Analyse de *malwares* et classification à l'aide de réseaux de neurones convolutionnels 1D

Romain Leveau(1), Jean-Sébastien Lerat(2)

Haute École en Hainaut (1) (2), ESTISIM (2)

jean-sebastien.lerat@heh.be (2)

Résumé

Les logiciels malveillants peuvent avoir des impacts particulièrement négatifs sur notre société, d'autant plus que celle-ci a de plus en plus recours à des systèmes informatiques. En vue de garantir la sécurité de ces systèmes, la détection et la prévention de ces logiciels malveillants sont nécessaires. Dans cette optique, l'objectif de ce travail est de procéder à une classification entre des exécutables sains et des fichiers malveillants appelés maliciels (*malwares*). Pour ce faire, nous avons récolté des rapports qui reprennent l'activité d'un fichier exécutable qui est lancé en mémoire dans une machine virtuelle. Ensuite ces fichiers sont traités dans le but d'extraire les données y étant contenues pour les rendre utilisables et y appliquer des réseaux de neurones convolutionnels en vue de procéder efficacement à cette classification. Dans ce travail la convolution 1D est exploitée comme première étape vers la prédiction du comportement des maliciels en analyse dynamique en tant que série temporelle.

Mots-clés : *deep learning, framework, machine learning, malware, cybersecurity*.

1 Introduction

Ces dernières années, de nombreuses cyberattaques ont fait la une de la presse. A titre d'exemple, celle du logiciel malveillant WannaCry qui aurait touché plus de 200 000 victimes d'après Europol (WANNACRY RANSOMWARE : RECENT CYBER-ATTACK (2017)), en particulier des entreprises dans environ 150 pays différents.

La détection de maliciels inédits est une tâche compliquée pour les antivirus qui se basent principalement sur une signature, une emprunte numérique, soit du virus soit sur un ensemble d'instructions qui correspondent à un comportement (exemple : ouvrir une communication Internet). De plus, les maliciels peuvent évoluer et se réécrire comme les *shellcodes* polymorphes et métamorphes. A cette complexité s'ajoute le fait qu'un maliciel peut être fragmenté en différents morceaux logiciels qui coopèrent afin de mener à bien une attaque.

Pour pallier ces méthodes malveillantes, l'apprentissage automatique peut être utilisé comme le montre la littérature Sgandurra, Muñoz-González, Mohsen & Lupu (2016). Toutefois les méthodes d'apprentissage automatique traditionnelles ne prennent pas en considération la coopération de maliciels dans un temps imparti. C'est pourquoi dans cet article, il est envisagé d'exploiter l'apprentissage profond exécuté sur un matériel spécifique (exemple : GPU) afin d'améliorer le temps d'exécution et la détection de ces maliciels coopératifs.

Comme première étape du projet, un réseau de neurones capable de reconnaître un comportement malveillant doit être conçu. Cette problématique est l'objet du présent travail. Le jeu de données est constitué de rapports d'exécution de maliciels, effectués par l'outil Cuckoo Sandbox. Ceux-ci vont servir à entraîner un réseau de neurones convolutif qui a la particularité de ne pas utiliser la convolution à deux dimensions (2D) mais bien à une seule dimension (1D). Un rapport Cuckoo est une suite de

chaîne de caractères sans relation directe par opposition aux images qui sont constituées de pixels où chacun a un lien direct avec les pixels voisins de la même ligne ou de la même colonne. Les images possèdent donc une relation spatiale, ce qui explique l'utilisation de la convolution 2D. Ce qui n'est pas le cas du jeu de données de maliciels.

Dans ce travail, plusieurs architectures sont envisagées et adaptées afin d'atteindre une précision supérieure à 90 %. A la suite des résultats, un protocole de simulation va être mis en œuvre afin de tester la solution logicielle sur un virus inédit conçu spécifiquement pour la simulation. Ce virus est composé d'un ensemble de logiciels qui doivent interagir afin de gagner des privilèges.

