Un apprentissage fédéré avec une sélection de clients pour la détection d'intrusions réseau

par

Selma BOUOUDINA

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE
M. Sc. A.

MONTRÉAL, LE 7 FÉVRIER 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Chamseddine Talhi, directeur de mémoire
Département de génie logiciel et TI à l'École de technologie supérieure

M. Kaiwen Zhang, président du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Ali Ouni, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 22 JANVIER 2020

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

En préambule à ce mémoire je remercie ALLAH le tout puissant et miséricordieux qui m'a donné la force et la patience d'accomplir ce travail durant ces longues années d'étude.

Je souhaite adresser mes sincères remerciements à mon directeur de recherche ; Dr.Chamseddine Talhi, pour son précieux conseil et son aide durant toute la période de ma maîtrise. Mes vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner ce mémoire et de l'enrichir par leurs propositions.

C'est avec une profonde gratitude que j'exprime mes sincères remerciements envers ma petite famille surtout mes beaux parents, ma mère Fatiha et mon père Ali qui m'ont donné tous les moyens pour réussir, sans oublier mon cher mari Mohamed qui a su me motiver et qui ne cesse jamais de m'encourager, c'est par leur prières que j'ai pu surmonter tous les obstacles.

Enfin, j'aimerais remercier l'École de technologie supérieure et Ericsson Canada qui ont financé ce travail et toutes les personnes qui ont contribué de près ou de loin à l'élaboration de ce mémoire ainsi qu'à la réussite de ces formidables années universitaire.

Un apprentissage fédéré avec une sélection de clients pour la détection d'intrusions réseau

Selma BOUOUDINA

RÉSUMÉ

Une quantité toujours plus large de données sensibles sont produites et stockées sur des machines personnelles. La vision centralisée qui prévalait auparavant en apprentissage automatique est donc en train de changer et un fort accent est mis sur la vie privée. Ces données importantes peuvent aider les chercheurs à analyser et à comprendre des phénomènes et des modèles intéressants. Bien que les avantages d'un traitement massif de ces données soient incontestables, ils représentent malheureusement une menace considérable pour la vie privée des utilisateurs qui peuvent subir des conséquences importantes. L'objectif principal est d'étudier le problème de la confidentialité des données des utilisateurs privés et de proposer des solutions qui préservent la vie privée. Les méthodes d'apprentissage automatique standard exigent que les données d'apprentissage soient centralisées en un seul endroit. L'un des objectifs est de concevoir, de mettre en œuvre et d'évaluer des méthodes fédérées novatrices et pratiques d'analyse des données privées. Différents ensembles de données, avec des structures différentes et parfois complexes, seront pris en compte. Dans ce cadre, nous proposons l'apprentissage fédéré ou encore federated learning une orientation prometteuse qui utilise une approche décentralisée sur un jeu de données connu dans la littérature. Pour ce faire, toutes les entités participantes, comme les utilisateurs d'un téléphone intelligent, doivent partager leurs ensembles de données. L'apprentissage fédéré permet à différentes entités d'apprendre en collaboration un modèle de prévision partagé sans partager leurs données locales. Nous étudierons les méthodes existantes basées sur une approche d'apprentissage profond pour la détection d'intrusions sur le réseau. Enfin, nous offrons une solution au problème posé en proposant une optimisation d'un algorithme d'apprentissage distribué s'appuyant sur deux paradigmes d'apprentissage existant les autoencoders et l'apprentissage fédéré. Nous détaillerons notre approche, son implémentation en comparant ses performances avec les méthodes existantes.

Mots-clés: Préservation de la vie privée, approche décentralisée, algorithmes distribués, apprentissage fédéré, réduction de la dimensionnalité, autoencodeurs

Federated approach preserving privacy for network intrusion detection

Selma BOUOUDINA

ABSTRACT

An ever increasing amount of sensitive data is produced and stored on personal machines. The centralized view that prevailed previously in machine learning is therefore changing and a strong emphasis is placed on privacy which can help researchers analyze and understand interesting phenomena and models. While the benefits of massive data processing are indisputable, they also represent a significant threat to the privacy of users who may suffer significant consequences. Our main objective in this research is to study users privacy problem and to propose solutions that preserve privacy. Traditional machine learning methods require that learning data be centralized in one place. One of the goals is to design, implement, and evaluate innovative federated and practical methods for private data analysis. Different sets of data, with different (sometimes complex) structures, will be taken into account. In this context, we will propose federated learning ; is a promising direction that uses a decentralized approach on a well-known dataset in the literature. To achieve the task, all participating entities, such as smartphone users, must share their datasets. Federated learning allows different entities to collaboratively learn a shared model without sharing their local data. We will study existing methods based on a deep learning approach for network intrusion detection. Finally, we will offer a solution to the problem by proposing an optimization for our distributed learning algorithm based on two existing learning paradigms, autoencoders and federated learning. We will detail our approach, its implementation by comparing its performance with existing methods.

Keywords: Preserving privacy, Decentralized approach, Distributed Algorithms, Federated learning, Features selection, Autoencoders.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

Acc	Accuracy
ANN	Artificial neural network
AUC	Area under the curve
CNN	convolutional neural network
DL	Deep learning
FIDS	Federated Intrusion detection system
FL	Federated learning
FN	False Negative
FP	False Positive
GD	Gradient descent
HIDS	Host intrusion detection system
IDS	Intrusion detection system
IID	Independent and identically distributed
LSTM	Long short-term memory
NIDS	Network intrusion detection system
NON-IID	Non Independent and identically distributed
OCSVM	One class Support vector machine
RNA	Réseau de neurones artificiels
RNN	Recurrent neural network
SAG	Stochastic Average Gradient
SAGA	Stochastic Average Gradient Accelerated
SGD	Stochastic gradient descent
SVM	Support vector machine

