Synthèse PAF
Détection d'intrusion et contre-attaque

Julien Béguinot, Arthur Biapo, Victor Masiak, Zirui Peng, Quentin Sopheap, Huidi Zhu,
encadrés par Thomas Robert

6 juillet 2021

Table des matières

1 Introduction **4**
1.1 Quelques définitions . . . 4
1.2 Les différents types d'attaques . . . 4
1.3 Objectifs . . . 4

2 Les NIDS **5**
2.1 Principe de fonctionnement . . . 5
2.2 Evaluation des performances . . . 5
2.3 Attaques détectées par un NIDS . . . 6

3 L'apprentissage : notions mathématiques **7**
3.1 Théorie de la décision, du vocabulaire . . . 7
3.2 Position du problème et encore du vocabulaire . . . 7
3.3 Prédicteur . . . 7
3.4 Fonction de coût . . . 7
3.5 Rappel de MDI104 : Lois conditionelles . . . 7
3.6 Risque associé . . . 8
3.7 Comment estimer ce risque? . . . 8
3.8 Exemple de la régression linéaire . . . 8
3.9 Décomposition du risque . . . 8
3.10 Biais Variance pour l quadratique . . . 8
3.11 Illustration de l'overfitting (sur-apprentissage) . . . 9
3.12 Mais attention à l'underfitting (sous-apprentissage) . . . 9
3.13 La régularisation de ridge (de Tikhonov) . . . 10
3.14 estimateur et paradoxe de Stein : Une justification du 'shrinkage' . . . 10

4 Algorithmes considérés **11**
4.1 Algorithmes non supervisés . . . 11
4.1.1 K-means . . . 11
4.1.2 DBscan . . . 11
4.1.3 Auto-encodeur . . . 12
4.2 Algorithmes supervisés . . . 13
4.2.1 Decision Tree . . . 13
4.2.2 Réseau de neurones artificiel . . . 13

5 Études des bases de données pour les IDS **14**
5.1 The different types of data used by the ML/DM approaches . . . 14
5.2 Structure du dataset . . . 14
5.3 Structure du dataset . . . 14
5.4 Les 4 types d'attaques du dataset . . . 14
5.5 Améliorations par rapport au dataset KDD 99 . . . 15

6 Qu'est-ce qu'un GAN? **16**
6.1 Position du problème . . . 16
6.2 Intuition . . . 16
6.3 Formellement . . . 17
6.4 Problème de convergence . . . 17
6.4.1 Le gradient évanescent . . . 17
6.4.2 Le "mode collapse" . . . 17

6.4.3 Qu'est-ce qui converge 18

7 La distillation vers des réseaux plus résistants aux exemples adversariaux? 19
7.1 Position du problèmes : attaques adversariales 19
7.2 Distillation 19
7.3 La méthode CW 20

8 Expérimentations 21
8.1 Détails techniques pour les exécutions à distance 21
8.2 IDS-WGAN 21
8.2.1 Comparaison des dépôts 21
8.2.2 Exécution des codes 22
8.2.3 Modifications des paramètres 23
8.2.4 Intégration d'un decision tree 24
8.3 Expérimentation sur les décisions trees 24
8.3.1 Contextualisation 24
8.3.2 Résultat observé sans sélection des features 24
8.3.3 Utilités des features 26
8.3.4 Résultats avec sélection des features 28
8.4 Exploration de voisinages 30
8.4.1 Exploration des voisinages d'une attaque adversariale 30
8.4.2 Observation de la boule formées par ces attaques adversariales 31

9 Conclusion et prolongement 33

1 Introduction

Avec l'augmentation exponentielle du nombre de systèmes d'informations, la sécurité de ces derniers est devenue un enjeu capital. En particulier, le nombre d'attaques a fortement bondi durant les dernières années. Il est donc devenu important de savoir détecter et réagir rapidement et efficacement à ces attaques. Alors qu'auparavant, des méthodes reposant sur des reconnaissances de signature ou de types de comportement étaient fortement utilisées, elles ne suffisent plus aujourd'hui. L'essor des algorithmes de Deep Learning et de Machine Learning a donné de nouvelles manières de reconnaître les attaques. Depuis de nombreuses années, la détection d'intrusion repose sur l'apprentissage automatique à partir des données représentant des échanges réseau valides ou malicieux (émis par des attaquants).

1.1 Quelques définitions

Risque : l'exposition accidentelle ou imprévisible d'informations, ou la violation de l'intégrité des opérations en raison du mauvais fonctionnement du matériel ou de la conception incomplète ou incorrecte des logiciels

Pénétration : une attaque réussie - la capacité d'obtenir un accès non autorisé (non détecté) à des fichiers et programmes ou l'état de contrôle d'un système informatique

Attaque : formulation ou exécution spécifique d'un plan de mise en œuvre d'une menace

Vulnérabilité : faille connue ou suspectée dans le matériel ou le logiciel ou le fonctionnement d'un système qui expose le système à la pénétration ou ses informations à une divulgation accidentelle

Menace : danger possible qui pourrait exploiter une vulnérabilité pour enfreindre la sécurité et donc causer un préjudice éventuel

Pare-feu : dispositif de sécurité des réseaux qui surveille le trafic réseau entrant et sortant et permet ou bloque les paquets de données sur la base d'un ensemble de règles de sécurité

1.2 Les différents types d'attaques

— **Confidentialité** : l'agresseur a accès à des données confidentielles et inaccessibles par ailleurs
— **Intégrité** : l'attaquant peut modifier l'état du système et altérer les données sans autorisation appropriée du propriétaire
— **Disponibilité** : le système est soit fermé par l'attaquant, soit rendu indisponible pour les utilisateurs généraux.
— **Contrôle** : l'attaquant obtient le contrôle total du système et peut modifier les privilèges d'accès au système, déclenchant ainsi potentiellement les trois attaques ci-dessus

1.3 Objectifs

Pour ce PAF, nous essayerons d'implémenter un IDS *(Intrusion Detection System)* et de le confronter GAN *(Generatif Adversarial Network)*. Pour cela, nous utiliserons la base de données NSL-KDD pour entraîner notre GAN.

2 Les NIDS

2.1 Principe de fonctionnement

En particulier, nous nous intéressons aux **NIDS**. Un **NIDS** (Network Intrusion Detection System) vise à détecter les intrusions possibles telles que les activités malveillantes, l'utilisation abusive d'un ordinateur, la propagation d'un virus et à alerter les personnes concernées dès leur détection. Le NIDS prend des mesures réactives, comme la surveillance et l'analyse des paquets de données qui circulent sur un réseau à la recherche de telles activités suspectes, et peut signaler l'itinéraire emprunté par le paquet d'attaque. Un serveur NIDS peut être installé sur un lien d'un réseau fédérateur ou sur un serveur, un commutateur, une passerelle ou un routeur.

2.2 Evaluation des performances

L'IDS peut généralement être évalué de deux points de vue :

— **Rentabilité** : les ressources à allouer au système, y compris les cycles de l'unité centrale et la mémoire principale.

— **Efficacité** : la capacité du système à distinguer les activités intrusives des activités non intrusives.

Nous évaluons couramment les performances d'un NIDS en mesurant la précision et l'efficacité de l'IDS en termes de taux de faux positifs et de pourcentage d'attaques détectées avec succès. Les mesures de l'efficacité peuvent être divisées en trois catégories : seuil, classement et probabilité.

Les mesures de seuil comprennent le taux de classement (CR), la mesure F (FM) et le coût par exemple (CPE), etc. La proximité d'une prédiction par rapport à un seuil n'a pas d'importance, seulement si elle se situe au-dessus ou en dessous du seuil. Les mesures de classement comprennent le taux de faux positifs (FPR), le taux de détection (DR), la précision (PR), l'aire sous la courbe (AUC) et la capacité de détection des intrusions (CID, c'est une mesure basée sur la théorie de l'information qui permet une meilleure comparaison des différents IDS). Les mesures de probabilité comprennent l'erreur quadratique moyenne (EQM)

Toutes ces mesures se situent entre 0 et 1.

En général, nous calculons ces mesures à partir de la matrice de confusion, qui est une matrice qui représente le résultat de la classification, qu'il soit vrai ou faux :

— **Vrai positif (TP)** : Intrusions détectées avec succès par l'IDS

— **Faux positif (FP)** : Comportement normal/non intrusif qui est classé à tort comme intrusif par l'IDS

— **Vrai négatif (TN)** : Comportement normal/non intrusif qui est qualifié avec succès de normal/non intrusif par l'IDS

— **Faux négatif (FN)** : Intrusions manquées par l'IDS et classées comme normales/non intrusives

Pour citer quelques métriques de performance usuelles :

— L'accuracy est le ratio des résultats correctement identifiés

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

— Le Taux de Vrais Positifs(TVP)est la proportion de positifs qui sont correctement détectés

$$TVP = \frac{TP}{TP + FN}$$

— Le Taux de Faux Positifs(TFP) est la proportion de négatifs incorrectement détectés comme des positifs.

$$TFP = \frac{FP}{TN + FP}$$

— La précision est la proportion de prédictions de positifs qui sont en effet des positifs.

$$Precision = \frac{TP}{TP + FP}$$

— Le F1 score est la moyenne harmonique de la précision et du TVP

$$F1\ score = 2 \times \frac{Precision \times TVP}{Precision + TVP}$$

2.3 Attaques détectées par un NIDS

Scanning attacks L'attaquant envoie différents types de paquets pour sonder un système ou un réseau à la recherche d'une vulnérabilité pouvant être exploitée. Lorsque les paquets de sondage sont envoyés, le système cible répond ; les réponses sont analysées pour déterminer les caractéristiques du système cible et s'il existe des vulnérabilités. Ainsi, l'analyse d'une attaque identifie essentiellement une victime potentielle et des informations telles que la topologie du réseau, le type de pare-feu, l'identification des hôtes qui répondent, les systèmes d'exploitation et les applications serveur en cours d'exécution. La signature NIDS peut être conçue pour identifier une telle activité d'analyse malveillante à partir d'une activité d'analyse légitime avec un degré de précision assez élevé.