2 État de l'art

Récemment, la recherche au croisement de l'auto-apprentissage et de la cybersécurité s'est intensifiée comme le démontre Dua & Du (2016), à cause de la forte croissance du nombre de maliciels inédits. Les méthodes de détections traditionnelles sont encore utilisées comme le montre Polychronakis, Anagnostakis & Markatos (2010). Celles-ci consistent à calculer une signature, c'est-à-dire un grand nombre qui représente de manière unique[1] un morceau de code reconnu comme étant malveillant. Lorsqu'un antivirus calcule ce nombre sur des fichiers et obtient la même valeur, c'est que le code correspond à un comportement malveillant. Preda, Christodorescu, Jha & Debray (2007) ont adapté cette méthodologie afin de détecter les maliciels à l'aide de règles sémantiques. Par la suite, l'utilisation de l'apprentissage automatique permet aux antivirus de concevoir des heuristiques de détection, combinées aux signatures.

Kolosnjaji, Zarras, Webster & Eckert (2016) ont utilisé les réseaux de neurones sur d'anciens maliciels au format PE (exécutable Windows 32bits). Le jeu de données est constitué de rapport d'un logiciel nommé Cuckoo Sandbox qui exécute le maliciel dans un environnement bac à sable, c'est-à-dire isolé d'un système sain. Le pré-traitement du jeu de données consiste à supprimer des actions répétées comme la création de fichier et à convertir les rapports Cuckoo Sandbox à l'aide de l'encodage *one-hot*. Bien que les résultats soient intéressants, cette méthodologie ne peut pas être appliquée en temps réel à cause de la suppression des actions répétées. En effet, ceci nécessite un post-traitement, c'est-à-dire lorsque le maliciel a terminé son exécution. De plus l'encodage *one-hot* détruit la précision des informations en réduisant simplement les données sous forme d'un vecteur binaire où 1 signifie la présence d'une action et 0 l'absence d'une action.

Au lieu de s'intéresser à l'ensemble des traces d'exécution, Rathore, Agarwal, Sahay & Sewak (2018) se sont concentrés sur les instructions CPU appelées *opcode*, exécutées par les maliciels. Les maliciels sont alors analysés sur base de la fréquence d'utilisation des *opcodes*. Cette méthodologie réduit plus drastiquement les informations disponibles par rapport à l'exécution d'un maliciel mais permet de réduire considérablement la quantité des données à analyser. La représentation sous forme de fréquence nécessite un post-traitement mais si cette information était connue avant l'exécution d'un maliciel, il pourrait être détecté en temps réel étant donné la vitesse d'analyse engendré par la réduction de dimensionalité.

Sahay & Rathore (2018) ont comparé la prédiction de maliciels entre l'apprentissage profond et une forêt d'arbres décisionnels, une méthode classique de l'apprentissage automatique. Le jeu de données est constitué à l'aide de l'analyse statique des *opcode*. Ils ont montré que dans certains cas, les méthodes traditionnelles étaient plus performantes. Toutefois, le jeu de données n'exploite que l'analyse statique,

[1] Une méthode de hachage est utilisée à cette fin, la valeur n'est pas unique mais à une faible probabilité d'être associée à un autre morceau de code.

ce qui ne permet d'analyser que des maliciels basiques. En effet, des logiciels malicieux plus sophistiqués existent et ne sont pas analysables à l'aide de l'analyse statique. Par exemple lorsque l'obfuscation, la compression, le chiffrement sont utilisés ou bien lorsque les maliciels sont polymorphes ou métamorphes.

Hasegawa & Iyatomi (2018) ne s'intéressent pas aux maliciels s'exécutant sous Microsoft Windows au format exécutable mais aux maliciels d'Android au format APK. Ceux-ci exploitent l'apprentissage profond mais ils conçoivent un réseau de neurones qui n'utilise qu'une seule dimension leur permettant ainsi d'accélérer le traitement.

Les travaux plus récents de Ijaz, Durad & Ismail (2019) s'intéressent à des maliciels modernes représentés via leurs traces d'exécution récoltées à l'aide de l'outil Cuckoo Sandbox. Ceux-ci conçoivent une solution qui analyse le comportement sur base des appels aux différentes API, l'utilisation du registre Windows, l'interaction de DLLs et des informations générales à propos de l'exécutable comme sa taille. Les auteurs montrent qu'un meilleur taux de prédiction est atteint lors de l'analyse statique parce qu'il y a plus de maliciels simples mais que l'analyse dynamique est préférable à cause des mécanismes de protection des maliciels. Les auteurs ne s'intéressent pas à alléger leur modèle afin de l'exploiter en temps réel.