SVRG	Stochastic Variance Reduced Gradient
TN	True Negative
TP	True Positive

LISTE DES SYMBOLES ET UNITÉS DE MESURE

GB	GegaByte
RX Byte	Received Byte
s	Second
TX Byte	Transmit Byte

INTRODUCTION

La cybersécurité continue d'être un grave problème pour tous les secteurs dans le cyberespace que le nombre de failles de sécurité augmente plus en plus. On sait que des milliers d'attaques apparaissent constamment en raison de l'ajout de divers protocoles principalement à partir d'Internet des objets (IdO). La plupart de ces attaques sont de petites variantes de cyberattaques connues précédemment. Cela indique que les mécanismes même avancés tels que les systèmes d'apprentissage machine traditionnels font face à la difficulté de détecter ces petits mutants d'attaques au fil du temps. D'autre part, le succès de l'apprentissage profond dans divers grands champs de données a attiré plusieurs intérêts dans les domaines de la cybersécurité où l'application des algorithmes de réseau de neurones a été pratique en raison de l'amélioration des aspects de la surconsommation. L'utilisation de DL pour la détection des attaques dans le domaine de la cybersécurité pourrait être un mécanisme résistant aux petites mutations ou des nouvelles attaques en raison de sa capacité d'extraction de caractéristiques de haut niveau. Les capacités autodidactes et de compression d'architectures de modèle dans un apprentissage profond sont des mécanismes clés pour la découverte de motifs cachés à partir des données d'apprentissage afin que les attaques sont victimes de discrimination de la circulation bénigne. Cette recherche vise à adopter une nouvelle approche se basant sur l'apprentissage fédéré à la cybersécurité pour permettre la détection d'intrusions rèseau.

Problèmatique

En tant que des technologies émergentes, elles ont permis la collecte, le traitement et la communication des données dans les applications intelligentes. Ces nouvelles fonctionnalités ont séduit les créateurs des villes intelligentes (smart cities) et les professionnels de la santé gagne une application massive dans le bord des réseaux pour les applications en temps réel qui ont accès à une quantité considérable de données issues de leurs puissants capteurs (appareils photos, microphones, GPS, etc.), ainsi qu'à des objets connectés auxquels ils sont reliés. Cette source

riche de données est potentiellement porteuse de nombreuses promesses pour les opérateurs télécoms et leurs clients car elle permet de contrôler au plus près la qualité des services et leur ergonomie, et elle permet aussi d'envisager des services innovants par apprentissage statistique. Mais ces données sont aussi, et surtout, des données très personnelles et très sensibles.

Cependant, la croissance du nombre des cyber-attaques inconnus a jeté une ombre sur l'adoption de ces services intelligents. Cela découle du fait que la distribution et l'hétérogénéité des applications services rendent la sécurité plus complexe et difficile. En outre, la détection des attaques est radicalement différente des mécanismes existants en raison des exigences de service spécial du rèseau qui ne peuvent pas être satisfaits par le cloud centralisé : faible latence, la limitation des ressources, la distribution, la mobilité et l'évolutivité. Cela signifie que ni le clouding, ni des solutions de détection d'attaque autonomes résoudre les problèmes de sécurité rèseau. De ce fait, un roman actuellement émergé est l'intelligence distribuée, connue sous le nom fog computing, devrait être étudiée pour combler l'écart.

Bien que l'architecture du fog computing peut offrir les exigences de service nécessaires et des ressources distribuées, les mécanismes de sécurité robustes sont également les ressources nécessaires pour protéger les appareils. Comme les régimes de sécurité préventive sont toujours à la conception des lacunes et des défauts de mise en œuvre, les mécanismes de détective tels que la détection des attaques sont inévitables qui peuvent être dététecter soit par la signature ou les systèmes à base d'anomalies. La solution de signature correspond au trafic entrant contre les types d'attaques déjà connues dans la base de données en système basé sur la détection anomalie répond d'attaque comme un écart de comportement du trafic normal. La première approche a été largement utilisée en raison de sa grande précision de la détection et le faible taux de fausse alarme, mais critiqué pour son incapacité à capturer des nouvelles attaques. La détection d'anomalies, d'autre part, détecte de nouvelles attaques mais il manque une grande précision. Dans les deux approches, l'apprentissage machine classique a été largement utilisé. Avec le plus

en plus dans le pouvoir de l'attaquant et les ressources, les algorithmes d'apprentissage machine traditionnels sont incapables de détecter les infractions complexes de la cybersécurité.

L'intelligence artificielle d'aujourd'hui doit encore faire face à deux défis majeurs. La première est que dans la plupart des industries, les données existent sous la forme d'îles isolées. L'autre est le renforcement de la confidentialité et de la sécurité des données. Dans ce contexte, nous proposons une solution possible à ces défis appelée l'apprentissage fédéré. Au-delà du cadre d'apprentissage fédéré proposé pour la première fois par Google en 2016, nous présentons un cadre d'apprentissage fédéré sécurisé et complet pour la détection d'intrusion réseau.

Objectifs et réalisations L'objectif est de réaliser une instanciation d'un des concepts clés de l'apprentissage fédéré qui est basé sur un modèle de classification binaire de façon décentralisée et fortement distribuée en partageant les paramètres de modèles statistiques plutôt que les données sensibles. Nous fournissons des définitions, des architectures et des applications pour le cadre d'apprentissage fédéré, ainsi qu'un état de l'art des travaux existants. En outre, nous proposons une solution efficace pour permettre le partage des connaissances sans compromettre la vie privée des utilisateurs. La performance de tel modèle est comparé à l'approche traditionnelle de l'apprentissage machine, et la détection d'attaque distribuée est évaluée par rapport au système de détection centralisée. Les expériences ont montré que notre système de détection d'attaque distribuée est supérieure aux systèmes de détection centralisée à l'aide de modèle d'apprentissage profond. Il a également été démontré que le modèle profond fédéré est très efficace pour la détection d'intrusion sur le réseau.