Denial of Service (DoS) Une attaque par déni de service tente de ralentir ou de fermer complètement une cible afin de perturber le service et de refuser l'accès aux utilisateurs légitimes et autorisés. Il en existe deux types :

- **Les attaques par exploitation de faille DoS** : un attaquant exploite une faille dans le logiciel du serveur pour le ralentir ou l'épuiser de certaines ressources. L'attaque Ping of death est une de ces attaques bien connues, qui vise à envoyer un ping malformé ou autrement malveillant à l'ordinateur.
- **Denial of Service (DoS)** : un attaquant envoie simplement plus de requêtes à une cible qu'il ne peut en traiter. ces attaques peuvent soit épuiser la capacité de traitement de la cible, soit épuiser la bande passante du réseau de la cible. une version plus dangereuse de l'attaque DoS est appelée attaque par déni de service distribué (DDoS), qui utilise un grand nombre d'hôtes pour cibler un hôte victime donné

Attaques de pénétration Un attaquant obtient un contrôle non autorisé d'un système et peut modifier/altérer l'état du système, lire des fichiers, etc. En général, ces attaques exploitent certaines failles du logiciel, ce qui permet à l'attaquant d'installer des virus et des logiciels malveillants dans le système. Les types les plus courants :

- **User to Root (U2R)** : un utilisateur local obtient l'accès complet à chaque composant du système.
- **Remote to User (R2U)** : un utilisateur du réseau obtient un compte d'utilisateur et les contrôles associés.
- **Remote disk read** : un attaquant sur le réseau obtient l'accès aux fichiers inaccessibles stockés localement sur l'hôte.
- **Remote disk write** : un attaquant sur le réseau a non seulement accès aux fichiers inaccessibles stockés localement sur l'hôte, mais peut également les modifier.

3 L'apprentissage : notions mathématiques

Toute cette partie présente les notions mathématiques relatives à l'apprentissage. Elle est principalement tiré de notre compréhensions de [9].

3.1 Théorie de la décision, du vocabulaire

Théorie de la décision : Choix de critères pour évaluer la qualité d'une décision
But de l'Apprentissage : C'est choisir une "meilleure" décision δ en fonction de la donnée x en vu de y (non connu au moment de la décision.) Attention Y n'est en générale pas fonction de X d'où la difficulté. Rechercher la meilleure solution en moyenne et le but de l'apprentissage.

3.2 Position du problème et encore du vocabulaire

- On dispose d'observations $(x_1, x_2, ..., x_n)$ (apprentissage non supervisé) ou d'observations $((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$ (apprentissage supervisé)
- Apprendre serait réussir a faire des prédiction **qui se généralise** à partir de ces prédictions.
- **Régresseur, descripteurs, Covariables (features)** sont les noms donnés aux X_i
- **Étiquette (label)** est le nom donné aux Y_i

3.3 Prédicteur

- On considère $\mathscr{X}$ l'espace des données des descripteurs.
- On considère $\mathscr{Y}$ l'espace des donnés de sorties (étiquettes)
- On considère $\mathscr{A}$ l'ensemble des actions/prédictions possibles.
- $Apprentissage : D_n \in \bigcup_{n\in\mathbb{N}} (\mathscr{X} \times \mathscr{Y})^n \longrightarrow \hat{f} \in \mathscr{F}(\mathscr{X}, \mathscr{A})$
- Ceci est la phase d'apprentissage.

3.4 Fonction de coût

Soient $l : (a, y) \in \mathscr{A} \times \mathscr{Y} \longrightarrow coût \in \mathbb{R}$ une **fonction de coût** qui spécifie le coût à payer pour avoir pris la décisions a alors que y se produit et $f : x \in \mathscr{X} \longrightarrow a \in \mathscr{A}$ un **prédicteur**.

- Pour une classsification : indicatrice de la classe
- Pour une regression au sens des moindres carrés : la norme l^2

3.5 Rappel de MDI104 : Lois conditionelles

Soit X,Y deux v.a.r.

- Si pour toutes applications boréliennes g,h $\mathbb{E}[g(X)h(Y)] = \int g(x)[\int h(y)d\mathbb{P}_{Y|X=x}(y)]d\mathbb{P}_X(x)$ alors $\mathbb{P}_{Y|X=x}$ est la loi conditionelle de Y | X=x.
- $\phi(x) = \mathbb{E}[Y|X = x]$ est l'espérance de la loi ci-dessus.
- $\mathbb{E}[Y|X] = \phi(X)$ l'espérance conditionelle est une variable aléatoire.
- En particulier $\mathbb{E}[Y] = \mathbb{E}[\mathbb{E}[Y|X]]$ -espérance totale-
- Et $Var(Y) = \mathbb{E}[Var(Y|X)] + Var(\mathbb{E}[Y|X])$ -variance totale-

3.6 Risque associé

— Soit $f : x \in \mathscr{X} \longrightarrow a \in \mathscr{A}$ un **prédicteur**.
— On pose le risque associé a ce problème $\mathscr{R}(f) = \mathbb{E}[l(f(X), Y)]$
— Le **prédicteur de Bayes** est alors $f^* = \underset{f \in \mathscr{F}}{argmin} \mathscr{R}(f)$
— Le risque issu d'une règle d'apprentissage est alors la variable aléatoire $\mathscr{R}(\hat{f}) = \mathbb{E}[l(f(X), Y)|D_n]$

3.7 Comment estimer ce risque ?

— **Problème :** On ne connaît justement pas les lois X,Y alors comment estimer ce risque ? On se limite au **risque empirique** qui est la moyenne des coûts sur l'ensemble d'entraînement.
— **Nouveau Problème :** Le prédicteur de Bayes n'est plus pertinent car il n'est définis que sur les regresseurs. On se limite donc a un ensemble S de fonctions "pertinentes" appelés **espace d'hypothèse**

3.8 Exemple de la régression linéaire

— Par exemple pour une régression linéaire avec $\mathscr{X} = \mathbb{R}^p$ et $\mathscr{Y} = \mathbb{R}$ on se limite a l'ensemble des formes linéaires sur R^p.

$$S = \{f_w : x \in \mathbb{R}^p \longrightarrow w^\top x \in \mathbb{R} | w \in \mathbb{R}^p\}$$

— On s'intéresse donc à la **meilleure approximation**

$$f_S^* = \underset{f \in S}{argmin} \mathscr{R}(f)$$

— On appel X la **matrice de design**
— Ce problème est résolu avec l'**équation normal** $X^\top X w = X^{top} y$

3.9 Décomposition du risque

$$\underbrace{\mathscr{R}(\hat{f}_S) - \mathscr{R}(f^*)}_{excès de risque} = \underbrace{\mathscr{R}(\hat{f}_S) - \mathscr{R}(f_S^*)}_{erreur d'estimation} + \underbrace{\mathscr{R}(f_S^*) - \mathscr{R}(f^*)}_{erreur d'approximation}$$

3.10 Biais Variance pour l quadratique

$$\mathbb{E}[\mathscr{R}(\hat{f})] = \underbrace{\mathbb{E}[(\hat{f}(X) - \mathbb{E}[\hat{f}(X)|X])^2]}_{variance de \hat{f}} + \underbrace{\mathbb{E}[(\mathbb{E}[\hat{f}(X)|X] - \mathbb{E}[Y|X])^2]}_{biais de \hat{f}} + \underbrace{\mathbb{E}[(Y - \mathbb{E}[Y|X])^2]}_{variance du "bruit"}$$

— La variance est sensible aux petites fluctuations (sur-apprentissage ?)
— Le biais vient du choix de S (sous-apprentissage ?)
— Le dernier terme est le bruit.

3.11 Illustration de l'overfitting (sur-apprentissage)

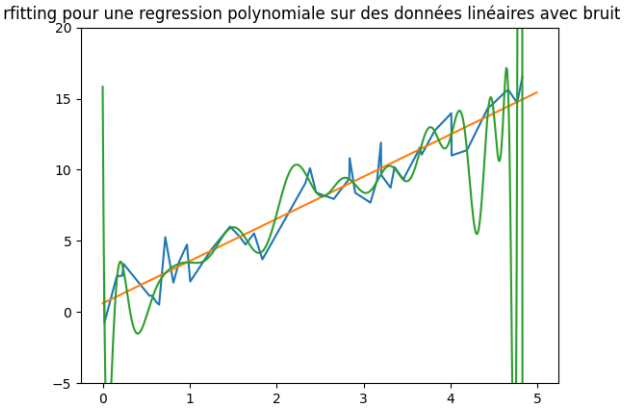


FIGURE 1 – Illustration de l'overfitting

3.12 Mais attention à l'underfitting (sous-apprentissage)

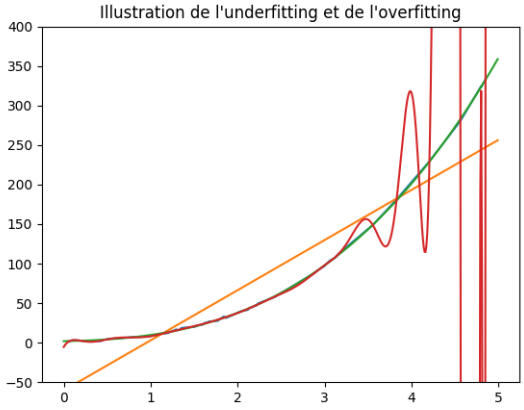


FIGURE 2 – Illustration de l'underfitting

3.13 La régularisation de ridge (de Tikhonov)

Pour éviter le sur-apprentissage on se propose de modifier légèrement le risque. On pose :

$$\mathscr{R}_{n,ridge}(f) = \mathscr{R}_n(f) + \lambda \left\| f \right\|^2$$

Dans la pratique il faut donc ajuster ce paramètre. On rajoute donc un deuxième jeu de données supplémentaire (cross validation set). Si on augmente λ le biais diminue mais la variance augmente et inversement. L'idée est que en rajoutant ce terme dépendant de la norme de f on pénalise la complexité de f et donc l'overfitting.