Contrairement aux travaux antérieurs nous nous intéressons à concevoir une solution basée sur l'apprentissage profond en entraînement un réseau de neurones convolutionnels très petit afin de rendre possible la détection en temps réel. De plus le jeu de données est constitué de maliciels récents, ce qui est plus adapté aux systèmes d'exploitation modernes.

3 Modèles et méthodes

Dans cette section la méthodologie utilisée afin de récolter les données et de les prétraiter pour les formater en entrée d'un CNN est expliquée. Les CNN choisis et transformés afin d'utiliser une convolution 1D sont également décrits.

3.1 Jeu de données

Afin de constituer un jeu de données par rapport à des fichiers exécutables considérés comme sains et malveillants, l'utilitaire Cuckoo Sandbox a été utilisé. C'est un système d'automatisation d'analyse de maliciels qui permet de tracer le comportement d'un exécutable lancé au sein d'une machine virtuelle. Cuckoo Sandbox génère un rapport au format *json* sur base des interactions avec le système d'exploitation à travers le système de fichiers, les interactions réseaux et bien d'autres caractéristiques.

Bien que ces rapports soient au format texte, ils peuvent être exploités par les réseaux de neurones. LeCun, Bottou, Bengio & Haffner (1998) ont montré qu'il est possible de reconnaître les documents à l'aide d'une approche basée sur le gradient.

Les maliciels analysés sont issus de www.malshare.com et les fichiers bénins sont issus du site www.portablefreeware.com, ils ont tous été analysés dans un environnement virtuel sous Windows 7. Tous les fichiers analysés sont au format PE. La structure générale d'un rapport de Cuckoo Sandbox est présentée en Figure 1. Ce rapport reprend différentes informations à propos des exécutables comme les méta-informations et les bibliothèques (fichier DLL) chargées mais également leur comportement via leurs interactions avec le réseau ou leurs appels systèmes.

Figure 1 : Structure d'un rapport JSON de Cuckoo Sandbox.