Les contributions de notre recherche sont :

• Concevoir et mettre en oeuvre un mécanisme de détection d'attaque distribuée basée sur l'apprentissage fédéré qui reflète les caractéristiques de distribution.

• Consacrer une phase de sélection de caractéristiques avec l'approche centralisée applée aussi une compression du modèle basée sur un apprentissage non supervisé en utilisant les encodeurs automatiques.

• Démontrer l'efficacité de l'appretissage profond dans les systèmes de détection d'attaques par rapport à l'apprentissage machine traditionel dans les applications distribuées.

• Comparer les performances du système de détection d'intrusion fédéré proposé avec une approche centralisée basé sur une classification binaire.

L'organisation de ce rapport sera comme suit :

Chapitre 1 : Ce chapitre présente un survol sur les concepts de base de systèmes de détection d'intrusion, en incluant les mécanismes, l'architecture et les limitations de sécurité.

Chapitre 2 : Nous allons passer en revue les systèmes de détection d'intrusion, en étudiant et analysant les travaux connexes pour arriver à comparer notre approche avec la référence ciblée.

Chapitre 3 : Nous allons présenter le cadre de sécurité proposé ainsi qu'une analyse de données choisis. Ce chapitre décrit les différents modules mis en place et donne des détailles sur les algorithmes déployés.

Le chapitre 4 : C'est le dernier chapitre de ce mémoire, il est consacré pour une étude comparative de trois algorithmes proposés avec les paramètres d'évaluation utilisés et les ensembles de données associés aux scénarios de test. Ce chapitre comporte les différents tests de détection et de surconsommations, suivis d'une discussion des résultats obtenus.

Enfin, nous allons décrire les différents taches accomplis la solution proposée dans ce travail. Cette partie énumère les perspectives prévus et conclut le rapport.

CHAPITRE 1

MISE EN CONTEXTE

1.1 Introduction

Au cours des dernières années, les réseaux ont évolués plus que jamais et l'Internet a changé l'informatique. Les possibilités et opportunités deviennent illimitées, mais d'autre part, les risques et les chances de menace ont augmenté. La sécurité est devenue un problème majeur dans la gestion des réseaux d'entreprise ainsi que pour les particuliers toujours plus nombreux à se connecter au net. La conception des mécanismes de sécurité de manière à empêcher l'accès non autorisé à des ressources système et de données est très importante.À l'heure actuelle, l'empêchement complet des violations est irréel mème avec plusieurs tentatives et des mesures qui puissent être prises afin de réparer les dégâts à long terme. Ce domaine de recherche est appelé la détection d'intrusion. Ce chapitre a pour but de présenter les notions de bases sur qui s'articulent les systèmes de détections d'intrusions ainsi que les différentes méthodes, techniques et outils pour développer des systèmes plus performants. Pour bien présenter les concepts du système de détection d'intrusion, nous allons donner tout d'abord une vision globale sur Les IDS et expliquer pourquoi les systèmes de détection d'intrusion sont nécessaires. Ensuite, nous présenterons les différentes sortes des NIDS et les critères importants pour sa construction et enfin, nous abordons la détection d'intrusions en utilisant une nouvelles approche dite fédérée avec une brève déscription des mèthodes distribuées existantes.

1.2 Mécanismes de sécurité : Concepts et outils

Depuis l'article d'Anderson en 1980 [85], plusieurs techniques pour la détection des intrusions ont été étudiées. Plusieurs nouvelles méthodes de détection ont été mises en place. Plusieurs efforts de recherche ont été lancés et des résultats efficaces ont été obtenus. Les intrusions sont causées par des attaquants qui accèdent aux systèmes par Internet, qui sont des utilisateurs autorisés par le système et qui tentent de gagner plus de privilèges pour lesquels ils ne sont pas

autorisés, ou des utilisateurs autorisés qui abusent des privilèges qui leurs sont attribués. La technologie de la détection d'intrusion renforce la sécurité du réseau en ajoutant une couche de protection. La détection d'intrusion permet aux organisations de protéger leurs systèmes contre les menaces qui s'introduisent avec la connectivité croissante des réseaux. Les IDS ont obtenu l'acceptation comme un complément nécessaire à l'infrastructure de sécurité de chaque organisation. Même si la technologie de détection d'intrusion ne puisse pas offrir une protection complète contre les attaques, ils améliorent l'approche de défense en profondeur, qui est la tendance à la mode de la sécurité réseau. En fin, nous allons discuter de différents aspects de détection d'intrusion et pourquoi nous avons besoin de les utiliser et comment ils renforcent la sécurité du réseau.

1.2.1 Système de détection d'intrusion

Il y a plusieurs types d'IDS disponibles aujourd'hui, caractérisé par différente approche de la surveillance et de l'analyse. Chaque approche a ses avantages et inconvénients distincts. Toutes les approches peuvent être décrites en termes d'un modèle d'IDS. Nous allons aborder deux la détection d'intrusion basée sur l'hôte, puis les IDS basée sur une application, avant de nous intéresser aux NIDS ou encore IDS réseaux.

La meilleure façon de classification les IDS est de les regrouper par emplacement de l'information source où ils opèrent. Les sources des informations primaires sont les paquets réseau capturés à partir des réseaux ou des segments de réseau local ou les systèmes d'exploitation et les fichiers critiques.