3.14 estimateur et paradoxe de Stein : Une justification du 'shrinkage'

A priori ce point n'est **pas essentiel** mais (je trouve) très intéressant (plutôt contre-intuitif).

On pose **l'estimateur de Stein** pour l'espérance μ d'une v.a. X :

$$\mu^{JS} = (1 - \frac{\sigma^2(p-2)}{\|X\|^2})X$$

Pour $p \geq 3$ l'excès de risque de l'estimateur de James-Stein est strictement inférieur à la variance de bruit ($= p\sigma^2$)

Plus fort, on peut montrer que elle est inférieur à $\sigma^2 min(p, 2 + \frac{\|\mu\|^2}{\sigma^2})$

Cela justifie l'usage de régularisation. Par rapport à l'estimateur de maximum de vraisemblance on augmente le biais (qui était nul) pour diminuer la variance.

4 Algorithmes considérés

4.1 Algorithmes non supervisés

4.1.1 K-means

Cette méthode d'apprentissage est basée sur le clustering.L'objectif est de trouver des patterns dans le jeu de données et de les regrouper en cluster. L'algorithme K-means regroupe les données en K cluster donc cela suppose qu'il faut avoir une idée du nombre de clusters à l'avance.Un point important dans cet algorithme est le centroïde qui représente un point du jeu de données que l'on choisira comme centre du cluster.C'est en fonction de la distance au centroïde que nous définirons l'appartenance à un cluster.

Plus formellement cet algorithme consiste à choisir K centroïdes aléatoirement parmi la base de donnée à l'initialisation. Puis à chaque itération d'associer chaque point de la base de donné a un de ces centroïdes. Les centroïdes sont alors mis à jours en devenant le barycentre de l'ensemble des points qui lui ont été associé. D'où ce nom de K-mean.

— Soit K le nombre de classe recherché
— Soit N-itération le nombre d'itérations.
— Soit DB la base de donnée.
— On note μ_i le i-ème centroïde

Initialiser aléatoirement les K centroïdes par des valeurs de la base de donnée.
for *i allant de 1 à N-itération* **do**
  forall *intrusion dans la base de donnée* **do**
    Affecter a intrusion le centroïde le plus proche.
  end
  forall *centroïde* **do**
    centroïde ⟵ barycentre des intrusions qui lui sont affectés.
  end
end

Algorithm 1: Pseudo-code : K-mean

4.1.2 DBscan

Cette méthode de clustering dite de densité est fondée sur une estimation de la densité nécessaire aux clusters pour le partitionnement.L'algorithme prend en compte une valeur ϵ considérée comme le rayon du voisinage et le nombre minimum de points présent dans ce voisinage MinPts.DBscan commence par un point de donnée arbitraire, il calcule le $\epsilon - voisinage$ puis si ce voisinage contient MinPts il calcule les $\epsilon - voisinages$ de chacun des points et ainsi de suite jusqu'à ne plus pouvoir agrandir le cluster.Si le point considéré ne dispose pas assez de voisins il sera considéré comme du bruit.Cela permet à DBscan d'être robuste aux données bruitées car le mécanismes les isole.

source :`https://github.com/jeremy191/clustering-based-anomaly-detection/blob/master`

4.1.3 Auto-encodeur

L'Auto-encodeur est un type de réseau de neurones. Son but est de reconstruire le vecteur d'entrée à sa sortie. En conséquence, la structure de l'auto-encodeur a les caractéristiques suivantes :

— Le nombre de neurones de la couche d'entrée doit être égale à la couche de sortie.
— Le nombre de neurones de la couche cachée est moins que la couche d'entrée.
— Parfois, pour éviter l'overfitting, nous ajoutons le drop-out [1] ou des bruits sur la couche d'entrée.

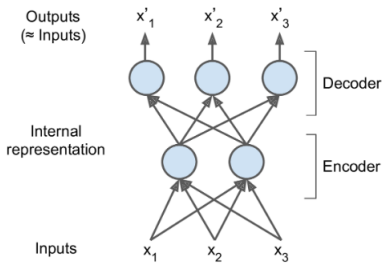


FIGURE 3 – Structure d'un auto-encodeur

L'auto-encodeur cherche des relations entre les attributs du vecteur d'entrée et réduit la dimension de ce vecteur d'entrée (codeur). Puis à partir des relations qu'il apprend, il est capable de reconstruire le vecteur d'entrée (décodeur).

L'auto-encodeur est souvent utilisé comme un IDS. En entraînant un système qui ne sais que reconstruire les données normales, nous distinguons les attaques des données normales.
Source : `https://github.com/r7sy/IntrusionDetection`

1. Le drop est une méthode qui consiste a oublier aléatoirement un neurone.

4.2 Algorithmes supervisés

4.2.1 Decision Tree

On utilise Une structure d'arbre qui représente les conjonctions des attributs conduisant aux différentes classifications. A chaque noeud, on choisit les attributs qui conduisent à un gain d'information maximal comme attribut du noeuds. On utilise des valeurs de seuils pour séparer l'espace en ses classes.
Source : `https://github.com/CynthiaKoopman/Network-Intrusion-Detection`

4.2.2 Réseau de neurones artificiel

Un réseau de neurones artificiel est constitué de couches dont chaque couche est constituée de neurones. Le neurone est conçu comme un automate doté d'une fonction de transfert qui transforme ses entrées en sortie selon des règles précises. Normalement, la fonction de transfert est de la forme $y = g(w * x + b)$ dont y est le vecteur de sortie, x est le vecteur d'entrée, w est une forme linéaire et g est la fonction d'activation qui est non-linéaire.

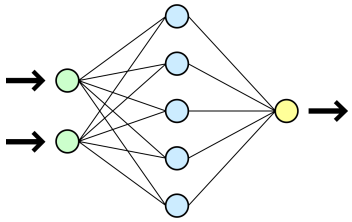


FIGURE 4 – Structure d'un réseau de neurones

Pour entraîner un réseau de neurones artificiel, nous devons faire varier les paramètres de chaque fonction de transfert afin d'obtenir un système optimisé. Par conséquent, nous définissons une fonction de perte dont la valeur indique la distance entre le résultat vrai et la sortie du réseau. En calculant le gradient local de cette fonction, on approche pas à pas le point optimal (la valeur de la fonction de perte à ce point est la valeur minimale).

La rétro-propagation consiste a propager ce gradient dans le réseau en comparant la valeur prédite au label de la base de donnée. Cela se justifie en terme de calcul différentiel par la règle de la chaîne.

5 Études des bases de données pour les IDS

5.1 The different types of data used by the ML/DM approaches

- Packet level data : User's program run the TCP/UDP/ICMP/IGMP protocols which generate the network traffic of the Internet. The API pcap could capture these network traffic package at the physical level.
- Network flow : A network flow is a unidirectional sequence of packets that share the exact same seven packet attributes : ingress interface, source IP address, destination IP address, IP protocol, source port, destination port, and IP type of service. It's a compressed and preprocessed version of the actual network packets.
- Public data set :
  - DAPRA1998, DAPRA1999 : Based on TCP/IP network data, Solaris Basic Security Module log data, and Solaris file system dumps for user and root
  - KDD199 : The data set has three components—basic, content, and traffic features—making for a total of 41 attributes. This dataset has certain limites (dropped packets, redundant data etc)
  - NSL-KDD : Consists of selected records of the complete KDD data set and does not experience the shortcomings.

5.2 Structure du dataset

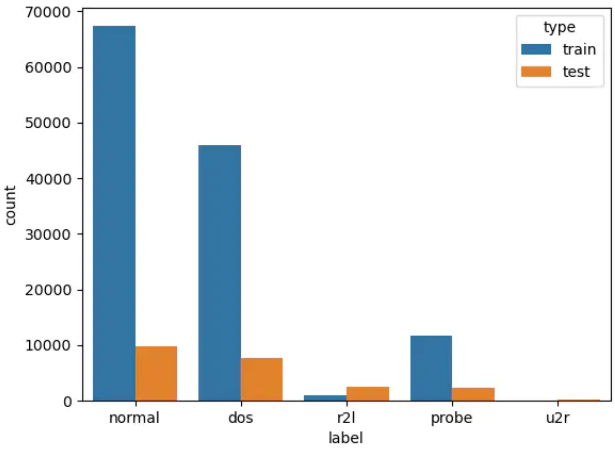
Le dataset NSL-KDD est en fait constitué de plusieurs fichiers :
- Des fichiers d'entraînement :
  - KDDTrain+.ARFF
  - KDDTrain+.TXT
  - KDDTrain+_20Percent.ARFF
  - KDDTrain+_20Percent.TXT
- Des fichiers de test :
  - KDDTest+.ARFF
  - KDDTest+.TXT
  - KDDTest-21.ARFF
  - KDDTest-21.TXT

5.3 Structure du dataset

- Les fichiers de type x_20Percent contiennent 20% du fichier x associé.
- Les fichiers .TXT sont au format CSV, les fichiers .ARFF au format ARFF.
- Chaque cas a été marqué comme attaque ou non-attaque.
- Dans chaque cas on peut extraire 41 attributs.
- Les données sont classifiées en 21 niveaux de difficulté.

