visit the 26/08/2020)

recorded twice (*run 1* and *run 2*) for a duration of 4 seconds with a sample rate of 5 Ms/s which corresponds to 20 millions I/Q samples. The process is repeated for different distances in the range 2ft to 62ft with an interval of 6ft. The SDR receiving platform is always the same for each experiment: USRP B210. For our experimentation, the over-the-air configuration is chosen, which is considered more realistic. We use the 2ft recordings, which can be considered as a LOS path and with a really high SNR ($>$ 45 dB). Only the first 10.24 ms (4000*128 samples) of *run 2* for each emitter was used to create the database, which is considered to be of better quality than the *run 1*. Contrarily to [10] and [11] we use non-overlapping windows to extract the examples from the recording, which allows a better independance (in term of sampling) between all the examples. Our dataset, called *baseline dataset*, is composed of 16 classes of emitters with 4000 examples per class, which is considered enough for CNN classification.

Siamese dataset preparation: From the *baseline dataset*, a second dataset has been created to train the siamese network: the *siamese dataset*. The strategy used to create this dataset is inspired with the previous works on siamese network [14], [15], [16], [18], [19] and [20]. It is composed of a equal number of positive pairs (i.e two inputs from the same emitter) and negative pairs (i.e two inputs from different emitters). The process to create the dataset is the following; for a specific input of the dataset we choose N (here 5) inputs with the same class (without the corresponding input) using a sampling without replacement to create the positive pairs and we choose N inputs with different class using a sampling without replacement to create negative pairs. This process is repeated for each input of the dataset to create the *siamese dataset*. Concerning the train/test split of the dataset, the scikit-learn train_test_split function is used on *baseline dataset* and the process described above is applied separately on the training set and testing set.

2.2 Architecture and learning paradigms

The architecture used for this work (see table 1(a)) is inspired by an architecture from [11]. The network processes the I/Q signal as an 2x128 image (i.e 2 for I and Q and 128 for sample number) with one channel, corresponding to an input size of 2x128x1. In our experiment, we compare several learning paradigms. The first learning paradigm comes from [20] and consists as a *logistic regression* problem (i.e the output predict the propability that two inputs are similar) using a *weighted L_1 norm*. Indeed, an element-wise absolute difference is applied to the latent representations followed by a logistic regression (using binary cross-entropy loss): $\hat{y}(x_1, x_2) = \sigma(\sum_i \alpha_i |G_W(x_1)[i] - G_W(x_2)[i]| + \alpha_0)$ where $G_W(x_i)$ represent the latent representation of the input x_i.
The second learning paradigm called the *contrastive loss*, is based on the work of LeCun et al. [18] and the third called *contrastive transfer*, is based on the work of [16] using *transfer learning*, but instead of using a *weighted L_1 norm* like

[20], we used a *contrastive loss*. The *contrastive loss* proposed in [18], to train a *mapping function* (G_W) for a dimensionnality reduction purpose, consists to constrain the latent representations to respect some properties. Especially, as mentioned by [15] and [19], similar points (x_1 and x_2) need to be near from each others and the distance between disimilar points (x_1 and $x_1^{'}$) need to be greater than a specific constant called margin m (here 1). This constraint can be express like: $E(x_1, x_2) + m < E(x_1, x_1^{'})$ where $E(x_i, x_j) = \|G_W(x_i) - G_W(x_j)\|_2$ is the distance (using L_2 norm) between the projection of (x_i, x_j) in the latent space. The associated loss function is the following:

$$L(I_1, I_2, Y) = Y * \|G_W(I_1) - G_W(I_2)\|_2^2 + (1 - Y) * max(0, m - \|G_W(I_1) - G_W(I_2)\|_2)^2 \quad (1)$$

Where:

- I_i is an input and $G_W(I_i)$ his corresponding latent representation
- Y indicated if the pair are similar ($Y = 1$) or dissimilar ($Y = 0$)

The first two learning paradigms are *end-to-end problems* unlike the third one which is based on *transfer learning*. For this approach we have proceded as following: we train the network proposed in [11] for a K-class classification problem (considered as our *high-level caracteristics* extractor), we remove the two last layers (i.e the softmax and the last dense layer), add an other dense layer of 128 neurons and train only the last layer (the parameters of the other layers are fixed) using the previously introduced *contrastive loss*.

We used Adam optimizer on 32 epochs with batch size of 128. We used regularization to avoid over-fitting like l_2 regularization on each layer and a dropout of 50% at the first dense layer. The hyperparameters (see table 1(b)) have been found using grid search and hold-out validation for the learning rate μ, the l_2 regularization parameter l and the number of neurons D of the last dense layer.

(a) Neural network architecture

Layers	Characteristics
Input	(2, 128, 1)
Conv2D	50 filters (1x7) + ReLu
Conv2D	50 filters (2x7) + ReLu
Flatten	
Dense	256 neurons + ReLu
Dense	D neurons + ReLu

(b) Hyperparameters

	μ	l	D
Logistic regression	0.0001	0.00001	128
Contrastive loss	0.001	0.0001	128
Contrastive transfer	0.001	0.0001	128

Table 1: Neural network parameters

3 Experimental data analysis and results

We train our model using Keras framework on the school cluster (Intel Skylable Gold 6132). The metric used to evaluate the first learning paradigm is the accuracy. For the two others, we define a specific metric, which consists to compare the distance of the latent reprensentations to the half of the margin. If the distance is lower than the half of the margin the inputs are considered as a similar pair (Y=1), otherwise the inputs are considered as dissimilar pair (Y=0).

3.1 Experiments

The performances of the several learning paradigms are shown in the table 2. The *logistic regression* have better performances than the others learning paradigms, which confirms the conclusion of [21]. One can say that is unsual that training loss is greater than testing loss. But this phenomena, discuted by A. Géron in this post[5], can be explained in our case by the regularization applied during the training (i.e dropout and l_2 regularization).

Learning paradigm	Train accuracy	Test accuracy
Logistic regression	0.9909	0.9952
Contrastive loss	0.9669	0.9744
Contrastive transfer	0.9195	0.933

Table 2: Performances of learning paradigms

3.2 The dataset problem

The performances obtained are good although slightly below than [21] (reaching 99.79%). There may be several explanations to this lack of performances. First of all, the dataset used is not really suitable for a siamese network problem (contrary to Omniglot, [20]). Indeed, a classic siamese dataset consists of many classes with few examples per class which possed high inter-class variability. On the contrary, the *baseline dataset* is composed of few classes and many examples per class which is usually more adequate for K-class classification problems. Maybe the variability of dissimilar pairs are not large enough to train a good network. On the second hand, the impairments present in our *siamese dataset* is less controled than in the dataset used by [21] which seems coming from simulation, including 4 emitters and having a single impairment (frequency offset). Conversely, our dataset is based on real-world measurements on 16 differents emitters (USRP X310) with various impairments. Futhermore, our approach does not require pre-processing like time-frequency transform (used in [21]) and directly work on I/Q signals.