1.2.1.1 La détection d'intrusion basée sur l'hôte

Les HIDS analysent l'information concernant l'hôte courant. Avec une grande fiabilité et précision, l'IDS peut déterminer exactement quels utilisateurs ou quels processus sont impliqués dans une attaque. Avec ces types de systèmes, le résultat peut être déterminée à la différence du NIDS, le HIDS peut accéder directement et de contrôler les fichiers de données et les processus

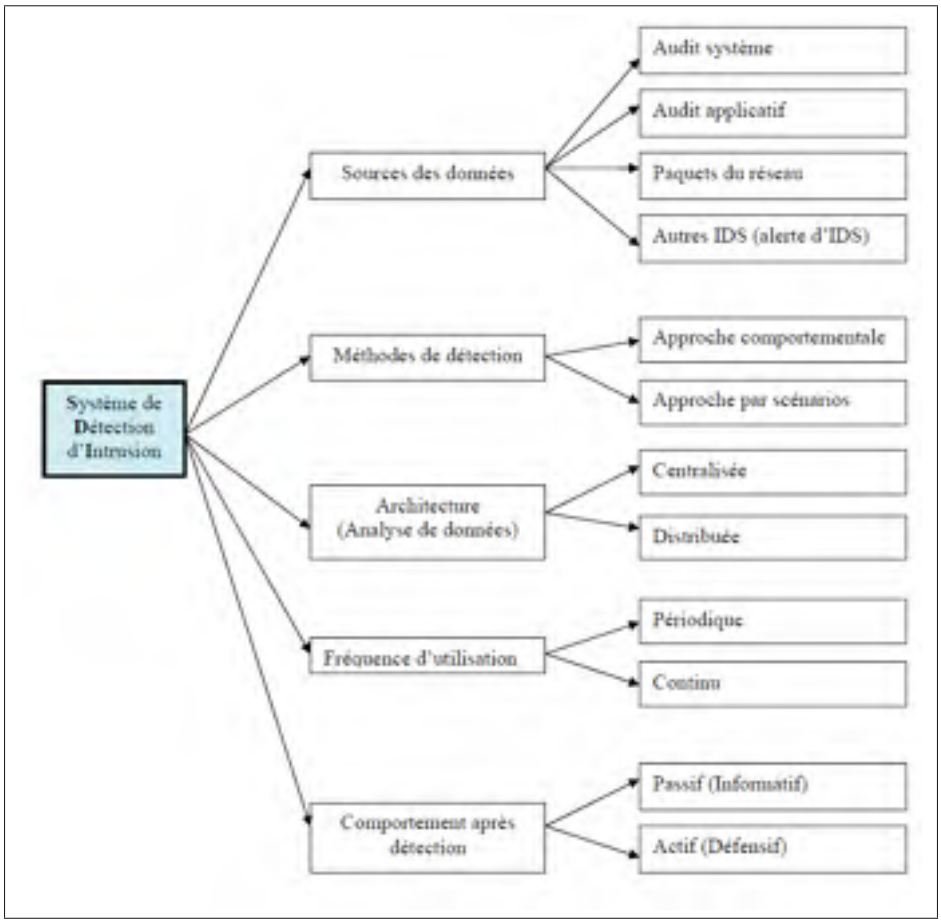


Figure 1.1 Les critères de classification des IDS

habituellement visés par les attaques. Ces IDS utilisent généralement les traces d'audit du système d'exploitation afin de fournir une information sur l'activité de la machine.

1.2.1.2 La détection d'intrusion réseau

La forme la plus commune de systèmes de détection d'intrusion est basée sur le réseau. Ces systèmes détectent les attaques en capturant et en analysant les paquets réseau en écoutant sur le segment de réseau ou d'un commutateur. Ils le font en faisant correspondre entre un ou plusieurs paquets contre des données de signatures d'attaques connues, ou d'effectuer le décodage du protocole pour détecter les anomalies. Le NIDS est capable à la fois de déclencher des alertes et de clôturer les connexions instantanément chaque fois qu'il remarque des activités suspectes. Le

« mode Promiscuous » est la forme la plus commune du fonctionnement, ils surveillent chaque paquet qui est en transmission du segment local.

Son déploiement à peu d'impact sur le réseau, les NIDS sont généralement des dispositifs passifs qui écoutent sur le fil réseau sans interférer avec le fonctionnement normal d'un réseau. Un large réseau peut être contrôlé par quelques NIDS bien placés. Certains NIDS ont des difficultés à traiter des fragments de paquets, ce qui peut provoquer le dysfonctionnement d'un IDS. Ils ont des problèmes dans le traitement des vitesses élevées et des volumes élevés de trafic et ils ne peuvent pas analyser les informations cryptées.

1.2.2 Modes de détection

Nous allons aborder deux modes de détection, la détection d'anomalies et de signatures qui sont les plus utilisé par les systèmes de détection d'intrusion.

1.2.2.1 La détection d'anomalies

Cette approche permet de produire l'information utile pour la définition des signatures pour les systèmes de détection d'intrusion à base de signatures. Le point négatif de cette approche est le grand nombre de fausses alarmes dues aux comportements imprévisibles des utilisateurs du réseau et elle exige souvent l'historique à long terme des évènements enregistrés afin de caractériser les modèles normaux de comportement. Les systèmes basés sur ce mode doivent être dotés d'une certaine intelligence pour raison d'apprentissage automatique.

1.2.2.2 La reconnaissance de signature

Cette approche est très efficace pour détecter des attaques sans produire un grand nombre de fausses alarmes avec un diagnostique rapid et sûre de l'utilisation d'un outil spécifique ou une technique d'attaque.Ceci peut aider les responsables de sécurité à donner la priorité aux mesures correctives.

En revanche, elle souffre de plusieurs inconvénients comme ele peut seulement détecter les attaques connues, dont les signatures sont introduites dans le système, donc le système de détection doit être constamment mis à jour avec les signatures des nouvelles attaques. Beaucoup de systèmes adoptant cette approche sont conçus pour employer un nombre limité de signatures qui peuvent être définis, ce qui les empêchent de détecter des variantes de ces attaques. Une fois l'attaque est détectée, un système de détection d'intrusion a deux types de réponses soit par une réponse passive qui est toujours disponible, ou une réponse active implémentée.