5.4 Les 4 types d'attaques du dataset

- DOS (Denial Of Service) : Utilisation excessive des ressources de la victime, l'empêchant ainsi de traiter les requêtes légitimes.
- Probing : Attaque dont l'objectif est de récolter des informations sur la victime.
- U2R (User To Root) : Accès non autorisée à l'utilisateur root, par un utilisateur normal du système.
- R2L (Remote To Local) : Accès non autorisé depuis une autre machine.
- Chacun de ces 4 types d'attaques est subdivisé en classes plus précises (39 au total).



5.5 Améliorations par rapport au dataset KDD 99

— Pas de cas redondants. Donc le classificateur ne tend pas vers les cas plus fréquents.
— Dans l'ensemble de test, il n'y a pas de cas répétitif. Donc le taux de détection est plus précis.
— Un ratio train/test plus raisonnable.
— ...

6 Qu'est-ce qu'un GAN?

6.1 Position du problème

Qu'est ce qu'un GAN?

Un GAN -**G**enerative **A**dversarial **N**etwork- a pour but de générer de "fausses observations" d'une distribution cible p^* non connue **pour laquelle seul des exemples positifs sont disponibles**. En général, on suppose que la dimension d des exemples est très grande.

6.2 Intuition

Un GAN se distingue généralement en deux parties.
— Une famille de générateurs $\mathcal{G} = (G_\theta)_{\theta \in \Theta}$
— Une famille de discriminateur $\mathcal{D}$
Intuitivement, le but du générateur est de créer de fausse observations. Le but du discriminateur est de de déterminer si on lui soumet un vrai exemple ou une création du générateur. En améliorant ces deux parties conjointement le générateur va s'améliorer de plus en plus jusqu'à donner des "faux" convaincants (et le discriminateur va être de plus en plus exigeant).

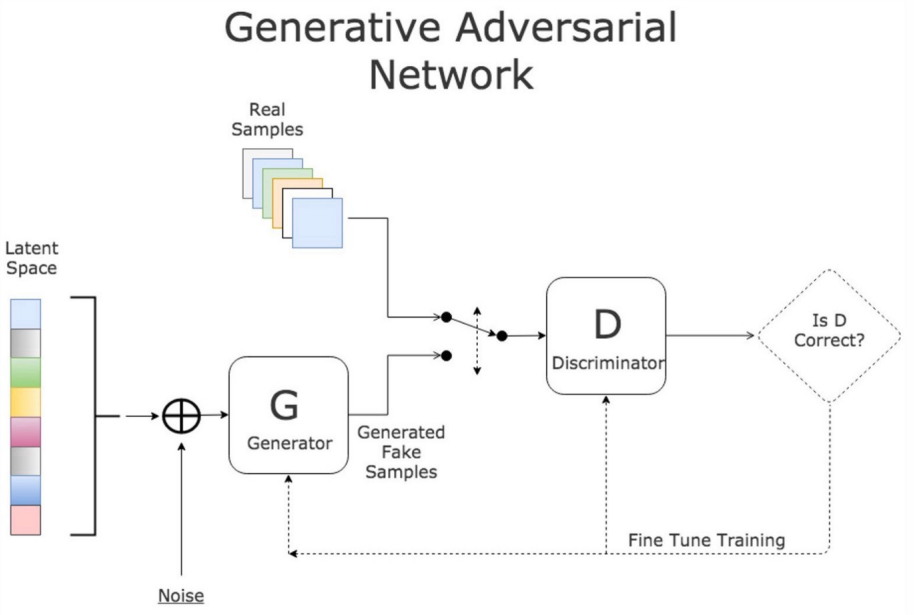


FIGURE 5 – Principe de fonctionnement du GAN tiré de : "GANs from Scratch 1 : A deep introduction. With code in PyTorch and TensorFlow"

6.3 Formellement

On considère E un borélien de $\mathbb{R}^d$. On considère μ une mesure sur E. On considère Z une v.a. de "bruit" généralement un vecteur gaussien de dimension $d' << d$. Un générateur est une application qui crée un"faux" à partir du bruit.

$$Generateur : \begin{cases} \mathbb{R}^{d'} \longrightarrow \mathbb{R}^d \\ Z \longrightarrow Fake \end{cases}$$

On dispose donc naturellement d'une famille de probabilité issue de cette famille de générateur : $\mathscr{P} = (P_\theta)_{\theta \in \Theta}$ telle que $G_\theta(Z) \overset{\mathscr{L}}{=} p_\theta d\mu$

Un discriminateur est une application borélienne de E dans [0,1]. On veut que D(x)=1 si x est bien une observation issue de p^*.

Chacun de ces p_θ est un candidat potentiel pour p^*. On **ne suppose pas** que $p^* \in \mathscr{P}$.
Il existe de nombreuses variantes de la fonction de coût qui ont été proposé. Cependant la plus utilisé (Celle proposé dans l'article de Goodfellow et Al 2014) est :

$$\hat{L} : \begin{cases} \Theta \times \mathscr{D} \longrightarrow \mathbb{R} \\ (\theta, D) \longrightarrow \sum\limits_i ln(D(X_i)) + ln(1 - D(G_{theta}(Z_i))) \end{cases}$$

Soit en version non-empirique,

$$L : \begin{cases} \Theta \times \mathscr{D} \longrightarrow \mathbb{R} \\ (\theta, D) \longrightarrow \int ln(D) p^* d\mu + \int ln(1 - D) p_\theta d\mu \end{cases}$$

6.4 Problème de convergence

Mais est-ce que c'est p_θ converge effectivement vers le distribution cible? Et bien en fait on ne sait toujours pas très bien ce qui ce passe ni pourquoi il y a convergence. C'eest un chalp actif de recherche. On dispose cependant de certains élément de réponses. Il est en effet facile pour le GAN de ne pas converger.

6.4.1 Le gradient évanescent

Le premier problème est du a une atténuation du gradient qui ne permet plus d'améliorer le générateur. D'après "Goodfellow et Al 2014" cela est du a un discriminateur trop performant qui rejete avec beaucoup de confiance toutes les productions du générateur. Cela mène donc a un effondrement du gradient nécessaire pour améliorer celui-ci. Il convient donc de trouver un équilibre entre les deux.

Intuitivement. Par exemple on peut chercher l'optimalité du générateur pour un discriminateur donné a chaque étape avant d'améliorer le discriminateur. Hélas il a été montré que cette méthode ne converge pas toujours.

6.4.2 Le "mode collapse"

Le second problème s'appelle "mode collapse". En fait dans ce cas le générateur tend à imiter un seul type de donnée exemple pour des entrés de bruits différentes. Puis quand le discriminateur apprend a détecter cette donnée le générateur passe a un autre type de donnée. Et ainsi de suite de façon "cyclique". Ce phénomène est décrit dans [13].

6.4.3 Qu'est-ce qui converge

Il existe différente solution proposé pour conter ces problèmes de convergence. Certains articles vantent les mérites des WGAN (Wasserstein GAN -> on passe de la divergence de Jensen-Shanon à la distance de Wasserstein.). Cependant d'autres articles montrent que cette approche n'est pas pertinente. Donc a éviter (si vous tombez dessus il s'agit probablement de "vieux articles") Actuellement il semblerait que la solution privilégiés soit celles des "pénalités de gradients". L'usage d'une moyenne glissante sur le gradient peut aussi donner de meilleurs résultats. Ce tableau issu de "Which Training Methods for GANs do actually Converge? [7]" résume l'état de l'art en matière de convergence des GANs.

Method	Local convergence (a.c. case)	Local convergence (general case)
unregularized (Goodfellow et al., 2014)	✓	✗
WGAN (Arjovsky et al., 2017)	✗	✗
WGAN-GP (Gulrajani et al., 2017)	✗	✗
DRAGAN (Kodali et al., 2017)	✓	✗
Instance noise (Sønderby et al., 2016)	✓	✓
ConOpt (Mescheder et al., 2017)	✓	✓
Gradient penalties (Roth et al., 2017)	✓	✓
Gradient penalty on real data only	✓	✓
Gradient penalty on fake data only	✓	✓

FIGURE 6 – Récapitulatifs des types de GANs qui convergent tirés de "Which Training Methods for GANs do actually Converge?"

7 La distillation vers des réseaux plus résistants aux exemples adversariaux ?

7.1 Position du problèmes : attaques adversariales

Considérons un classifieurs donnés F qui a une donné d'entrée x associe une classe y. Un attaquant peut être tenté de trouver x' très proche de x tel que la classe prédit de x' par F soit t ≠ y. Cela est dangereux pour les systèmes critiques, par exemples les systèmes embarqués qui exploitent le traitement d'image.