Cette collecte de données a permis de constituer un jeu de données de 3103 maliciels et de 1890 fichiers considérés comme sains, disponibles à l'adresse https://bit.ly/3b2I2HG. L'analyse n'a pas pu être menée à bien par Cuckoo Sandbox sur 17 fichiers, ce qui constitue une perte de 0,01 % de rapports d'analyse. Contrairement à l'utilisation de l'encodage *one-hot*, nous avons opté pour une représentation séquentielle des caractéristiques où la granularité de l'information est un peu moins fine mais permet d'être plus significative qu'une valeur binaire. A titre d'exemple, si un fichier tente d'accéder à 10 adresses web différentes qui ne sont spécifiques qu'à ce maliciel, la binérisation va créer 10 attributs. Chacun d'eux va correspondre aux 10 adresses web avec une valeur de 1 signifiant la présence de de l'URL dans le comportement du fichier exécutable. Par contre ces 10 attributs seront aussi utilisés afin de représenter le comportement des autres fichiers mais ils auront une valeur de 0 signifiant que les URL ne sont pas utilisées. Cependant la représentation choisie dans ce travail ne produirait qu'une seule caractéristique dont la valeur est 10 pour ce maliciel et peut varier pour les autres fichiers en fonction du nombre de tentatives d'accès à des adresses web. L'idée sous-jacente est qu'un maliciel a un objectif spécifique et va donc avoir tendance à avoir des valeurs importantes sur l'une ou l'autre caractéristique. Un maliciel publicitaire va tenter d'accéder à beaucoup d'adresses différentes afin de présenter diverses publicités. Un échantillon du jeu de données prétraité obtenu est repris sur la Figure 2.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	name	category	tls	udp	dns_servers	http	icmp	smtp	tcp	smtp_ex	mitm	hosts	pcap_sha256	dns	http_ex	domains	dead_hosts	sorted_pcap_sha256	irc	https_ex	pdb_path
2	000a3ea381	mal_reports	0	36	1	0	0	0	0	0	0	1	64	5	0	4	0	64	0	0	0
3	000adc56f2d	mal_reports	0	12	1	0	0	0	0	0	0	1	64	2	0	2	0	64	0	0	0
4	000b28e7a5	mal_reports	0	43	1	0	0	0	0	0	0	19	64	4	0	3	0	64	0	0	0
5	000c7ff25b8	mal_reports	0	41	1	0	0	0	0	0	0	1	64	6	0	5	0	64	0	0	18
6	000cbb278e	mal_reports	0	25	1	0	0	0	0	0	0	2	64	4	0	3	1	64	0	0	0
7	000ed458b7	mal_reports	0	17	1	0	0	0	0	0	0	1	64	3	0	2	0	64	0	0	0
8	000f6d35ea	mal_reports	0	22	1	0	0	0	0	0	0	1	64	4	0	3	0	64	0	0	0
9	001a255c29	mal_reports	0	37	1	0	0	0	0	0	0	1	64	9	0	8	0	64	0	0	0
10	001cfa63ad7	mal_reports	0	24	1	0	0	0	0	0	0	1	64	4	0	3	0	64	0	0	0
11	001d106b8f7	mal_reports	0	24	1	0	0	0	0	0	0	1	64	4	0	3	0	64	0	0	0
12	001eaf022d	mal_reports	0	27	1	0	0	0	0	0	0	4	64	4	0	3	3	64	0	0	0
13	001ed0ac1f	mal_reports	0	30	1	0	0	0	0	0	0	1	64	4	0	3	0	64	0	0	0
14	00a03d37ba	mal_reports	0	16	1	0	0	0	0	0	0	1	64	3	0	2	0	64	0	0	0
15	00a0f5fe1ba	mal_reports	0	31	1	0	0	0	0	0	0	8	64	4	0	3	6	64	0	0	0
16	00a38bce68	mal_reports	0	12	1	0	0	0	0	0	0	1	64	2	0	2	0	64	0	0	0
17	00a473640f8	mal_reports	0	17	1	0	0	0	0	0	0	1	64	3	0	2	0	64	0	0	0
18	00a4a67dc6	mal_reports	0	20	1	0	0	0	0	0	0	1	64	3	0	2	0	64	0	0	0
19	00a4b6a503	mal_reports	0	15	1	0	0	0	0	0	0	1	64	3	0	2	0	64	0	0	76

Figure 2 : Échantillon d'une analyse Cuckoo représentée sous forme de tableau.

3.2 Architecture de réseaux de neurones convolutifs

Au vu de nos données et de notre choix pour la transformation des données, il nous a semblé pertinent de partir sur de la convolution 1D. En effet, celle-ci est surtout utilisée pour la classification de données telle que la classification de données venant de dynamomètre, le traitement naturel du langage. D'autres types de convolution existent : la convolution 2D et la convolution 3D. Ces dernières s'effectuent, respectivement, pour la classification de données provenant d'images et pour la classification de données provenant de vidéos. L'utilisation de la convolution 3D a été directement écartée car elle ne présente aucune utilité au vu de nos données étant exclusivement textuelles. Une possibilité aurait été d'utiliser la convolution 2D. Nos données telles quelles ne se prêtent pas à l'utilisation de la convolution 2D ; il aurait été nécessaire de réaliser une matrice 2D pour chaque rapport sans certitude que cette méthode apporte une quelconque plus-value par rapport à la convolution 1D. Par ailleurs, la taille des matrices générées aurait été relativement petite et leurs dimensions auraient posé un problème, puisque la longueur aurait été bien plus importante que la largeur.

LeNet5 est une architecture de réseaux de neurones convolutifs qui a été conçu en 1998 et qui possède seulement 5 couches alternant les convolutions et la mise en commun. Une couche finale complètement connectée permet de récupérer les résultats des convolutions afin de prédire le résultat en sortie, dans notre cas le fait qu'un programme soit malicieux ou qu'il soit sain. Le schéma qui décrit cette architecture est présenté en Figure 3.