1.2.3 Fréquence d'utilisation

Enfin, une dernière différenciation est la caractéristique d'utilisation qui peut se faire d'une façon : continue (online) ou périodique (offline).

1.2.3.1 Utilisation continue

La détection d'attaque se fait au moment où elle se produit. Dans la plupart des cas avant d'attaquer un réseau, l'attaquant doit scanner l'environnement pour récolter des informations. Par conséquent, en voyant cela, on peut détecter une attaque avant même qu'elle ne se produise et ainsi y répondre le plus tôt possible.

1.2.3.2 Utilisation périodique

Cette méthode s'exécute périodiquement et on ne voit que le résultat d'attaque, elle est préférable pour avoir une défense plus fiable en terme du temps de calcul que pour la première utilisation. Contrairement à l'IDS online, l'attaque ne peut pas être détectée le plus tôt possible pour l'éviter. Plus une attaque est détectée tardivement, les dommages sont importants.

1.2.4 Critères d'évaluation

Malgré le fait que les systèmes de détection d'intrusions sont devenus les outils de défense omniprésents dans les systèmes informatiques d'aujourd'hui, jusqu'à présent on n'a aucune

méthodologie complète et scientifique rigoureuse pour examiner l'efficacité de ces systèmes. Dans ce qui suit, nous allons donner un ensemble de mesures partielles qui peuvent être utilisées pour tester le rendement des IDS.

1.2.5 Erreurs de classification

Il existe plusieurs types d'erreurs venant d'un détecteur, influençant plus ou moins sa puissance. Les vrais positifs sont les cas où une alarme se déclenche quand il y a une violation des politiques de sécurité. Les vrais négatifs sont les cas où aucune alarme ne se déclenche et rien d'anormal ne se produit. Les faux positifs sont les cas où une alarme se déclenche alors qu'il ne se produit rien d'anormal. Les faux négatifs sont les cas où une alarme ne se déclenche pas alors qu'il se produit une chose anormale. A première vue, on pourrait supposer qu'un faux positif est moins dangereux qu'un faux négatif.

1.2.6 Limitations

• Les fausses alarmes

La détection d'anomalie est capable de détecter les attaques inconnues, toutefois elle n'est pas aussi efficace. Notamment, un fort taux de faux positifs peut être rencontré si le choix de paramètres de l'IDS n'était pas bon.

• Le Surcharge : Les IDS peuvent être pollués ou surchargés face à un trafic le plus difficile et lourd possible à analyser ce qui implique une quantité importante d'attaques envoyée afin de surcharger les alertes de l'IDS. Des conséquences possibles seront la perte de paquets, la saturation de ressources, et le déni de service.

• Consommation de ressources : La détection d'intrusion est très gourmande en ressources. En effet un système NIDS doit générer des journaux d'activité anormale sur le réseau.

• limitation des performances La Perte de paquets s'est produit parfois quand la vitesse de transmission dépasse largement la vitesse d'écriture d'un disques dur, ou même la vitesse de

traitement d'un processeur. Par conséquent, des paquets ne soient pas traités par l'IDS, et que certains d'entre eux soient néanmoins reçus par la machine destinataire.

• Temps de détection : C'est un élément important pour l'IDS qui exige un certain laps de temps afin de détecter ou de reconstruire une attaque(temps d'analyse,de réaction...).

• Vie privé : Enfin, Les IDS basès sur le comportement enregistrent toutes les activités des utilisateurs cette technique est liée à la vie privée. Pour tenter de dépasser ces limites, la détection d'intrusion fait l'objet de notre recherche.

1.3 Aspects d'apprentissage machine pour la détection d'intrusion

Faire face à des logiciels malveillants devient de plus en plus difficile, étant donné leur croissance constante en termes de complexité et de volume. L'une des approches les plus courantes dans la littérature est l'utilisation de techniques d'apprentissage automatique, pour apprendre automatiquement les modèles et les paradigmes derrière une telle complexité, et pour développer des technologies pour suivre la vitesse de développement de nouveaux logiciels malveillants. L'étude approfondie qui va être réalisée a mis en évidence des points intéressants qui mériteraient d'être traités plus en détail, en effet nous prétendons qu'ils peuvent être développés pour étendre et améliorer la recherche académique actuelle sur l'utilisation de l'apprentissage automatique pour l'analyse des malwares :

• L'objectif spécifique de l'analyse de malware présentée (c'est-à-dire la sortie).

• Comment faire l'échantillonnage des données ?

• Quels types de caractéristiques ils considèrent (c'est-à-dire, l'entrée) ?

• Quels algorithmes d'apprentissage machine qu'ils considèrent ?

Le domaine d'apprentissage machine inclut la construction d'un modèle à partir de données grâce à l'utilisation d'un algorithme. Ce modèle va au mieux généraliser, en représentant ou en approximant les données. Il permet, selon les données qu'on lui donne en input, de prédire

celles inconnues ainsi que de mieux comprendre celles existantes. Le domaine d'application du machine learning est très varié en citant par exemple la prédiction de valeurs financières, la détection d'intrusion dans le domaine de la sécurité informatique, le moteur de recherche influençable par le profil de l'utilisateur, la détection de vols de machine, l'implémentation d'un anti-virus et la cryptanalyse.

Le cycle de vie d'une implémentation d'un algorithme d'apprentissage machine est la suivant :

. Obtention et nettoyage des données.

. Réalisation du modèle.

. Phase d'apprentissage.

. Phase de validation.

. Phase d'exécution.

Nous allons aborder chaqu'une des étapes précédentes dans les sections suivantes.