4 Potential benefits and further work

The main problem of deep learning architectures and more generally machine learning algorithms is what we defined as *scalability*, i.e a model needs to be retrained for a new group of unknow emitters which is not really scalable for IoT devices with computational and energy constraints. The majority of the work on RF Fingerprinting considered the problem as K-class problem with an relatively large amount of data to train the algorithm (approximatively a thousand exemples per class). These works are interresting because they proposed new architectures/algorithms for RF Fingerprinting but the authors rarely take into account the scalability problems introduce by theirs approaches. An other problem is that K-class classification is also not really performant when the number of emitters of the network is too large and changing over time: some emitters

[5] https://twitter.com/aureliengeron/status/1110839223878184960

can leave the network and new ones can join it. The last point is concerning the spoofing attack: if an architecture/algorithm is trained to recognize K emitters (legitime and known) how will it behave when an illegitimate emitter try to communicate on the network.
The main interest of a siamese network algorihtm is that it doesn't learn a classifier but an "advanced" similarity metric. This allows to train the siamese network on a big database which generalize well the variability of the emitter and use it as similarity metric with a K-Nearest Neighbors (KNN) algorithm for final implementation with unknow emitters. This approach has several advantages:

- The final implementation need at least one example per emitter: one-shot learning
- The architecture doesn't need to be retrained for the final implementation
- Outlier detection can be used to detect illegitimate emitters

This type of approach is widely use for facial recognition, where an input image is compared with a list of images, to identify the corresponding person (match) or an intruder (unmatch). It consists: to store the latent representations of known emitters (one per emitter), to compute the latent representation of the new input and to compute distance on the latent space to determine if the emitter belongs to the network (using a pre-determined threshold) and if that is the case to which emitter it belong. It is quite similar to 1NN (K=1) approach but with the concept of similarity metric replacing classical metric such as L_2 distance.
To our knowledge, only Ioannidis et al. [1] has proposed a method for one-shot learning for deep learning architecture based on an other type of approach. Our future work will explore the performance of deeper architectures for siamese network, complex-valued neural networks and others learning paradigms such as triplet loss. It will also be interesting to use 1NN algorithm (or more generally KNN) to test the performance of this approach from a *one-shot learning* point of view. Furthermore, we need to explore the performances under a range of SNR and multi-path environments.

5 Conclusion

The purpose of this article was to proposed a siamese approach for RF Fingerprinting based on the raw I/Q signal. We present the architecture of the network, the learning paradigms choosen and present the results on a real-world measurement dataset. We also introduced the potential benefits of this architecture for final implementation in a IoT network and some potential research works and improvements.
One of the main advantage of this approach compared to others (such as [21]) for RF Fingerprinting is that the network doesn't require preprocessing like time-frequency transform and directly works on I/Q signals. An other advantage of siamese network is their ability to perform *one-shot learning*. Use of deeper architectures and/or complex-valued neural networks (exploiting the complex nature of the signals) can further increase the obtained performances. This type of approach can be useful for final implementation, on IoT networks or more generally radio networks, to perform authentication and to allow a better and more flexible network management.

References

1. K. Sankhe , M. Belgiovine, F. Zhou , L. Angioloni, F. Restuccia, S. D'Oro , T. Melodia, S. Ioannidis and K. Chowdhury: No Radio Left Behind: Radio Fingerprinting Through Deep Learning of Physical-Layer Hardware Impairments, IEEE Trans. on Cognitive Communications and Networking, Vol. 6, NO. 1, 2020.
2. Y. Zou, J. Zhu, X. Wang and L. Hanzo: A Survey on Wireless Security: Technical Challenges, Recent Advances, and Future Trends, Proce. of the IEEE, Vol. 104, 2016.
3. K. Zeng, K. Govindan, and P. Mohatara,: Non-cryptographic Authentication and Identification in Wireless Networks, IEEE Wireless Communications, Vol. 17, 2010.
4. Pieter Robyns, Eduard Marin, Wim Lamotte, Peter Quax, Dave Singelée, Bart Preneel: Physical-Layer Fingerprinting of LoRa devices using Supervised and Zero-Shot Learning, Wisec'17, 2017.
5. Nam Tuan Nguyen, Guanbo Zheng, Zhu Han and Rong Zheng: Device Fingerprinting to Enhance Wireless Security using Nonparametric Bayesian Method, INFOCOM, 2011.
6. S. Capkun K.B. Rasmussen. Implications of radio fingerprinting on the security of sensor networks. SecureComm, 2007.
7. O. Ureten and N. Serinken: Wireless security through RF fingerprinting, Canadian Journal of Electrical and Computer Engineering, vol. 32, no. 1, pp. 27–33, 2007.
8. V. Brik, S. Banerjee, M. Gruteser and S. Oh: Wireless Device Identification with Radiometric Signatures, MobiCom'08, 2008.
9. H. Yuan, Z. Bao, A. Hu: Power Ramped-up Preamble RF Fingerprints of Wireless Transmitters, Radioengineering, Vol. 20, n. 3, SEPT. 2011, 703–709.
10. S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury: Deep Learning Convolutional Neural Networks for Radio Identification, IEEE Communications Magazine, 2018.
11. K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis and K. Chowdhury: ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks, INFOCOMM, 2019.
12. E. Mattei, C. Dalton, A. Draganov, B. Marin, M. Tinston, G. Harrison, B. Smarrelli and M. Harlacher: Feature Learning for Enhanced Security in the Internet of Things, GlobalSIP, 2019.
13. A. Ghasemi, C. Parekh and P. Guinand: Spectrum Awareness Under Co-Channel Usage Via Deep Temporal Convolutional Networks, WinCOMM, 2019.
14. J. Bromley, I. Guyon, Y. LeCun, E. Sickinger and R. Shah: Signature Verification using a Siamese Time Delay Neural Network, NIPS-94, 1994.
15. S. Chopra, R. Hadsell and Y. LeCun: Learning a Similarity Metric Discriminatively, with Application to Face Verification, CVPR, 2005.
16. Y. Taigman, M. Yang, M. Ranzato and L. Wolf: DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR, 2014.
17. F. Schroff, D. Kalenichenko, J. Philbin: FaceNet: A Unified Embedding for Face Recognition and Clustering, CVPR, 2015.
18. R. Hadsell ,S. Chopra and Y. LeCun: Dimension Reduction by Learning an Invariant Mapping, CVPR, 2006.
19. A. Greese, M. Quillot, R. Dufour, V. Labatut and J. Bonastre: Similarity Metric Based on Siamese Neural Networks for Voice Casting, ICASSP, 2019.
20. G. Koch,R. Zemel and R. Salakhutdinov: Siamese Neural Networks for One-shot Image Recognition, 2015.
21. Z. Langford, L. Eisenbeiser and M. Vondal: Robust Signal Classification Using Siamese Networks, WiseML, 2019.

Refining Simulated SAR images with conditional GAN to train ATR Algorithms

Benjamin Camus, Eric Monteux, and Mikael Vermet

Scalian-DS, 2 Allée de Becquerel, 35700 Rennes, France
benjamin.camus@scalian.com eric.monteux@scalian.com
mikael.vermet@scalian.com

Abstract. This work aims to train an ATR classifier using simulated SAR images, to overcome the lack of real images for targets of interest. We propose to train a cGAN with pairs of real and synthetic data to learn a refining function that adds specific features observed on measured datasets to simulated data generated by the MOCEM simulator, developed by SCALIAN DS for the French MoD (DGA). We also propose a multi-refining technic that leverages the cGAN oscillations around a local optimum to perform data augmentation. Thanks to this approach, a classifier can be trained with refined synthetic data and achieve similar accuracy than a classifier trained on real data. Our approach is not limited to MOCEM, although a refining model has to be trained specifically for each simulator. Yet, this model cannot generalize to new target classes.