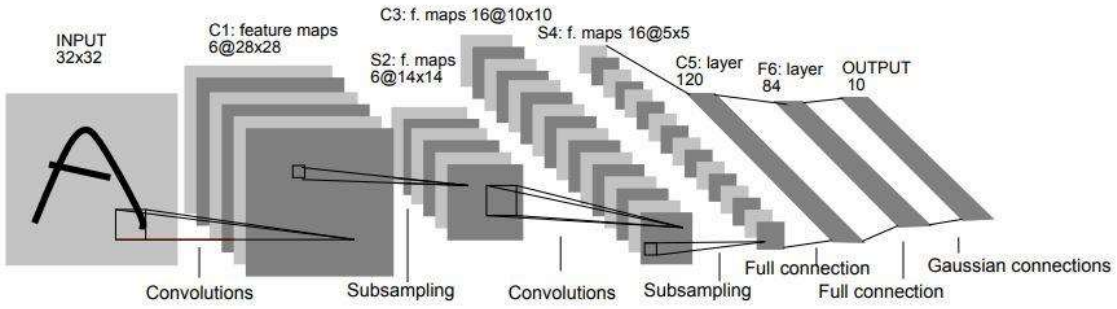

Figure 3 : Schéma de LeNet repris de sa publication d'origine.

En 2014, le Visual Geometry Group a proposé une architecture de CNN connue sous le nom de VGGnet de Simonyan & Zisserman (2014) ou plus simplement VGG. La version d'origine est composée de 16 couches (VGG16) de convolution avec 138 millions de paramètres et est schématisée sur la Figure 4.

Ces deux réseaux de neurones ont été adaptés en modifiant la couche d'entrée qui prenait un tenseur monocanal de taille 24x24 dans le cas de LeNet et un tenseur de taille 224x224 sur trois canaux dans le cas de VGG16 afin de prendre en entrée un tenseur monocanal à une dimension conformément au formatage des données expliqué précédemment. À la suite de cette modification, les couches de convolution 2D ont été remplacées par des convolution à 1D de manière analogue à ce qui se fait dans l'analyse de série temporelle. Deux variantes sont également proposées afin d'adapter ces CNN de base par rapport au problème de prédiction de ce travail :

> **VGG16*** : deux couches de convolution 1D de taille 32 avec un noyau de 3, un allongement de 1 et un remplissage de 1, une couche de mise en commun maximum, deux couches de convolutions 1D avec les mêmes paramètres que les premières couches mais de taille 64 et une couche complètement connectée.

LeNet5* : une triple séquence composée d'une couche de convolution 1D avec un noyau de 3, un allongement de 1 et un remplissage de 1, une couche de mise en commun maximum sans remplissage mais avec un allongement de 2. Les séquences passent d'une taille 32 vers une taille 64 et finalement une taille 128. La dernière couche est une couche complètement connectée.

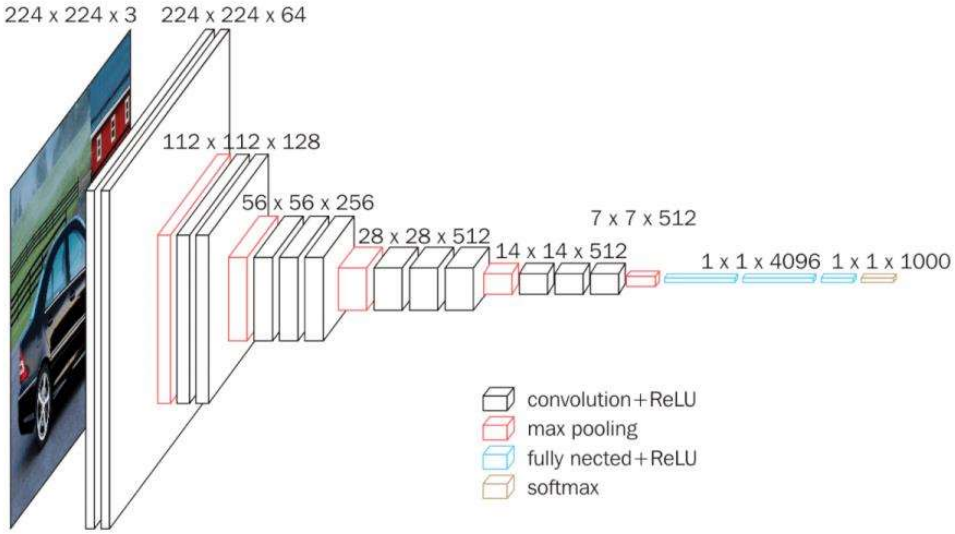

Figure 4 : Schéma de VGGnet repris de sa publication d'origine.