Une approche pour mener une telle attaque dite adversariale peut être de regarder le gradient de F autour de F puis de suivre la droite de plus grande pente autour de F. Cela permet de changer de classe localement. Et effectivement cela fonctionne.

7.2 Distillation

La distillation est une contre-mesure pour lutter contre ces phénomènes d'attaques adversariales. Le principe est d'entraîner un prédicteur classique sur une base de donnée ou la classe effective de chaque exemple est indiqué. Puis dans un second temps on réentraine le modèle depuis zéros mais en remplaçant ces labels par la probabilité d'être dans une classe donnée d'après le modèle préalablement entraîné. Ce procédè est décrit dans [10]

Cette méthode permettrait de réduire le gradient d'un facteur 10 ** 30 et de rendre donc le prédicteur significativement plus robuste.

Nous avons implémenter cette méthode durant ce PAF. Vous pourrez trouver le notebook associé dans le répertoire DistilledNet dur répertoire IDS du dépôt du projet.

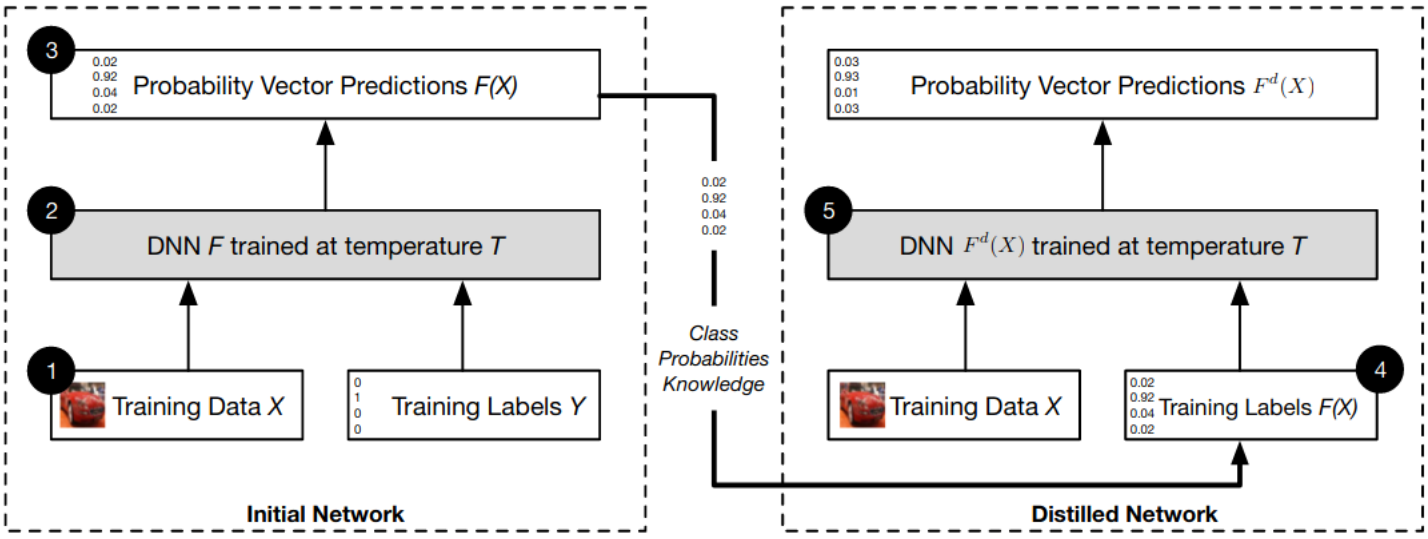


FIGURE 7 – Illustration du principe de Distillation tiré de : Towards Evaluating the Robustness of Neural Networks

7.3 La méthode CW

En fait la distillation n'est pas robuste aux attaques adversariales. C'est en effet ce que montre les résultat des attaques par la méthode C&W présentée dans l'article "Toward Evaluating the Robustness of Neural Networks". J'en explique brièvement la méthode ci-dessous :

On choisit une norme sur l'espace des intrusions. On utilise la distance issue de cette norme dans la suite du propos. L'idée est de trouver l'attaque la plus proche de l'attaque de base afin de changer de classe. Le problème est que ce problème est extrêmement coûteux algorithmiquement est difficile mathématiquement. Les auteurs propose donc de simplifier le problèmes de la façon suivante. On introduit une fonction f telle que $f(x) \leq 0$ si est seulement si x est dans la bonne classe. Puis afin de rendre le problème convexe et d'utiliser une descente de gradient (avec la méthode Adam) on considère la quantité $D = f(x+\delta x) + C\,\|\delta x\|$ pour C strictement positifs. Pour C fixé on prend alors le $\delta_C x$ qui minimise D. On cherche alors le plus petit C telle que $f(\delta_C x) \leq 0$ a l'aide d'une dichotomie. L'attaque adversariales obtenu sur x par la méthode CW est alors $x + \delta_{C_{min}} x$. Cette méthode a été introduite dans l'article [2].

Nous avons implémenter cette méthode durant le projet. Vous trouverez le code python associé dans le répertoire C&W du répertoire AS du dépôt du projet.

Les tests n'ont pas permit une exécution complète du code :
— La carte nvidia n'est pas suffisamment à jour pour utiliser les fonctionnalités de TensorFlow nécessaires.
— Une erreur que je ne comprends pas sur le module numpy. On peut consulter le détail de cette erreur dans le README associé du dépôt.

8 Expérimentations

8.1 Détails techniques pour les exécutions à distance

Afin d'exécuter les calculs à distance voici les méthodes utilisées -ce qui vous permettra de gagner du temps- .

— D'abord il faut installer le vpn open vpn de Télécom disponible sur Eole afin de se connecter aux machines
— La commande scp permet le transfert des fichiers
— La commande mail permet d'envoyer un mail en cli depuis votre adresse Télécom Paris. C'est pratique pour les calculs longs, vous pouvez vous envoyer un mailautomatiquement en fin d'execution.
— De base vous ne pouvez pas installer de bibliothèque python sur les pc de l'école. Il faut ou bien le faire en utilisateur. Ou bien utiliser un environment virtuel python.
— La bibliothèque papermil permet d'éxécuter un notebook en cli.
— La commande nohup permet de détacher un processus de son terminal et de rediriger sa sortie dans le fichier nohup.out
— Notamment la commande "nohup papermil mon-notebook.ipynb mon-notebook-out.ipynb" permet d'éxecuter le notebook mon notebook dans un processus détaché du terminal (on peut fermer la connexion ssh) et d'enregistrer le résultat dans mon-notebook-out. Les éventuels erreurs sont affichés dans nohup.out
— La commande "ps -u utilisateur" permet de voir vos processus. Pratique pour vérifier que tout ce passe bien ou pour tuer un processus que vous avez détaché du terminal avec nohup.

8.2 IDS-WGAN

Nous avons souhaité mettre en pratique un IDSGAN. Nous nous sommes pour cela basés sur le papier [8]. Le dépôt associé (`https://github.com/imoken1122/IDSGAN`) ne semblant pas exécutable, nous avons utilisé le code trouvé sur un dépôt forked (`https://github.com/HongQuangDevVN/IDSGAN-ver1.0`).

Il était alors nécessaire de comparer les deux codes pour comprendre les modifications, et vérifier que celles-ci n'impactaient pas le fonctionnement théorique du code.

8.2.1 Comparaison des dépôts

Fichiers BlackBox Aucune modification significative constatée.

Fichiers de preprocessing Ce code de preprocessing est celui utilisé par la blackbox, et par l'IDS_WGAN original. Aucune modification significative constatée.

Fichiers IDS_WGAN Les différences suivantes sont à noter :

— Au lieu d'utiliser le même fichier de prétraitement que pour la blackbox, un fichier preprocessing2.py a été créé. Les différences de celui-ci avec la version originale sont détaillées dans la section suivante.
— Plutôt que d'utiliser le fichier half_GAN_KDDTrain+.csv pour l'entraînement, c'est le fichier, KDDTrain+.csv qui est utilisé (on exploite donc plus de données).
— Le vecteur d'entrée du générateur (attaque du dataset + bruit) est explicitement converti au format float64.

Fichiers de preprocessing 2 Ce fichier n'existe pas dans la version originale, et est un fork du fichier preprocessing.py. Seules les fonctions preprocess et create_batch2 ont été analysées, car ces ont les deux seules à être utilisées.

Aucune modification significative constatée dans le code de create_batch2.

Dans le fichier original, deux fonctions de preprocessing sont présentes, mais le code d'IDS_WGAN contient une erreur dans leur appel, ce qui fait qu'il est impossible de savoir laquelle des deux est sensée être utilisée.

En tout état de cause, la fonction utilisée par la version fork est un mélange de ces deux fonctions. Elle se rapproche très fortement de la 2e fonction, en empruntant toutefois l'étape de normalisation de la 1e.

Fichiers Model Aucune modification significative constatée.

Conclusion des comparaisons Le code du projet forked semble donc se limiter à des corrections pratiques, sans toutefois modifier le fonctionnement théorique.

La seule exception à cette remarque est l'utilisation de deux fois plus de données pour l'entraînement dans la version forked.

8.2.2 Exécution des codes

Par défaut, la BlackBox est entraînée sur 50 epochs, avec un learning rate de 0.001, et la cross entropy loss comme fonction de loss.