Keywords. Generative adversarial networks, synthetic aperture radar, automatic target recognition, simulation, data refining

1 Introduction

Automatic Target Recognition (ATR) on Synthetic Aperture Radar (SAR) images is a long-standing problem that aims to automatically classify objects imaged by a SAR [1]. It is particularly relevant in the defense sector, especially for ISTAR (Intelligence, Surveillance, Target Acquisition and Reconnaissance) applications. It is not obvious, even for a trained human operator, and there is a great hope in doing this task by emerging computing technologies. Several works have demonstrated the ability of Deep Learning algorithms to solve the ATR problem [2, 3, 4]. For instance, the reference work of Morgan reaches 92.3 % of accuracy at test time [5]. However, these algorithms are limited by the large datasets they require for training. Very few of them are publicly available (most studies use the MSTAR images [6]), and the cost of doing measurements to build a dataset from scratch remains prohibitive. As already done in the literature to classify images in the visible spectrum [7], a simulator can be used to bypass this limitation by generating a large enough synthetic training set. However, because of the specific physics of EM, SAR images are more complex to simulate than images in the visible spectrum, and subject to extreme variability regarding observation conditions (e.g. targets shapes, material diversity, environment noise, sensor parameters). As a consequence, synthetic datasets have features significantly different than real measurements. Hence, ATR models trained with simulated data make important errors at

test time when classifying real SAR measurements (Cha et al. reported an accuracy of only 19.5 % [8]). In this work, we propose to use a Conditional Generative Adversarial Network (cGAN) [9] to learn a refining function that adds some "measurement specific" features to synthetic SAR images. We show that refined synthetic data can be used instead of real measurements to train an ATR classifier and get similar accuracy at test time. Our experiments are based on the MSTAR real measurements dataset. To produce synthetic training sets that mimic these measurements, 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 [10]. We show the versatility of our approach by applying it to the synthetic and real data of the SAMPLE challenge [11] with similar results. Section 2 details the MSTAR dataset, and its reproduction with MOCEM. Section 3 shows the impact of synthetic data injection in the training set of an ATR classifier. Section 4 presents the training of the refiner using a cGAN. Section 5 analyzes the impact of refining data on ATR accuracy. In Section 6, we show the versatility and limits of our approach. Finally, Section 7 presents some related works of the literature.

2 Dataset Production and Analysis

The MSTAR dataset [6] comprises SAR measurements of twenty seven different targets taken at different depression and azimuth angles by an airborne radar. Following the standard ATR evaluation procedure with MSTAR [12], we use the data collected at 17° (3202 images) and 15° (3671 images) depression angles as respectively training and test sets. 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. We use MOCEM to reproduce each MSTAR measurement in simulation (Fig. 1). Each vehicle was modelled by a 3D CAD model.

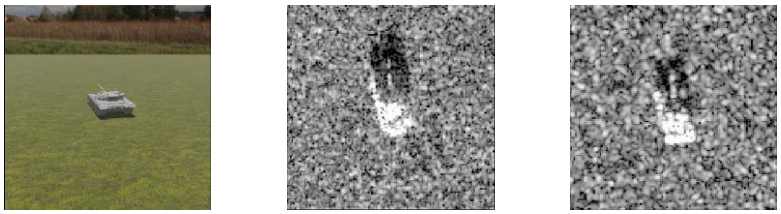


Fig. 1. A T72 CAD Model (left), MSTAR image (center), MOCEM image (right)

Pairwise comparisons of real and synthetic images show a similar Radar Cross Section (RCS) and a high Structural SIMilarity (SSIM) score. This indicates that MOCEM dataset faithfully reproduces the overall structure and physical properties of the MSTAR images. However as shown in Fig. 2, to compare the data distributions, we run a t-SNE [13] that embeds the two datasets in a 2D space while preserving the original distances between the images. We observe that real and synthetic distributions do not completely overlap, meaning that some features are different in the two datasets, which impacts ATR classifiers accuracy as shown in the following section.

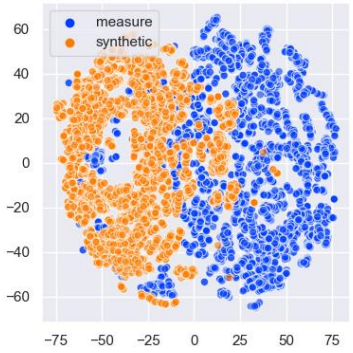


Fig. 2. Two-dimensional t-SNE representation of the MOCEM and MSTAR datasets.

3 Impact of Synthetic Data on ATR Accuracy

To measure the impact of synthetic data on ATR classifiers accuracy, we choose to use Morgan's reference architecture [5] (we achieved similar results with the residual network of He et al. [14]). We produced several training sets for this classifier by varying the proportion of synthetic data they contain. To do so, we randomly substitute a given proportion of the MSTAR measurements by the corresponding MOCEM images. As shown on Fig. 3, in accordance with Morgan's study, a classifier trained with the full 17° MSTAR data reaches approximately 95 % of accuracy on the 15° MSTAR images. The accuracy of the classifier at test time decreases when the proportion of simulated images increases. When we train the classifier with only synthetic data, the accuracy at test time drops to about 33 %. In the following section, we detail how we overcome this issue with our refining model.

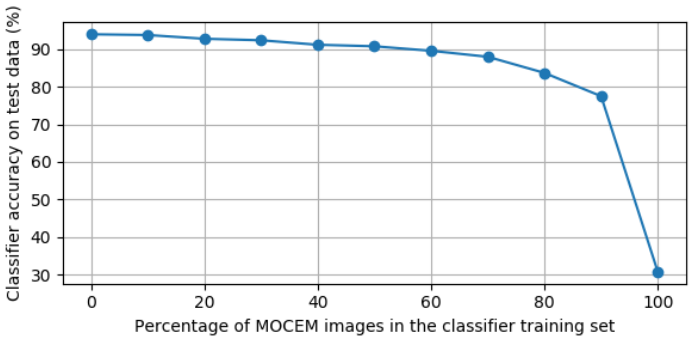


Fig. 3 Impact of MOCEM data on the classifier's accuracy.

4 Refining Model

To reduce the distance between real and synthetic distributions, we train a cGAN [9] to refine simulated data by adding MSTAR-specific features. Following the GAN paradigm, two networks –a generator and a discriminator— are trained in parallel with antagonist objectives. Thus, they compete with each other in a game theory scenario. The generator is tasked to transform the synthetic data it observes. The discriminator is trained to distinguish the transformations made by the generator from the optimal MOCEM-to-MSTAR transformation. Hence, the discriminator observes pairs of images corresponding to input and output of the transformation functions. In the optimal transformation case, the pair is composed of a MOCEM image and the corresponding MSTAR measurement. In the other case, the pair is composed of a MOCEM image and the image refined by the generator. Through an adversarial loss function the generator is trained to fool the discriminator by behaving like the optimal transformation function. The architecture and loss functions used are based on [15]. We train our GAN using the paired 15° MOCEM and MSTAR datasets. Fig. 4 shows a sample of MOCEM images refined with our model. To test our refining model, we refined the whole 17° MOCEM dataset and use it to train the ATR classifier. Finally we measure the accuracy of the classifier on the 15° MSTAR dataset. Thus, the generator is tested with new data not used during its training, and the classifier is still trained on 17° data and tested on 15° measurements, in accordance with the standard MSTAR ATR evaluation procedure.