1.3.1 Prétraitement de données

L'obtention de données en suffisance, représentatives selon le problème à résoudre n'est pas toujours facile car certaines informations sont plus coûteuses à obtenir que d'autres, par exemple, un header d'un paquet réseau est plus simple à obtenir qu'une information dans la partie data quand celle-ci est chiffrée. La deuxième étape est le nettoyage, appelé aussi le prétraitement de données qui consiste à enlever l'information inutile représentant du bruit. Il est souvent très compliqué à mettre en oeuvre une bonne connaissance des données à traiter dont nous avons besoin des techniques de selection des caractéristiques, un ensemble de méthodes pour choisir les données les plus pertinents à présenter à un IDS pour une meilleure précision du modèle, une optimisation du temps d'apprentissage.

1.3.2 Phase d'apprentissage

L'ensemble selectionné des caractéristiques forme les données d'entrainement permettant d'exécuter la phase d'apprentissage du modèle et ceci permet d'ajuster les paramètres du modèle. Après avoir entraîné le modèle, il est important de le valider pour éviter le sur-apprentissage.

1.3.3 Le sur-apprentissage

Le sur-apprentissage a lieu quand le modèle prédit correctement les données présentées lors de la phase d'apprentissage mais a des mauvais résultats lors de la phase d'exécution. Pour éviter le sur-apprentissage, on peut utiliser l'une des méthodes suivantes :

. Estimer empiriquement le nombre de périodes de la phase d'apprentissage pour qu'il n'y ait pas de sur-apprentissage.

. Le early stopping est une technique qui consiste à diviser l'information en deux parties.La première partie permet d'entraîner le modèle, la deuxième permet de le tester.

Nous procédons ainsi en le réalisant plusieurs fois par surveiller les erreurs. Au début, les fautes vont diminuer mais, petit à petit, les erreurs sur la partie test vont commencer à augmenter lorsqu'il y aura un sur-apprentissage. Et c'est à ce moment qu'il faut arrêter l'apprentissage ainsi que sur le temps nécessaire pour l'apprentissage. Nous allons discuter sur d'autres méthodes existantes liée à notre modèle présentées dans le chapitre 4.

1.3.4 Phase de validation

Apres avoir évalué le modèle, il est inmportant de pouvoir quantifier ses performances. Pour se faire, nous comparerons modèles sur le même jeu de données. En effet, certains sont plus adaptés pour certains problèmes. Néanmoins, ça n'inclut pas la taille du modèle ni son temps d'apprentissage ou son temps d'exécution. La technique receiver operating characteristic (ROC) est largement utilisée et permet de tester une structure.

1.3.5 Types du modèle

Il existe plusieurs types de modèles. Outre le fait qu'ils sont différenciables par leur côté supervisé ou non supervisé, ils le sont aussi par leur côté classification ou régression. Le premier classifie les données et le deuxième prédit des valeurs pour chaque donnée. Puisque le problème de l'IDS semble être un problème de classification, ce qui suit est une liste nonexhaustive de modèles de classification qui pourraient être envisagées pour la mise en place d'un IDS.

Les apprentissages supervisés utilisent généralement des datasets représentant des attaques connues. Les apprentissages non supervisés se basent uniquement sur les comportements des utilisateurs pour détecter une variation par rapport aux comportements normaux qui représentent une attaque. Inversement, pour des attaques inconnues, les algorithmes supervisés voient une réduction drastique de leur efficacité contrairement aux algorithmes non supervisés.

Ceci peut s'expliquer par le fait que, puisque les algorithmes non supervisés ne font que partitionner des données, les attaques sont toujours inconnues. En effet, étant donné que ces derniers ne font que regrouper des données proches, ils ne savent pas quand elles représentent une attaque. Bien que les algorithmes supervisés et non supervisés obtiennent des résultats semblables pour des attaques inconnues, les modèles non supervisés sont préférés pour leur robustesse.

En effet, ceux-ci ne changent pas leur taux de détection selon que l'intrusion est connue ou non. De plus, ils n'ont pas d'oracle leur disant à quelle classe appartient telle donnée donc ils sont donc plus indépendants que les techniques supervisées et leur point fort est la classiffication d'attaques appartenant à un label inconnu.

En effet, on ne doit pas leur présenter l'ensemble des labels. De plus, il est parfois très dur de classifier une information dans une classe, c'est pourquoi cette méthode est parfois préférée pour les IDS.

Par ailleurs, les algorithmes supervisés et non supervisés peuvent être combinés pour obtenir un apprentissage semi-supervisé où on utilise les avantages des deux techniques précédentes

pour réaliser un IDS. Cependant, une hypothèse sur les IDS supervisés reste à lever. Dans notre recherche nous avons intéressé par l'apprentissage supervisé dans l'approche décetralisée proposée sauf pour les deux approches centralisées, car pour comparer les résultats par rapport à l'état de l'art qui concentre sur les approches distribuées fédérées basé sur des réseaux de neurones artificiels supervisé ce qui implique notamment d'utilisation d'un modèle supervisé.

1.3.5.1 Les encodeurs automatiques

Les auto-encodeurs sont un type spécifique de réseaux neuronaux à anticipation où l'entrée est identique à la sortie. Ils compressent l'entrée dans un code de dimension inférieure, puis reconstruisent la sortie à partir de cette représentation. Le code est une compression compact de l'entrée, également appelé représentation en espace latent. Selon la Figure1.3 un encodeur automatique est composé de trois composants soient encodeur, code et décodeur. Le codeur compresse l'entrée et produit le code, le décodeur reconstruit ensuite l'entrée uniquement à l'aide de ce code.