Malheureusement, le code fournit des résultats décevants :

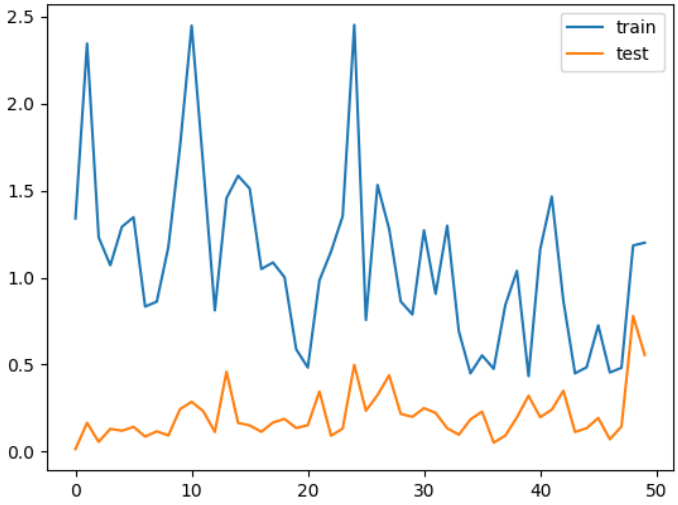


FIGURE 8 – Évolution des loss lors de l'entraînement de la BlackBox

Par la suite, l'entraînement adversarial (avec 100 epochs et pas de gradient penalty) est lui aussi peu performant :

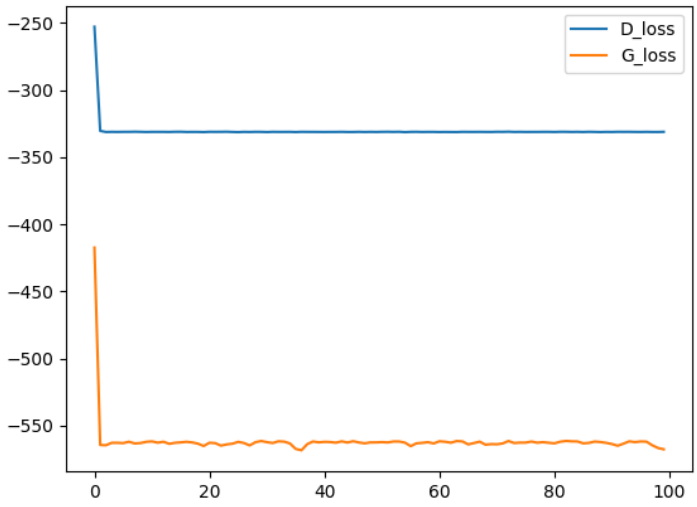


FIGURE 9 – Évolution des loss lors de l'entraînement adversarial

8.2.3 Modifications des paramètres

Un des problèmes notés est l'absence de normalisation des données au moment de l'entraînement de la BlackBox. Nous avons donc corrigé cela, diminué le learning rate à 0.0005, et augmenté le nombre d'epochs à 200 :

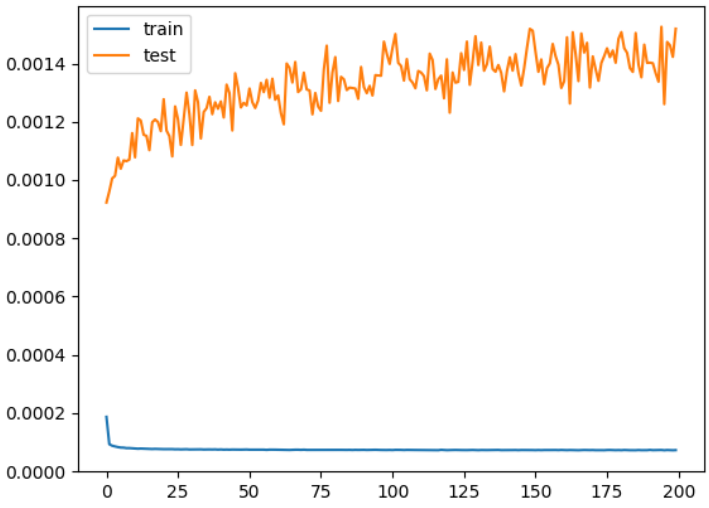


FIGURE 10 – Évolution des loss lors de l'entraînement de la BlackBox

Malheureusement, la BlackBox restait globalement mauvaise, ce qui rendait inexploitable la suite de l'entraînement.

8.2.4 Intégration d'un decision tree

Nous avons donc décidé de brancher un decision tree à la place de la BlackBox.

8.3 Expérimentation sur les décisions trees

8.3.1 Contextualisation

Nous avons exploités le notebook sur les decisons tree se basant sur NSL-KDD.

Ce code n'a pas eu besoin d'être beaucoup modifié pour fonctionner et s'exécute relativement vite. Il comporte en fait plusieurs versions et des variantes.

Chaque arbre de décision est spécifique a une des quatre attaques de NSL-KDD (DoS, Probe, U2R, R2L). Puis chacun des ces arbres dispose d'une variante avec selections des features qui consiste à ne conserver que les 13 features les plus significatives pour cette arbre de decision avec la méthode ANOVA-F test (implémenté dans la bibiothèque sklearn). Cela consite a retirer récursivement les features qui apportent le moins d'information. Les résultats de ces arbres de décisions sont impressionnants.

8.3.2 Résultat observé sans sélection des features

Nous disposons ci-dessous les différentes métriques de performances mesurées lors de nos tests. En fonction des différentes variantes. Vous pouvez les répliquez dans le notebook DecisionTree$_IDS$ du dépôt dans le répertoire IDS.

```
In [50]: metrique(Y_DoS_test,Y_DoS_pred)

         Predicted attacks     0     1
         Actual attacks
         0                  9499   212
         1                  2830  4630

         Accuracy :  0.822840836293751

         F1-Score : 0.752723134449683

         Precision  : 0.956216439487815

         Recall : 0.6206434316353887
```

FIGURE 11 – Performance du Decision Tree pour les DoS v1

```
In [64]: metrique(Y_Probe_test,Y_Probe_pred)

Predicted attacks     0     2
Actual attacks
0                  2337  7374
2                   212  2209

Accuracy :  0.3747115067589845

F1-Score : 0.374641894789828

Precision  : 0.5736712475550175

Recall : 0.5765439031887726
```

FIGURE 12 – Performance du Decision Tree pour les Probe v1

```
In [66]: metrique(Y_U2R_test,Y_U2R_pred)

Predicted attacks     0  4
Actual attacks
0                  9703  8
4                    60  7

Accuracy :  0.9930456125997137

F1-Score : 0.5836199360247685

Precision  : 0.7302605073577111

Recall : 0.5518269019437874
```

FIGURE 13 – Performance du Decision Tree pour les U2R v1

```
In [65]: metrique(Y_R2L_test,Y_R2L_pred)

Predicted attacks     0    3
Actual attacks
0                  9707    4
3                  2573  312

Accuracy :  0.7954112416640203

F1-Score : 0.538877389433873

Precision  : 0.888907042427741

Recall : 0.5538668382814464
```

FIGURE 14 – Performance du Decision Tree pour les R2L v1

8.3.3 Utilités des features

Les graphiques suivants donnes les utilités des différends features pour l'efficacité des arbres de décisions. Ils permettent de voir qu'on pert très peu d'information a ne garder que 13 features. En effet au dela de ce seuil l'efficacité oscille mais ne s'améliore plus.