3.3 Entraînement

Afin d'entraîner les différentes architectures neuronales, nous avons considéré deux méthodes en tant qu'algorithme d'optimisation. La première est celle de *Adam* qui fournit rapidement un résultat acceptable et la seconde est la méthode du gradient stochastique qui nécessite un entraînement plus long mais qui peut s'évader plus facilement d'un optimum local afin d'explorer d'autres valeurs. La fonction de perte quant à elle est l'entropie croisée afin d'obtenir des résultats comparables à l'état de l'art et de passer facilement à un problème multi-étiquettes par rapport au type de maliciel identifié. Ces recommandations suivent l'analyse de Torres, Comesaña & García-Nieto (2019) lorsque l'apprentissage profond en cybersécurité est exploité.

Afin de délimiter la recherche des hyperparamètres, étant donné le nombre de combinaisons importantes, nous avons discrétisé les valeurs continues en espaçant dans un premier temps les valeurs de manière équidistante, puis en se concentrant autour des valeurs qui fournissent un meilleur résultat. Nous avons également choisi une approche itérative. En effet, un ensemble d'hyperparamètres initiaux ont été définis dans un premier temps, ensuite chacun des paramètres est changé par l'ensemble des valeurs possibles, un paramètre à la fois. Lorsqu'un paramètre a été exploré, il est considéré comme fixe afin de déterminer la meilleure valeur possible pour les autres paramètres.

Lorsque les hyperparamètres sont fixés, le jeu de données est divisé en k sous-jeux de données. Durant k itérations, toutes les données vont être utilisées afin d'entraîner le modèle excepté le sous-jeu de données qui correspond à l'itération courante. Celui-ci est utilisé comme jeu de test. A la fin de chaque itération, le modèle est évalué sur base de son taux de bonnes prédictions. La moyenne des taux de prédiction correspond à la qualité de la généralisation des données pour les hyperparamètres fixés.

4 Résultats

Les différentes architectures de CNN présentées en Section 3.2 ont été entraînées à l'aide de la descente stochastique du gradient avec un taux d'apprentissage faible. Le choix des hyperparamètres a été effectué en fonction de la précision moyenne sur les 10-validations croisées à chacune des architectures. Les courbes d'apprentissages sont assez similaires à celle de VGG16 qui est reprise sur la Figure 5.