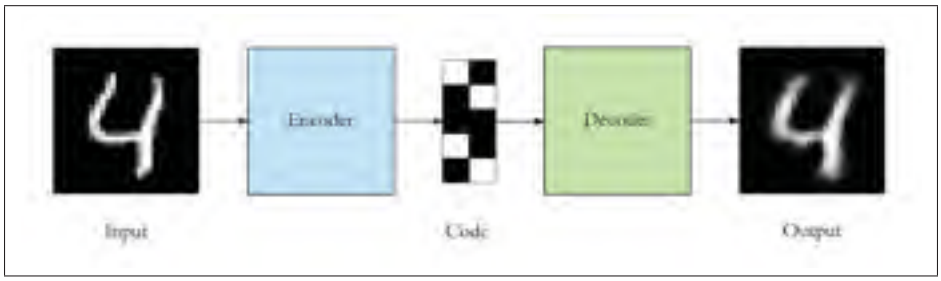


Figure 1.2 Les composants d'un autoencodeurs.

Pour construire un auto-codeur, nous avons besoin de trois éléments : Une méthode de codage, une méthode de décodage et une fonction de perte pour comparer la sortie à la cible. Nous allons les explorer dans la section suivante.

Les autoencodeurs sont principalement un algorithme de réduction de dimensionnalité (ou compression) avec quelques propriétés importantes :

• Données spécifiques : les auto-encodeurs ne peuvent que compresser des données similaires à celles sur lesquelles ils ont été formés. Comme ils apprennent des fonctionnalités spécifiques aux données d'apprentissage données, ils diffèrent d'un algorithme de compression de données standard tel que gzip. Nous ne pouvons donc pas nous attendre à ce qu'un codeur automatique formé à l'aide de chiffres manuscrits compresse les photos de paysages.

• La perte des données : La sortie de l'auto-codeur ne sera pas exactement la même que l'entrée, ce sera une représentation proche mais dégradée.

• Apprentissage non supervisé : Les autoencodeurs sont considérés comme une technique d'apprentissage non supervisée car ils n'ont pas besoin d'étiquettes explicites pour s'entraîner. Mais pour être plus précis, ils sont auto-supervisés car ils génèrent leurs propres étiquettes à partir des données d'apprentissage.

1.3.5.2 Architecture

Explorons les détails de l'encodeur, du code et du décodeur. Le codeur et le décodeur sont tous deux des réseaux de neurones à anticipation entièrement connectés, essentiellement les réseaux à réseau alternatif. Le code est une couche simple d'un réseau de neurones artificiels avec la dimensionnalité de notre choix. Le nombre de nœuds dans la couche de code (taille du code) est un hyperparamètre que nous avons défini avant de former l'auto-codeur.

La Figure1.3 est une visualisation plus détaillée d'un autoencoder. Tout d'abord, l'entrée passe par le codeur, qui est un RNA entièrement connecté, pour produire le code. Le décodeur, qui a la structure de réseau de neurones similaire, produit alors la sortie uniquement en utilisant le code. L'objectif est d'obtenir une sortie identique à l'entrée car l'architecture de décodeur est l'image miroir du codeur. Ce n'est pas une obligation mais c'est généralement le cas. La seule exigence est que la dimensionnalité de l'entrée et la sortie doit être la même. On peut jouer avec n'importe quoi au milieu. Il existe quatre hyperparamètres que nous devons définir avant de construire un autoencoder :

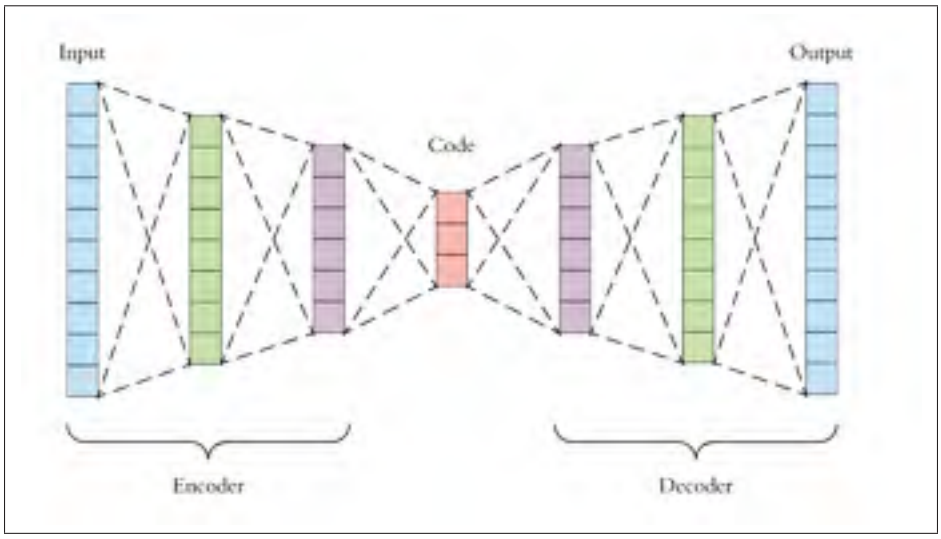


Figure 1.3 Architecture d'un encodeur automatique

• Taille du code : Nombre de noeuds dans la couche intermédiaire. Une taille plus petite entraîne plus de compression.

• Nombre de couches : L'auto-codeur peut être aussi profond que nous le souhaitons.

• Nombre de noeuds par couche : L'architecture à codeur automatique sur laquelle nous travaillons est appelée un codeur automatique à empilement, car les couches sont empilées les unes après les autres. Les auto-encodeurs généralement empilés ressemblent à un «sandwitch». Le nombre de nœuds par couche diminue à chaque couche suivante du codeur et augmente dans le décodeur. De plus, le décodeur est symétrique au codeur en termes de structure de couche. Comme indiqué ci-dessus, cela n'est pas nécessaire et nous contrôlons totalement ces paramètres.

• Fonction de perte : Les auto-encodeurs sont entraînés de la même manière que les ANN via la rétropropagation, nous utilisons soit l' erreur quadratique moyenne (mse), soit la crossentropie binaire.