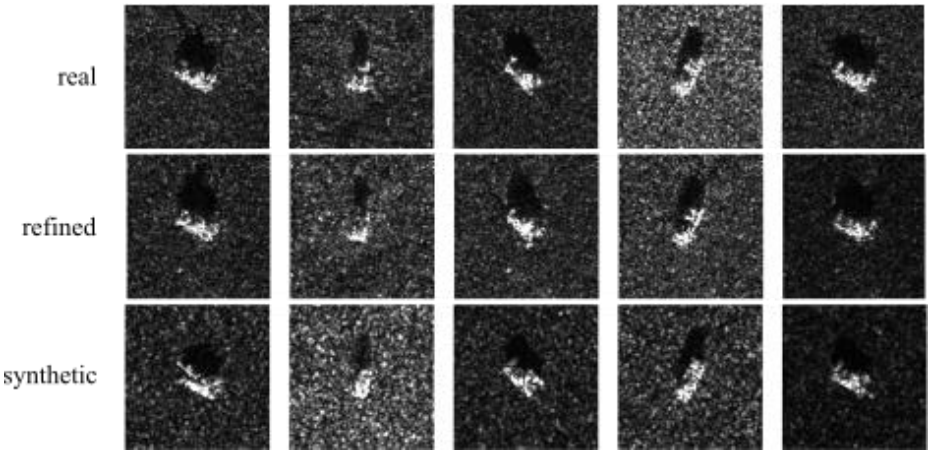


Fig. 4. Sample of corresponding MSTAR, MOCEM, and refined-MOCEM images.

5 Impact of (Multi-)Refined Synthetic Data on ATR Accuracy

We can see from the t-SNE results of Fig. 5a that the refined MOCEM and MSTAR test data distributions overlap better. This may indicate that the refiner successfully transforms simulated data into more “MSTAR like” images. We observe from Fig. 6a

that the ATR classifier trained with refined images achieves an accuracy of around 88 %, that is close to the accuracy of a classifier trained on measured data.

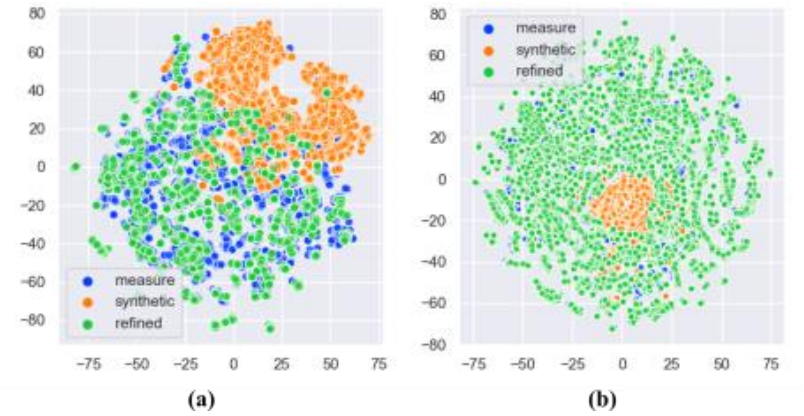


Fig. 5. Two-dimensional t-SNE representation of the MSTAR and MOCEM test sets (a) simple refining, (b) multi-refining.

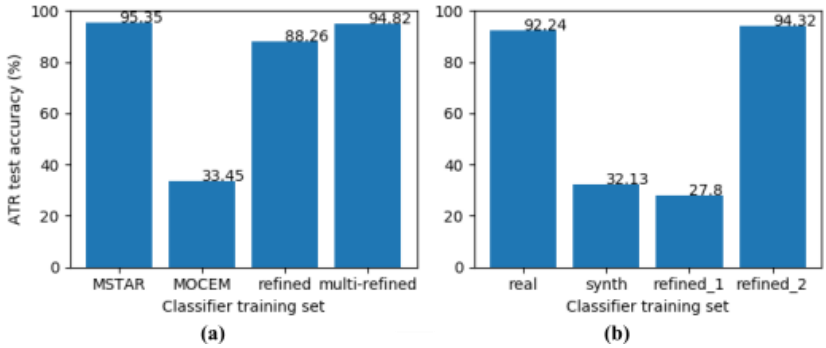


Fig. 6. Accuracy of ATR classifiers trained with (a) MSTAR/MOCEM, (b) SAMPLE (*refined_1* is generated with the MOCEM-trained refiner, and *refined_2* with the SAMPLE-trained refiner, both without multi-refining).

To further increase accuracy, we propose a specific technic of data augmentation called multi-refining. This approach is grounded on the assumption that the cGAN training does not converge to a (may be local) optimum, but instead oscillates around this optimum. The multi-refining technic takes advantage of these oscillations by simultaneously using several refining models produced at different epochs of the GAN training. More formally, multi-refining a synthetic dataset S with n refining models $r_1, r_2, .., r_n$, consists to compute $\{r_1(S), r_2(S), .., r_n(S)\}$. Thus, each original MOCEM image corresponds to n different refined images. As shown in Fig. 5b, we can see that the union of the data distributions produced by eleven different refining models overlaps more the MSTAR distribution than the data distribution produced by a single refining model.

Therefore, an ATR classifier trained with this augmented synthetic dataset reaches a test-time accuracy of almost 95 % that is similar to the accuracy of a classifier trained on real data (Fig. 6a). As shown in the following section, we get similar results with the SAMPLE challenge [11] data.

6 Versatility and Limits of Our Approach

6.1 Application to Other Simulators

To test if our approach can be applied to synthetic data generated by another simulator, we use the dataset of the SAMPLE challenge [11]. This dataset is composed of real MSTAR measurements and their reproduction with a simulator of the Air Force Research Laboratory. Compared to the previous data, the SAMPLE dataset contains five new vehicle classes and the azimuth angle only ranges from 0° to 80°. The dataset is also significantly smaller with 1366 pairs of real and synthetic measurements. As shown on Fig. 6b, we observe similar results than the ones we get with MOCEM. When the classifier is trained with the real SAMPLE data, it achieves an accuracy of 92.24 %. However, a classifier trained with the synthetic SAMPLE data only reaches an accuracy of 32.13 % on the real measurements. The model trained to refine MOCEM images fails to refine the synthetic SAMPLE data: the refined training set (called *refined_1* on the graph) degrades even more the classifier accuracy. However, when the cGAN is trained specifically to refine the synthetic SAMPLE data, the accuracy of the classifier trained on the refined synthetic data (called *refined_2* on the graph) becomes similar to the accuracy of the classifier trained on real data. This means that our approach can be applied to other simulators than MOCEM but that new cGAN trainings are needed.