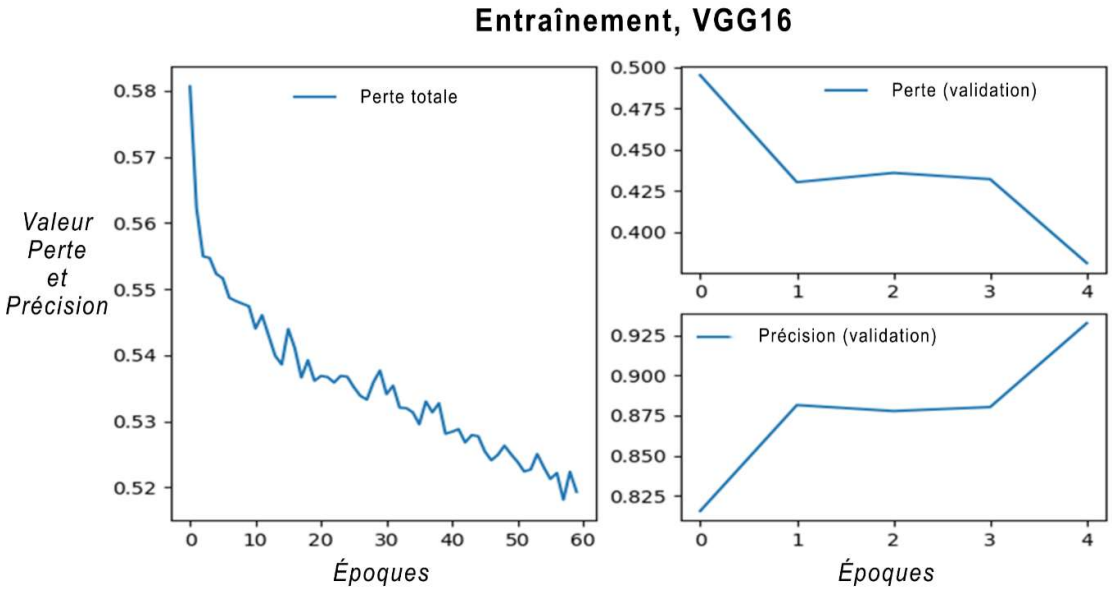

Figure 5 : Perte et précision lors de la validation croisée de VGG16. L'abscisse correspond au nombre d'époques tandis que l'ordonnée correspond aux valeurs de pertes et précisions comprises entre 0 et 1.

Les taux de prédiction sont reportés sur le Tableau 1. Les résultats de LeNet5 et VGG16 sont assez proches de l'état de l'art lors de l'utilisation des CNN 2D. Toutefois les versions transformées plus spécifiquement à la convolution 1D permettent d'atteindre un taux de prédiction proche de 83 %.

Architecture	Précision lors de l'entraînement	Précision sur le jeu de test
LeNet5	81,55 %	80,30 %
LeNet5*	81,96 %	82,04 %
VGG16	76,25 %	73,51 %
VGG16*	87,77 %	82,99 %

Tableau 1 : Précision des différents modèles après 100 époques.

5 Conclusion et discussion

Dans le cadre de notre travail, nous avons tenté de trouver une solution efficace permettant une classification entre des maliciels et des fichiers bénins. Une telle classification a pour objectif de détecter les maliciels en vue d'assurer une meilleure sécurité informatique. A l'heure où notre société s'informatise, lutter efficacement contre les logiciels malveillants relève de la priorité.

Les modèles personnalisés ont obtenu les meilleurs résultats ; le modèle personnalisé sur l'architecture VGG16 présentant une performance légèrement meilleure. En effet, pour le set de tests, le résultat issu

de ce dernier était de 91,33 % de véracité alors que celui obtenu à la suite de l'utilisation du modèle LeNet5 était de 88,68 %. Les résultats se sont donc révélés relativement similaires.

Au regard de ces résultats, l'application de réseaux de neurones convolutionnels à un problème de classification entre des maliciels et des fichiers bénins se montre efficace. Quelques petites réserves doivent toutefois être émises : un certain nombre de maliciels ne sont pas correctement classifiés et se retrouvent dans la catégorie des fichiers bénins. Cette constatation peut bien entendu se révéler très problématique en pratique. Pour pallier ce problème, d'aucuns pourraient analyser les maliciels se retrouvant dans la catégorie des fichiers bénins en vue de déterminer une caractéristique ayant été ignorée par les modèles que nous avons sélectionnés (un processus similaire pourrait être réalisé pour les fichiers bénins) et recommencer le processus en tenant compte de cette caractéristique. En tout état de cause, l'intelligence artificielle peut se révéler particulièrement utile à la détection de maliciels.

Les données ne possédant pas naturellement une structure spatiale, nous nous sommes concentrés sur la convolution 1D. Toutefois les maliciels plus évolués peuvent être morcelés et distribués au sein de plusieurs morceaux logiciels. Ceux-ci effectuent des demandes aux systèmes d'exploitation afin de mener à bien leurs actions malveillantes, en traitant le système d'exploitation comme une ressource commune. Lerat, Han & Lenaerts (2013) ont montré l'importance de la coopération lorsqu'il s'agit de partager une ressource commune et cela nous permet d'envisager de mettre en avant ces liens coopératifs. A l'aide de ces liens, une représentation spatiale est possible permettant ainsi de tirer profit de la convolution 2D. Afin de la mettre en œuvre, il faut effectuer des transformations et découper le vecteur sous forme de lignes x colonnes sur plusieurs canaux. Les différents canaux peuvent aisément être représentés par les différents axes sur lesquels travaillent les exécutables et plus spécifiquement les maliciels. En effet lorsqu'un maliciel exploite une connexion Internet, il se peut qu'il utilise également les fichiers. Le rançongiciel est un exemple : une partie de son code chiffre les fichiers pendant que les clefs de chiffrement sont envoyées sur des serveurs distants. Cependant, transformer un vecteur sous forme de matrice est plus complexe. Le vecteur doit être découpé afin de former différentes lignes mais à l'endroit de coupe, les données vont être spatialement espacées alors que les couches de convolution travaillent sur des zones locales de l'image.