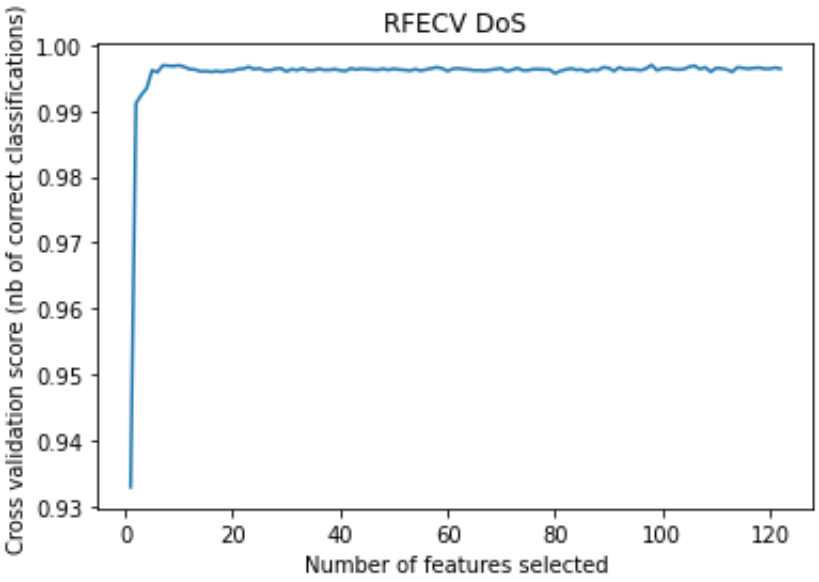


FIGURE 15 – Utilité des features pour les DoS

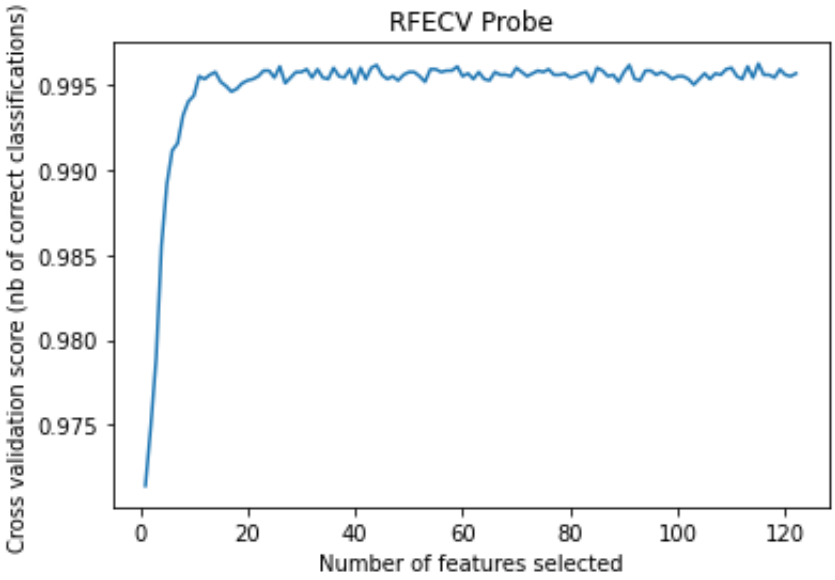


FIGURE 16 – Utilité des features pour les Probes

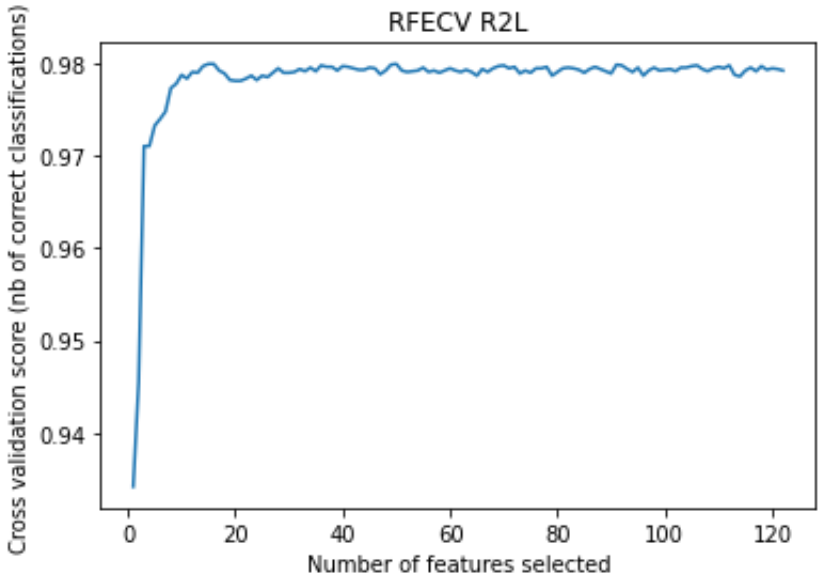


FIGURE 17 – Utilité des features pour les R2L

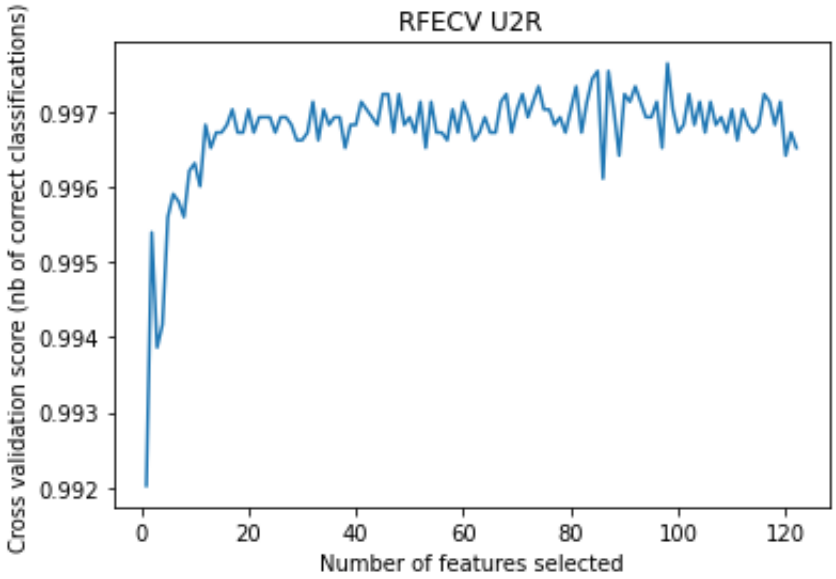


FIGURE 18 – Utilité des features pour les U2R

8.3.4 Résultats avec sélection des features

On donne les métriques mesurés sur les arbres de décisions dans le cas ou l'on a fait une sélection des features. On voit que les performances sont presques identiques lorsqu'on sélectionne des features.

```
In [82]: metrique(Y_R2L_test,Y_R2L_pred2)

         Predicted attacks     0    3
         Actual attacks
         0                  9649   62
         3                  2560  325

         Accuracy :  0.791838678945697

         F1-Score : 0.5395192342012742

         Precision  : 0.8150559495335652

         Recall : 0.5531335670192657
```

FIGURE 19 – Performance du Decision Tree pour les R2L v2

```
In [82]: metrique(Y_R2L_test,Y_R2L_pred2)

Predicted attacks     0    3
Actual attacks
0                  9649   62
3                  2560  325

Accuracy :  0.791838678945697

F1-Score : 0.5395192342012742

Precision  : 0.8150559495335652

Recall : 0.5531335670192657
```

FIGURE 20 – Performance du Decision Tree pour les R2L v2


```
In [82]: metrique(Y_R2L_test,Y_R2L_pred2)

Predicted attacks     0    3
Actual attacks
0                  9649   62
3                  2560  325

Accuracy :  0.791838678945697

F1-Score : 0.5395192342012742

Precision  : 0.8150559495335652

Recall : 0.5531335670192657
```


FIGURE 21 – Performance du Decision Tree pour les R2L v2

```
In [82]: metrique(Y_R2L_test,Y_R2L_pred2)

Predicted attacks     0    3
Actual attacks
0                  9649   62
3                  2560  325

Accuracy :  0.791838678945697

F1-Score : 0.5395192342012742

Precision  : 0.8150559495335652

Recall : 0.5531335670192657
```

FIGURE 22 – Performance du Decision Tree pour les R2L v2

8.4 Exploration de voisinages

8.4.1 Exploration des voisinages d'une attaque adversariale

Le but de cette expérimentation était d'interpréter les résultats obtenus pour les méthodes de génération d'attaque adversariales. Nous avons écrit un notebook qui permet de comparer les attaques adversariales obtenus et celles de la base de donnée. Ce notebook retourne les k attaques de la base de donnés les plus proches de l'attaque adversariale considérée -où k est spécifiés par l'utilisateur-. Notamment selon les sections on peut aussi ne conserver que les éléments de la base de donnés que l'IDS attaqué détecte pas ou les éléments que l'IDS détecte.

```
In [18]: print(Rayons_echappant)

[[5.31956869 5.31937739 5.33689433 5.32621614 5.3002451  5.3275871
  5.31370627 5.30327373 5.10565421 5.33546127]]

In [19]: print(Attaque_echappant)

[[array([-9.93194524e-02, -2.56920747e-02, -5.23087470e-02, -1.48580045e-02,
         -9.44133763e-02, -6.64411495e-03, -8.55959191e-02, -2.15171149e-02,
         -8.70242969e-01, -1.22491853e-02, -3.47989093e-02, -2.56356488e-02,
         -1.30071692e-02, -2.63597259e-02, -1.81631753e-02, -4.27709864e-02,
          0.00000000e+00, -2.97128607e-03, -8.81312159e-02, -4.34629698e-01,
         -1.53736344e-01,  1.49753446e+00,  1.50257981e+00, -3.12759485e-01,
         -3.15134925e-01, -4.17404199e-01,  5.26998392e-02, -3.50012854e-01,
         -1.53684565e+00, -5.35431013e-01, -1.86005350e-01,  2.74436700e-01,
         -2.68180110e-01,  2.43737337e-01,  1.39168390e+00, -5.56149238e-02,
         -3.28182176e-01, -3.18156129e-01, -1.95163021e-01,  4.27208618e-01,
         -3.65150198e-01, -4.05560725e-02, -2.43281135e-02, -8.70051422e-02,
          0.00000000e+00, -9.14287924e-02, -7.87998382e-02, -8.03169369e-02,
         -6.87591292e-02, -6.90824183e-02, -6.67871604e-02, -6.79115115e-02,
         -6.95966164e-02, -2.94395505e-01, -6.07139078e-02, -6.63858664e-02,
         -1.65899877e-01, -6.50990723e-02, -6.42040318e-02, -1.23916822e-01,
         -1.12151172e-01,  4.15812516e+00, -6.55760416e-02,  0.00000000e+00,
         -6.29440872e-02, -7.43213869e-01,  0.00000000e+00, -6.81080344e-02,
          0.00000000e+00, -7.44876116e-02, -7.74269824e-02, -6.13696017e-02,
         -5.08387056e-02, -5.97542820e-02, -6.34369914e-02, -6.10061845e-02,
```