6.2 Azimuth Generalization and Training Set Reduction

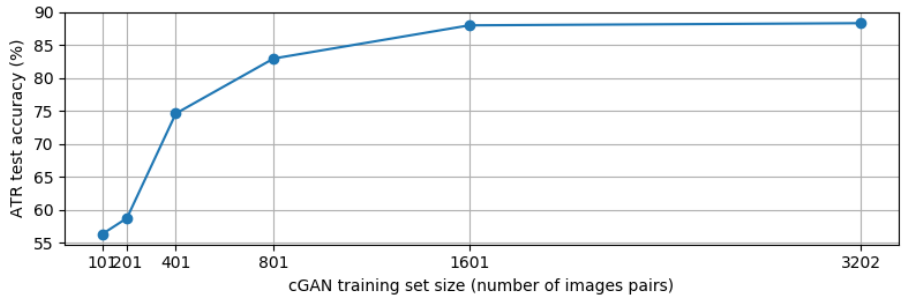


Fig. 7. Influence of cGAN training set on ATR test accuracy

To evaluate if smaller datasets can be used to train our cGAN, we progressively reduce its training set. At each step of this experiment, we sort the training set by targets and azimuth angles, and remove one image out of two. Then, we train the cGAN and test the refining model using the ATR classifier as done previously (nb. we only reduce the

cGAN training set, not the training and test sets of the classifier). Thus, with this test we also evaluate if the refining model is able generalize at test time to new azimuth angles. This is especially interesting considering the important sensitivity of electromagnetic effects (and thus of SAR images) to the azimuth angle caused by the complex shapes of the targets. From Fig. 7, we observe a correlation between the size of the cGAN training set and the accuracy of the ATR classifier. The distance between the distributions of real and refined images increases as the training set becomes smaller – i.e. the smaller the cGAN training set, the lower the quality of refined images, which degrades the ATR model accuracy on real measurements classification. However, we can see that a cGAN trained with only 410 pairs of real and synthetic measurements (i.e. an average of 41 pairs per target class) can be used to train an ATR classifier with almost 75 % of test-time accuracy. Even when the cGAN is trained with an average of only 10 pairs of images per class, the refined synthetic data significantly improves the ATR accuracy that goes from about 32 % without refining, to 56 %.

6.3 Refining New Vehicles Classes

To test if the refining model is able to generalize to new targets, we train the cGAN only on half of the vehicles classes (the first five classes in the alphanumeric order) on the 15° dataset. Then, we refine the whole 17° dataset and train the ATR classifier as done previously. We observe a high ATR accuracy on the classes used to train the cGAN (85.23 %), but a poor accuracy on the classes that are new for the refining model (21.77 %). This means that the refining model somehow encodes target-specific knowledge during the training, and therefore is not able to generalize to new classes of vehicles. This is currently the strongest limitation of our approach.

7 Related Works

Several works trained of ATR classifier with simulated SAR images [16]. These studies faces the same issues we encountered with the MOCEM synthetic data. Malmgren-Hansen et al. propose a transfer learning strategy where the classifier is pre-trained on synthetic data before being trained on a smaller real dataset [17]. However, this strategy requires a non-negligible amount of the training set to be real measurements. Cha et al. trained a residual network to refine synthetic data for ATR [8]. However, their classifier only achieve an accuracy of 55 % at test time. Lewis et al. trained a DualGAN with non-paired data to refine synthetic SAR images, with promising results [18].

8 Conclusion

In this work, we proposed to train a cGAN with pairs of real and synthetic images to refine simulated SAR datasets. We also proposed a multi-refining technic that takes advantage of the oscillations of the cGAN around a (local) optimum to perform data augmentation for synthetic ATR training sets. We have shown that an ATR classifier

can be trained only with refined synthetic data, and achieves similar accuracy on real SAR measurements than a classifier trained directly on measurements. We applied our approach to datasets produced by two different simulators, with similar results, and showed that a refining model has to be trained specifically for each simulator. Despite the fact that small training sets reduce the quality of the refinement, our cGAN can be trained with only a few hundred of pairs of images and still significantly improve ATR accuracy. However, the refining model cannot generalize to new target classes. In future works, we plan to tackle this limitation by performing data augmentation, trying other GAN architectures, and adding a SAR specific loss function to train the network.

9 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 vol.1.
2. H. Wang, S. Chen, F. Xu and Y. Jin, "Application of deep-learning algorithms to MSTAR data," 2015 IEEE IGARSS, Milan, 2015, pp. 3743-3745.
3. A. El Housseini, A. Toumi and A. Khenchaf, "Deep Learning for target recognition from SAR images," 2017 Seminar on DAT, Algiers, 2017, pp. 1-5.
4. Y. Li, et al. "DeepSAR-Net: Deep convolutional neural networks for SAR target recognition," 2017 IEEE ICBDA, 2017, pp. 740-743.
5. D. Morgan. "Deep convolutional neural networks for ATR from SAR imagery." Algorithms for Synthetic Aperture Radar Imagery XXII. Vol. 9475. SPIE, 2015.
6. The MSTAR public datasets. https://www.sdms.afrl.af.mil/index.php?collection=mstar
7. J. Tremblay et al., "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization," 2018 IEEE/CVF CVPRW, 2018, pp. 1082-10828.
8. M. Cha, et al., "Improving Sar Automatic Target Recognition Using Simulated Images Under Deep Residual Refinements," IEEE ICASSP, 2018.
9. I. Goodfellow, et al. 2014. Generative adversarial nets. In Proceedings of NIPS'14. MIT Press, Cambridge, MA, USA, 2672–2680.
10. C. Cochin, et al. "Radar simulation of ship at sea using MOCEM V4 and comparison to acquisitions," 2014 International Radar Conference, Lille, 2014, pp. 1-6.
11. 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.
12. T. Ross, et al., "Standard SAR ATR evaluation experiments using the MSTAR public release data set," Proc. SPIE 3370, Algorithms for Synthetic Aperture Radar Imagery V, 1998
13. L.J.P. van der Maaten and G.E. Hinton, “Visualizing High-Dimensional Data Using t-SNE”, Journal of Machine Learning Research, vol. 9, nov. 2008, p. 2579–2605
14. K. He, et al., "Deep Residual Learning for Image Recognition," IEEE CVPR, 2016
15. P. Isola, et al., "Image-to-Image Translation with Conditional Adversarial Networks," *2017 IEEE CVPR,* pp. 5967-5976
16. 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.
17. 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.
18. B. Lewis, J. Liu, A. Wong, "Generative adversarial networks for SAR image realism," Proc. SPIE, Algorithms for Synthetic Aperture Radar Imagery XXV, 1064709, 2018.

Active learning for object detection in high-resolution satellite images

Alex Goupilleau[1], Tugdual Ceillier[1], and Marie-Caroline Corbineau[1]

[1]Earthcube, 75009 Paris, France
www.earthcube.eu, [name].[surname]@earthcube.eu

Abstract. In machine learning, the term *active learning* regroups techniques that aim at selecting the most useful data to label from a large pool of unlabelled examples. While supervised deep learning techniques have shown to be increasingly efficient on many applications, they require a huge number of labelled examples to reach operational performances. Therefore, the labelling effort linked to the creation of the datasets required is also increasing. When working on defense-related remote sensing applications, labelling can be challenging due to the large areas covered and often requires military experts who are rare and whose time is primarily dedicated to operational needs. Limiting the labelling effort is thus of utmost importance. This study aims at reviewing the most relevant active learning techniques to be used for object detection on very high resolution imagery and shows an example of the value of such techniques on a relevant operational use case: aircraft detection.