Une alternative à la convolution 2D est d'étudier les interactions entre le maliciel et le système comme une séquence d'événements, ce qui en fait une série temporelle. Différents types de réseaux peuvent alors être exploités comme les réseaux *Long-Short-Term memory*. Un réseau de neurones capable de prédire la prochaine action d'un exécutable permet de prévenir l'exécution de code malicieux avant que celui-ci en ait l'opportunité.

Remerciements

Pour mener à bien cette étude la Haute École en Hainaut a bénéficié d'un subside de la Fédération Wallonie-Bruxelles (JCM/TP/BS/mo/c999). L'auteur remercie le service ILIA de l'université de Mons qui a fourni des jeux de données ainsi que des commentaires qui ont permis d'améliorer la qualité du travail.

Références bibliographiques

Dua, S. & Du, X. (2016). Data mining and machine learning in cybersecurity. S.l. : CRC press.

Hasegawa, C. & Iyatomi, H. (2018). One-dimensional convolutional neural networks for android malware detection. In *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), 9-10 March 2018, Penang, Malaysia,* 99-102, doi: 10.1109/CSPA.2018.8368693.

Ijaz, M., Durad, M. H. & Ismail, M. (2019). Static and dynamic malware analysis using machine learning. *In 16th International bhurban conference on applied sciences and technology (IBCAST) 8-12 Jan. 2019*, Islamabad, Pakistan, 687-691, doi: 10.1109/IBCAST.2019.8667136

Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016). Deep learning for classification of malware system call sequences. *In 29th Australasian Joint Conference on Artificial Intelligence*, *Springer, Cham*, 137-149.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11), 2278-2324.

Lerat, J. S., Han, T. A. & Lenaerts, T. (2013). Evolution of common-pool resources and social welfare in structured populations. *In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 3-9 August 2013, Beijing, Chine, 2848-2854. Issu de : https://www.ijcai.org/Proceedings/13/Papers/419.pdf (consulté le 16/01/22).

Polychronakis, M., Anagnostakis, K. G. & Markatos, E. P. (2010). Comprehensive shellcode detection using runtime heuristics. *In Proceedings of the 26th Annual Computer Security Applications Conference*, 287-296. doi.org/10.1145/1920261.1920305

Preda, M. D., Christodorescu, M., Jha, S. & Debray, S. (2007). A semantics-based approach to malware detection. *ACM SIGPLAN Notices* 42(1), 377-388.

Rathore, H., Agarwal, S., Sahay, S. K. & Sewak, M. (2018). Malware detection using machine learning and deep learning. *In* 6th International Conference, BDA 2018, Warangal, India, December 18–21, 402-411, doi: 10.1007/978-3-030-04780-1_28.

Sahay, S. K., & Rathore, H. (2018). Comparison of deep learning and the classical machine learning algorithm for the malware detection. *In 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 293-296, doi: 10.1109/SNPD.2018.8441123.

Sgandurra, D., Muñoz-González, L., Mohsen, R. & Lupu, E. C. (2016). *Automated dynamic analysis of ransomware: Benefits, limitations and use for detection*. arXiv preprint arXiv:1609.03020.

Simonyan, K. & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv:1409.1556.

Torres, J. M., Comesaña, C. I. & García-Nieto, P. J. (2019). Machine learning techniques applied to cybersecurity. *International Journal of Machine Learning and Cybernetics* 10(10), 2823-2836.

Wannacry ransomware: recent cyber-attack. (2017). Issu de https://www.europol.europa.eu/newsroom/news/wannacry-ransomware-recent-cyber-attack (consulté le 19/11/2020).