FIGURE 23 – Résultat obtenus echappant pour un élément bruité

```
In [14]: print(Rayons_decouvert)

[[1.21598514 1.21095796 1.22515242 1.22026906 1.21514114 1.22560962
  1.21204516 1.22439145 1.2256392  1.21939344]]

In [15]: print(Attaque_decouvert)

[[array([-9.93194524e-02, -2.41694947e-02, -5.23087470e-02, -1.48580045e-02,
         -9.44133763e-02, -6.64411495e-03, -8.55959191e-02, -2.15171149e-02,
         -8.70242969e-01, -1.22491853e-02, -3.47989093e-02, -2.56356488e-02,
         -1.30071692e-02, -2.63597259e-02, -1.81631753e-02, -4.27709864e-02,
          0.00000000e+00, -2.97128607e-03, -8.81312159e-02, -7.63362269e-01,
         -3.63954411e-01, -6.77468066e-01, -6.74869545e-01, -3.12759485e-01,
         -3.15134925e-01,  7.84104085e-01, -3.61209020e-01, -3.50012854e-01,
         -3.85496362e-01, -8.84122928e-01, -8.15525519e-01, -1.94791730e-01,
          3.33144290e-01, -2.83192321e-01, -6.79073200e-01, -6.66814438e-01,
         -1.49019397e-01, -3.18156129e-01, -1.95163021e-01,  4.27208618e-01,
         -3.65150198e-01, -4.05560725e-02, -2.43281135e-02, -8.70051422e-02,
          0.00000000e+00, -9.14287924e-02, -7.87998382e-02, -8.03169369e-02,
         -6.87591292e-02, -6.90824183e-02, -6.67871604e-02, -6.79115115e-02,
         -6.95966164e-02, -2.94395505e-01, -6.07139078e-02, -6.63858664e-02,
         -1.65899877e-01, -6.50990723e-02, -6.42040318e-02, -1.23916822e-01,
         -1.12151172e-01,  4.15812516e+00, -6.55760416e-02,  0.00000000e+00,
         -6.29440872e-02, -7.43213869e-01,  0.00000000e+00, -6.81080344e-02,
          0.00000000e+00, -7.44876116e-02, -7.74269824e-02, -6.13696017e-02,
         -5.08387056e-02, -5.97542820e-02, -6.34369914e-02, -6.10061845e-02,
```

FIGURE 24 – Résultat obtenus découvert pour un élément bruité

8.4.2 Observation de la boule formées par ces attaques adversariales

Ce second notebook s'inscrit dans le prolongement du précédent. En observant les attaques générés il est apparu que ces attaques était très proche les unes des autres. C'est éventuellement une observation du **mode collapse** sur les GANs. Il est alors

venus l'idée de regarder la boule constituée par ces attaques afin de regarder si cela définissait une zone entière ignorée par l'IDS.

	0	1	2	3	4	5	6	7	8	9	...	104	105	106	
count	58631.0	58631.0	58631.0	58631.0	58631.000000	58631.000000	58631.000000	58631.000000	58631.000000	58631.0	...	58631.0	58631.0	58631.0	58
mean	0.0	0.0	0.0	0.0	0.976387	0.981704	0.992231	0.993298	0.000014	0.0	...	0.0	0.0	0.0	
std	0.0	0.0	0.0	0.0	0.006441	0.005748	0.004616	0.004489	0.000288	0.0	...	0.0	0.0	0.0	
min	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	...	0.0	0.0	0.0	
25%	0.0	0.0	0.0	0.0	0.973401	0.979320	0.991051	0.992279	0.000000	0.0	...	0.0	0.0	0.0	
50%	0.0	0.0	0.0	0.0	0.976917	0.982158	0.992529	0.993550	0.000000	0.0	...	0.0	0.0	0.0	
75%	0.0	0.0	0.0	0.0	0.979974	0.984620	0.993761	0.994616	0.000000	0.0	...	0.0	0.0	0.0	
max	0.0	0.0	0.0	0.0	0.990610	0.992981	0.997621	0.997937	0.016664	0.0	...	0.0	0.0	0.0	

FIGURE 25 – Ditribution statistiques des attaques générées

La démarche est alors la suivante :
$\forall i$, on pose $\mu_i = \mathbb{E}(features)$ on pose $\sigma_i = Std(features)$
On considère alors le pavé P = $[\mu_0 - \sigma_0, \mu_0 + \sigma_0] \times ... \times [\mu_n - \sigma_n, \mu_n + \sigma_n]$
L'hypothèse que nous souhaitons tester est que ce pavé défini une **région entière de mauvaise classifications par l'IDS** que le GAN aurait découvert.

Éventuellement cette théorie ce porte a généralisation. Si le GAN se dirige vers K zones. Alors on applique un algorithme de clustering de type K-means puis on applique la même processus sur chaque compact obtenu. L'intérêt éventuel de ce modèle est que si l'hypothèse est vérifié alors on peut se passer du GAN une fois ce compact découvert. Cela mettrait aussi en évidence une défaillance de l'IDS.

Dans l'exécution de ce code nous avons cependant rencontré un problème a savoir que 8 features semblent manquer au vecteurs d'attaques du fichier attack.csv. En observant sur la base de sonnées ce qui était probable j'ai comlété les 8 features manquants. Cependant cela nécessiterait d'être corrigé car manque de rigeur. De plus le fichier attack.csv a été produit lorsque l'architecture du GAN n'était pas encore stabiliser. Cela explique sûrement cette incohérence.

Modulo les limites évoquées ci-dessus les résultats semblent infirmer notre hypothèse. En effet en générant des vecteurs aléatoires dans la boule sur mille essais nous ne trompons jamais le decision tree.

In [16]:
```
NOMBRE_TEST = 1000

NOMBRE_ECHAPEMENT = 0

for i in range(NOMBRE_TEST):
    if IDS.predict([np.concatenate((genere_attaque(),[0,0,0,0,0,0,0,0]))])[0]>0.5:
        NOMBRE_ECHAPEMENT += 1

Taux_echappement = NOMBRE_ECHAPEMENT / NOMBRE_TEST

print("Taux d'échapement de:   " + str(Taux_echappement))
```

```
Taux d'échapement de:   0.0
```

FIGURE 26 – Résultats obtenus

9 Conclusion et prolongement

Il reste de nombreuses voies afin de prolonger ce projet. Notamment une extension importante serait d'étudier les effets d'une présélection des exemples de la base de donnée.

En effet, il semblerait que le paramétrage de nos GANs influent drastiquement sur la convergence de ceux-ci. Or cette convergence dépend aussi de la base de donnée. Tester la convergence du GAN sur une base de données ou nous aurions pré-sélectionné une attaque en particulier pourrait donc être intéressant. Cela pourrait permettre de concentrer la complexité du GAN sur un type d'attaque.

Il serait également intéressant d'étudier l'évolution de la robustesse des IDS entraînés par un GAN. Notamment de tester des attaques adversariales entraînés pour un autre IDS black-box dessus. Cela permettrait de tester les hypothèses de transferabilité.

Enfin achever la méthode C&W (débuguage) afin d'en observer les performances pourrait être très intéressant.

[1] et [4] et [6] et [3] et [5] et [12] et [11]

Références

[1] Anna L. BUCZAK et Erhan GUVEN. *A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection*. 2015. URL : https://ieeexplore.ieee.org/document/7307098.

[2] Nicholas CARLINI et David WAGNER. *Towards Evaluating the Robustness of Neural Networks*. 2017. URL : https://arxiv.org/pdf/1608.04644.pdf.

[3] Ansam KHRAISAT et al. *Survey of intrusion detection systems : techniques, datasets and challenges*. 2019. URL : https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7.

[4] Sailesh KUMAR. *Survey of Current Network Intrusion Detection Techniques*. 2007. URL : https://www.cse.wustl.edu/~jain/cse571-07/ftp/ids/.

[5] Dr. S.P. Shantharajah L.DHANABAL1. *A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms*. 2015. URL : http://faratarjome.ir/u/media/shopping_files/store-EN-1484204753-3159.pdf.

[6] Maxime LABONNE, Alexis OLIVEREAU et Djamal ZEGHLACHE. *Automatisation du processus d'entraînement d'un ensemble d'algorithmes de machine learning optimisés pour la détection d'intrusion*. 2017. URL : https://www.cesar-conference.org/wp-content/uploads/2018/11/articles/C&ESAR_2018_J2-09_M-LABONNE_automatisation_ML_pour%5C%20detection_intrusion.pdf.

[7] Sebastian Nowozin LARS MESCHEDER Andreas Geiger. *Which Training Methods for GANs do actually Converge ?* 2018. URL : arXiv:1801.04406.

[8] Zilong LIN, Yong SHI et Zhi XUE. *IDSGAN : Generative Adversarial Networks for Attack Generation against Intrusion Detection*. 2019. URL : https://arxiv.org/pdf/1809.02077.pdf.

[9] OBOZINSKI. *Cours Apprentissage, théorie de la décision*. 2013. URL : http://www.math.ens.fr/cours-apprentissage/Obozinski/cours_2013/cours1.pdf.

[10] Nicolas PAPERNOT et al. *Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks*. 2015. URL : https://arxiv.org/abs/1511.04508.

[11] Jason Brownlee PHD. *A tour of machine learning algorithms*. 2019. URL : https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/.

[12] Aurobindo SUNDARAM. *An Introduction to Intrusion Detection*. 1996. URL : https://neuro.bstu.by/ai/To-dom/My_research/Paper-0-again/For-research/D-mining/Anomaly-D/Intrusion-detection/Intrusion-Detection-Intro.pdf.

[13] Hoang THANH-TUNG, Truyen TRAN et Svetha VENKATESH. *On catastrophic forgetting and mode collapse in Generative Adversarial Networks*. 2018. URL : https://www.groundai.com/project/on-catastrophic-forgetting-and-mode-collapse-in-generative-adversarial-networks/1.