Keywords: Deep learning · Convolutional neural networks · Earth observation · Active learning · Object detection.

1 Introduction

1.1 Context

Active learning is a sub-domain of machine learning, whose objective is to smartly select the relevant data to label in order to maximize the performance of a given model. Active learning can help achieve two goals:

- Get the best possible performance given a fixed labelling budget,
- Minimize the labelling effort to reach some target performance.

In remote sensing use cases, the labelling effort is particularly laborious, as one often has to find objects of interest in very large images, before creating a label for them. The integration of active learning methods could therefore allow annotators to focus only on the most relevant images or zones inside images – which would make the whole labelling process easier, faster, and more efficient.

In particular, deep learning models – including convolutional neural networks (CNN) – would greatly benefit from these methods, as these types of machine learning models generally need a large amount of labelled images to work well. In a larger context, active learning methods could help improve the whole lifecycle of an algorithm: on a regular basis, some raw images could be ingested in a database of unlabelled images, of which only a relevant subset would be selected by active learning methods in order to be labelled and used for the improvement of a given deep learning model.

These techniques are particularly interesting when the user has limited labelling capabilities and owns a lot of unlabelled images – and one can expect the latter to be especially true in the future where the number of sensors is assumed to increase over time.

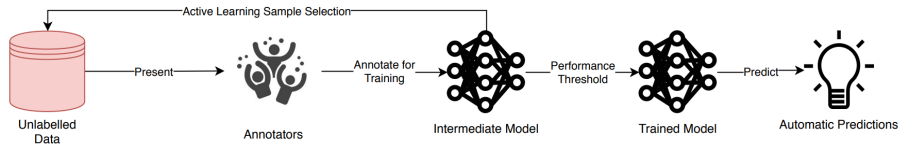


Fig. 1. Schematics of the active learning iterative process (from [3]).

1.2 Problem setting

The objective of this section is to formulate the problem that active learning methods try to solve, as well as to establish the necessary vocabulary to understand the different methods that will be described in the following subsections.

The classic active learning scenario can be described in the following way: a given operator owns a database of unlabelled images B and a CNN classification model M – that takes as input an image X and outputs class probabilities $p(X \in c) = M(X)$.

The operator has a given labelling budget: this means that he can only label a maximum of N images in the database B. After selection and labelling, the resulting labelled dataset can be called $B*$. The objective of any active learning method is to find the optimal $B*$ subset of images in B that maximizes the performances of the model M on its task, when trained on $B*$.

In practice, this scenario is often iterative. The operator already owns a dataset of labelled images that was used to train an initial model. Then, at each step, the active learning technique selects new unlabelled images that are then labelled and used to retrain or fine-tune the initial model, either on the concatenation of the initial dataset and the new images or on the selected images only. This iterative process can be stopped if targeted performances or maximum labelling budget are reached, or continuously used to accommodate for novelties in the data distribution (in the remote sensing use case, it can be new types of objects, new geographic areas, new sensors). Figure 1 illustrates this iterative process.

2 Related work

The objective of an active learning method is to evaluate the informativeness of a given image X, i.e. how much information that image can bring to the model M when it is trained on it. To do so, most techniques use the model itself. The different approaches used by active learning methods differ in the way informativeness is defined. The different strategies can be split into four categories: uncertainty-based active learning, representativity-based active learning, hybrid methods that combine uncertainty and representativity, and methods that use neither. In this section, we will detail examples of these four categories.

2.1 Uncertainty-based active learning

The techniques based on uncertainty select images that generate the most uncertainty for the model. A well known and relevant technique to estimate model uncertainty is the one used by [6]. In order to evaluate uncertainty, the authors use Bayesian dropout at inference time to obtain slightly different predictions. The authors test different ways of selecting images based on these multiple predictions, called *acquisition functions*. Using the MNIST dataset, they show that compared

to a random selection of the images, all acquisition functions yield positive performance gains – except one based on the variance – and that these techniques work especially well if the labelling budget is low.

Other approaches also use uncertainty in order to select relevant images to label, either by using an ensemble of classifiers [2], or by using generative adversarial networks (GANs) to compute the distance between an example and its adversarial equivalent [5].

2.2 Representativeness-based active learning

One drawback of uncertainty-based methods is that they tend to select very similar images. In order to counter this problem, methods based on representativeness aim is to select a dataset $B*$ such that the images in this subset are as representative as possible of the whole dataset B.

A popular example of these technique is described in [12]. The authors show that this problem can be solved with the geometric K-Center problem, using the L2 distance between activations of the last dense layer in the CNN. Using the CIFAR-10 and CIFAR-100 datasets, they compare their selection method with various other techniques, including a random selection, and the uncertainty approach by [6]. For both datasets, the technique seems to yield better results than all other approaches when the the labelling budget is less than 40% of the dataset. According to the authors, the technique seems more efficient when the number of classes is low. They also conclude that uncertainty could be added to the recipe in order to improve the method.

A more recent work [7] presents a technique inspired from GANs [8] by deceiving a discriminator so that it can no longer decide if an example image selected by the generator belongs to $B*$ or B. Once the generator is trained well enough, it can be used to generate a dataset $B*$ which is representative of B. Using the MNIST and CIFAR-10 dataset, they compare it to other popular approaches and conclude that their method works as well as any other method. This approach is interesting because it is simple to adapt to other tasks such as object detection or segmentation. However, this technique requires an additional optimization step – which makes the process much more computation-heavy than other methods.

2.3 Hybrid active learning

Few recent papers propose relevant methods that try to get the best of uncertainty and representativeness. Among them is [1]. The authors consider that uncertainty can be evaluated by studying the magnitude of the loss gradients in the last layer of the network: if the model is certain about a given prediction, the magnitude of the gradients would be low, and model parameters would only be weakly updated. The same loss gradients are used to evaluate diversity: a batch of examples has a good diversity if the gradients of the examples have complementary directions. Computing loss gradients is not possible if one does not have ground truth labels. To overcome this, the authors use model predictions instead of the groundtruth to compute the gradients. The authors test their method on a large diversity of use cases by varying the number of selected images, the CNN architecture used, and the dataset used (SVHN, OpenML #156, CIFAR-10). They compare their method with the uncertainty-only approach and the diversity-only approach (Core-Set). The authors conclude that their approach is almost always the best, and that it seems to be robust to the diversity of use-cases.

2.4 Other methods

Some other methods in the litterature offer to solve the active learning problem, without directly measuring uncertainty or representativeness. This is the case of [13], where the authors propose to learn to predict the value of the loss during training by using a loss prediction module. The core hypothesis is that the loss value is directly linked to the quantity of information that a single example brings to the network during training. A clear advantage is that the module can be easily stacked to any architecture, meaning that the method can be easily adapted from the classification tasks to segmentation or detection tasks. For the detection task, the results seem to indicate that the Core-Set approach works better when the number of labelled images is low, whereas the loss prediction approach seems to work better when this number is higher.

Interestingly, some recent approaches have tried to treat the active learning problem as a reinforcement learning task. In [4], the authors propose to train a reinforcement learning algorithm to select the most useful patches in images from two public datasets – CamVid and Cityscape – to improve an existing segmentation model. Their solution is not straightforward to implement and remains more computation heavy than the others presented here but their results on both datasets seem to outperform other uncertainty-based approaches. It will be interesting to watch closely such reinforcement learning approaches in the future as they gain in maturity.

3 Proposed approach

In this work, we apply two active learning techniques to segmentation of satellite images: Bayesian dropout from [6] and Core-Set from [12]. However, we need to adapt these methods to the segmentation task, as they were designed for image classification, and to allow the use of large rasters instead of pre-calibrated data.

3.1 Pre-selection of tiles

In order to use large raster images, the first step is to split the original images into fixed-size tiles. We choose 512x512 tiles to prevent any memory-related issues during training. However, considering the amount of data we use, processing all created tiles (several hundred thousands) through the active learning techniques proves unpractical. For this reason, we choose to perform a first selection on the tiles, before applying the active learning techniques. To do so, we use the initial network to predict once on all available tiles and we then order these tiles according to the average intensity of the corresponding segmentation map. This technique comes from initial experiments showing that tiles with a very low response were almost never among the ones selected by active learning techniques.

This pre-selection therefore allows to speed up considerably the whole process while not modifying the selection performed through active learning. The number of selected tiles is set to 5% of all the available tiles, based on computational limitations.

3.2 Bayesian dropout

The first active learning method we use is derived from [6]. While this method was developed for image classification, we modify it to apply it for semantic segmentation. To do so, we follow the same approach of predicting multiple times on

a given tile while activating dropout in the network, using the same parameters as the ones used during training. We choose to perform 10 predictions, to get sufficient variations while limiting the computing cost of the method.

We then need to derive an estimation of the uncertainty of the whole tile from the 10 segmentation maps thus obtained. We first compute, for each pixel, the variance of the 10 values. Then, we take the average of all the variances as an estimation of the uncertainty of the 512x512 tile.

3.3 Core-Set

The second active learning method we use uses the approach of [12]. Similarly to the first method, this technique has been developed for image classification. To adapt it to segmentation, one need to find a way to derive a reasonably-sized vector from the features extracted by the network.

As the CNN we use has a U-Net structure, we use the feature map at the end of the decoder, which has a size of 128x128x128 (width, height, and number of filters). We first use max pooling to get an 8x8x128 matrix and then average pooling to compress it into a 1x1x128 matrix, interpreted as a 1D vector. We then use the Robust k-center algorithm from [12] to select the k most representative tiles, k being equal to our labelling budget.

4 Experiments and results

To evaluate the potential of active learning for a defense-related use-case, we choose to test the developed methods on aircraft detection, using image segmentation models. As a first step, we performed only one iteration of the different methods and measure the performance increase compared with random selection.

4.1 Initial models and pool of unlabelled images

Before applying the active learning techniques, we need to train initial models which will be incrementally improved. We choose to create two different models, with the same modified U-Net architecture [11], that correspond to two different use-cases:

- a *weak model* that has been trained on relatively few images, corresponding to a model still in development that we want to improve as fast as possible;
- a *strong model* that has been trained on much more data and that reaches correct performances, representative of a model in production that could be improved with time.

We choose to select full-sized satellite images based on their date of acquisition, to mimic a real-life situation where images are gradually available. In this work, we use images from *Maxar/DigitalGlobe* satellites WorldView 1, 2, and 3 and GeoEye 1. For the *weak model*, we select images acquired 2010 January 1st and 2012 January 1st and for the *strong model* we extend the second limit to 2017 January 1st. We then create a dataset from these images by splitting them into 512x512 tiles, with 90% of the tiles being the ones containing aircraft and 10% of the tiles being randomly sampled among negative tiles. We train each model until convergence, using the Adam optimizer [9] and weighted cross-entropy loss.

Now that we have initial models to improve, we need a vast amount of "unlabelled" images to select from. In our case, all the labels are already available, but only imagettes selected through active learning will be actually used. Following

Table 1. Data used to train the initial models and create the pool of unlabelled images.

Model	Acquisition period	# images	Total area	# aircraft	# tiled imagettes
Weak	2010/01/01-2012/01/01	40	222 km^2	951	2,740
Strong	2010/01/01-2017/01/01	3,891	14,445 km^2	67,008	54,851
Pool	2017/01/01-2017/07/01	467	5,905 km^2	11,017	$\sim$ 400k possible

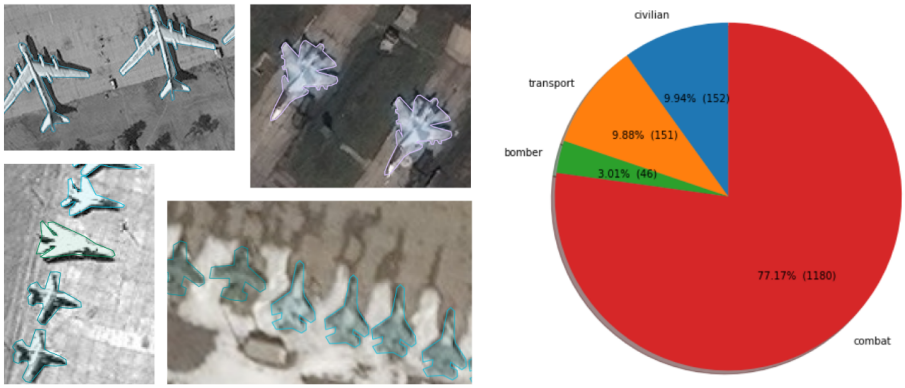


Fig. 2. Testing set used to measure performances. Left: examples of labelled aircraft. Right: Distribution of the aircraft models between four categories (combat, bomber, transport, and civilian).

the same approach we followed to train the initial models, we select images from the *Maxar/DigitalGlobe* satellites WorldView 1, 2, and 3 and GeoEye 1 based on their acquisition date. We therefore choose only images acquired between 2017 January 1st and 2017 July 1st. It is worth noting that while the images used to train the initial models contained mostly civilian aircraft, the images in our pool contains mostly military aircraft, which is an interesting setup to see if our models can improve on these new aircraft types.

Table 1 summarises the data used to train the initial models and to create the pool of "unlabelled" images.

4.2 Testing set and performance metrics

To measure the performances of the different models, we use a testing set composed of images present neither in the data used to train the initial models nor in the pool of unlabelled images. To focus the evaluation on military aircraft, we select 30 full-size satellite images from 16 different locations that contains mostly military aircraft. These images contain 1532 individual aircraft in total. Figure 2 presents examples of the labelled aircraft and the distribution of the models present in the testing set.

The metrics we consider are the classical precision, recall and F1-score. An aircraft in the ground truth is considered as a true positive if it is at least partly covered by the predicted segmentation. On the other hand, the segmentation outputed by the network is divided into connected components; if one component does not cover any aircraft in the ground truth, it is considered as a false positive. To compare the behaviour of different models, we consider the precision-recall curve of each model, obtained by varying the threshold applied to the segmentation output in order to obtain a binary map aircraft/background. We then choose as the operational threshold the one yielding the best F1-score.

Table 2. Labelling budget and number of tiles of the different approaches compared and gains in F1-score.

Model	Approach	Labelling budget	# Tiles selected	F1-score gain
Weak	Baseline *Unlimited*	unlimited	5700	+6.6%
	Baseline *Random*	1000	1000	+2.0%
	Bayesian dropout	1000	1000	-0.2%
	Core-Set	1000	1000	**+6.7%**
Strong	Baseline *Unlimited*	unlimited	5700	+3.6%
	Baseline *Random*	5000	5000	+4.5%
	Bayesian dropout	5000	5000	**+8.0%**
	Core-Set	5000	5000	+6.6%

4.3 Results

To evaluate the contribution of the uncertainty and core-set approaches, we compare them to two different baselines. The first baseline, called *Unlimited*, corresponds to an unlimited labelling budget, with all the objects in the pool of images being labelled. In this case, a training set is built by selecting all the tiles containing aircraft (90% of the set) and randomly selected negative tiles (10% of the set), leading to a training set containing 5700 tiles, which is higher than the labelling budgets used for the active learning approaches. The second baseline, called *Random*, uses the same labelling budget as the active learning approaches but in this case the tiles are randomly selected after the pre-selection step.

To improve the *weak model*, we choose a labelling budget of 1000 tiles while for the *strong model* we choose a higher budget of 5000 tiles. This choice was made to match the difference in the number of tiles in their initial training sets. Table 2 summarises the labelling budget and the number of tiles added to the initial datasets for the different approaches, as well as the gains in term of F1-score of each experiment.

Results can be seen in Figure 3. For the *weak model*, the best approach appears to be the Core-Set. It leads to an F1-score increase of 6.7% while the Baseline *Unlimited* – that selects more than 5 times more tiles – improves the F1-score of 6.6%. However, in the high recall regime, the Baseline *Unlimited* is performing better. This is likely due to the fact that the model trained for this baseline has bee presented with all the new aircraft while active learning techniques cannot necessarily select all the tiles containing aircraft due to their limited labelling budget. Interestingly, the Bayesian dropout approach does not lead to any significant improvement over the initial model and is outperformed by the Baseline *Random*. This only exception lies in the very high precision regime, meaning that the Bayesian dropout impact is limited to removing some false positives detected by the initial *weak model*.

For the *strong model*, even if the Core-Set approach is also performing well (+6.6% f F1-score), the Bayesian dropout approach outperforms it by a large margin (+8.0% of F1-score). Both active learning approaches are behaving much better than the baselines – including the *Unlimited* one that selects 700 tiles more – which shows the usefulness of selecting the examples to present to an already well-performing model to improve it. This follows the intuition than as a model gets more and more efficient, new data bring relatively small information to it. It is especially striking to see that active learning methods perform on part with the *Unlimited* baseline in the high recall regime. This might be an indication that the *strong model* has learned more generic features, which can be leveraged by both active learning methods to identify difficult aircraft models.

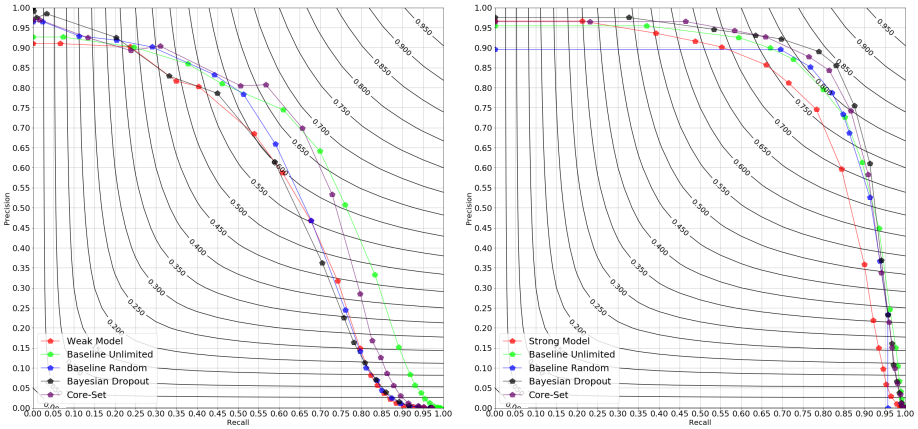


Fig. 3. Performance of the various approaches. Left: weak model. Right: strong model. The contour lines indicate similar values of the F1-score.

5 Conclusion

In this work, we have presented the various techniques of active learning that can be used to improve a object detection algorithm working on very high resolution remote sensing images. We have adapted two approaches – Bayesian dropout and Core-Set – to the segmentation task and tested them on two CNN models with different levels of performances. By comparing them with two baselines, we have shown that active learning approaches are very efficient to boost performances of existing models while limiting the amount of labels needed. The most consistent active learning technique tested here seems to be the Core-Set one, as it yields good results on both test cases. However, the Bayesian dropout technique is significantly more efficient for the strong model, which could imply the necessity to change the active learning strategy at one stage during model development.

Future developments could include a variety of different experiments. First, it could be informative to apply these two techniques in an iterative manner, with several pools of data. It could allow to identify at which point the uncertainty approach becomes more efficient than the core-set one. Another obvious option would be to follow the same experimental on other types objects, such as terrestrial vehicles or vessels for instance. Finally, these techniques still rely on the initial training set, which both makes the new trainings long and can be impractical in an industrial context where data cannot be transferred. Combining the active learning techniques developed in this paper with continuous learning such as Elastic Weight Consolidation [10] to avoid re-using initial data could lead to an even more operational solution.

Even with the aforementioned cautions, we believe that this work has proven how efficient and relevant active learning solutions are in a defense-related context. The techniques presented here have a high potential to ease adoption of CNN-based solutions into the workflow of military analysts. They can allow them to adapt products developed on commercial data to their own specific needs, either linked to sovereign sensors or to new types of objects of interest.

References

1. Ash, J.T., Zhang, C., Krishnamurthy, A., Langford, J., Agarwal, A.: Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In: ICLR (2020)

2. Beluch, W.H., Genewein, T., Nürnberger, A., Köhler, J.M.: The power of ensembles for active learning in image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9368–9377 (2018)
3. Budd, S., Robinson, E.C., Kainz, B.: A Survey on Active Learning and Human-in-the-Loop Deep Learning for Medical Image Analysis. arXiv preprint arXiv:1910.02923 (2019)
4. Casanova, A., Pinheiro, P.O., Rostamzadeh, N., Pal, C.J.: Reinforced active learning for image segmentation. arXiv preprint arXiv:2002.06583 (2020)
5. Ducoffe, M., Precioso, F.: Adversarial active learning for deep networks: a margin based approach. arXiv preprint arXiv:1802.09841 (2018)
6. Gal, Y., Islam, R., Ghahramani, Z.: Deep bayesian active learning with image data. arXiv preprint arXiv:1703.02910 (2017)
7. Gissin, D., Shalev-Shwartz, S.: Discriminative active learning. arXiv preprint arXiv:1907.06347 (2019)
8. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014), http://arxiv.org/abs/1406.2661, arXiv: 1406.2661
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., others: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017), publisher: National Acad Sciences
11. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
12. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489 (2017)
13. Yoo, D., Kweon, I.S.: Learning loss for active learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 93–102 (2019)