1.3.5.3 Mise en œuvre

Nous avons un contrôle total sur l'architecture de l'auto-codeur. Nous pouvons le rendre très puissant en augmentant le nombre de couches, de noeuds par couche et surtout la taille du code. L'augmentation de ces hyperparamètres permettra à l'auto-codeur d'apprendre des codages plus complexes. Mais il faut faire attention à ne pas le rendre trop puissant. Sinon, l'auto-codeur apprendra simplement à copier ses entrées dans la sortie, sans apprendre aucune représentation significative. Cela imitera simplement la fonction d'identité. L'encodeur automatique reconstruira parfaitement les données d'apprentissage, mais il surchargera sans pouvoir généraliser de nouvelles instances.

C'est pourquoi nous préférons une architecture «sandwitch» et gardons délibérément la taille du code. Comme la couche de codage a une dimensionnalité inférieure à celle des données d'entrée, l'auto-codeur est dit incomplet. Il ne pourra pas copier directement ses entrées dans la sortie et sera obligé d'apprendre des fonctions intelligentes. Si les données d'entrée ont un motif, par exemple, dans le domaine de reconnaissance des formes le chiffre «1» contient généralement une ligne plutôt droite et le chiffre «0» est circulaire, il va apprendre ce fait et le coder sous une forme plus compacte. Si les données d'entrée étaient complètement aléatoires sans aucune corrélation interne ni dépendance, un autoencodeur incomplet ne pourra pas les récupérer parfaitement contrairement au monde réel où il y a beaucoup de dépendance.

1.3.5.4 Cas d'utilisation

Les autoencodeurs ne sont pas largement utilisés dans les applications réelles, cependant ils ont trois cas d'utilisation courants :

• Réduction de la dimensionnalité : La visualisation de données haute dimension est un défi. Le t-SNE est la méthode la plus couramment utilisée, mais se heurte à un grand nombre de dimensions (généralement supérieur à 32). Les auto-encodeurs sont donc utilisés comme étape de prétraitement pour réduire la dimensionnalité, et cette représentation compressée est utilisée par t-SNE pour visualiser les données dans un espace 2D.

• Variation Autoencoders (VAE) : Il s'agit d'un cas d'utilisation plus moderne et plus complexe des auto-encodeurs et nous les couvrirons dans un autre article. Mais pour résumer rapidement, VAE apprend les paramètres de la distribution de probabilités en modélisant les données d'entrée, au lieu d'apprendre une fonction arbitraire dans le cas des auto-encodeurs vanilles. En échantillonnant des points de cette distribution, nous pouvons également utiliser la VAE en tant que modèle génératif.

• Autoencodeur débruitant(Denoising autoencoder) : Les contraintes de régularisation imposées à l'encodeur automatique l'empêchent de simplement copier l'entrée vers la sortie et de surajuster les données, de plus le décrochage présenté sur les entrées fait de l'encodeur automatique un cas particulier d'encodeur automatique de débruitage dont nous allons l'appliquer sur le modèle centralisé avec NSL-KDD. Ce type est fait pour reconstruire l'entrée d'une version altérée et déformée de lui-même, obligeant le codeur automatique à apprendre encore d'avantage de propriétés des données.

1.4 Apprentissage fédéré

Suite aux événements récents liés à la protection de la vie privée par diverses méthodes de collecte de données par de grandes entreprises, il est important de réfléchir à des moyens alternatifs de collecte de données. L'apprentissage automatique a conduit à des avancées majeures dans divers domaines, tels que le traitement du langage naturel. Une grande partie de ce succès repose sur la collecte d'énormes quantités de données. Par exemple, l'un des derniers modèles Detectron de Facebook pour la détection d'objets a été formé sur 3,5 milliards d'images provenant d'Instagram.

Pour certaines applications d'apprentissage automatique, le besoin de collecte de données peut être extrêmement envahissant pour la vie privée. Cela se fait généralement avec l'apprentissage profond par exemple avec des réseaux de neurones récurrents. Un tel modèle pourrait être utilisé pour améliorer les résultats de la reconnaissance vocale, mais aussi pour prédire le mot suivant saisi sur un téléphone mobile afin d'aider les utilisateurs à interagir plus rapidement. Dans les deux cas, il serait avantageux de s'entraîner directement sur ces données au lieu d'utiliser du

texte provenant de Wikipedia. Cela permettrait de construire un modèle sur la même distribution de données que celle utilisée pour effectuer des prédictions.

Toutefois, la collecte directe de ces données est une idée terrible car elles sont extrêmement privée. Les utilisateurs ne veulent pas envoyer tout ce qu'ils tapent sur un serveur. L'envoi de versions aléatoires des données d'origine au serveur, sur la base des idées de Différential Privacy, est une solution potentielle à ce problème.

Notre solution proposée est l'apprentissage fédéré (federated learning), une nouvelle approche de l'apprentissage automatique dans laquelle les données d'apprentissage ne quittent pas du tout l'ordinateur des utilisateurs. Au lieu de partager leurs données, les utilisateurs calculent eux-mêmes leur poids en utilisant les données disponibles localement. C'est un moyen de former un modèle sans inspecter directement les données des utilisateurs sur un serveur. Ce point offre une introduction à l'apprentissage fédéré et aux défis qui se posent dans ce contexte.

1.4.1 Méthode principale

L'apprentissage distribué de réseaux de neurones est à ce jour la principale méthode d'application de l'apprentissage fédéré de McMahan, Moore, Ramage, Hampson, Arcas, (2017). Un modèle unique est défini et partagé à l'ensemble des agents qui se reposant sur une architecture client/serveur, la tâche d'apprentissage de ce modèle est réalisée par un ensemble d'appareils communément appelés clients. Un serveur central coordonne l'apprentissage. Le serveur sélectionne aléatoirement des clients, auxquels il transmet les poids actuels du modèle global(A), comme le représente la Figure 1.4. Chacun de ces clients, en prenant comme paramètres initiaux les poids qui entraîne indépendamment le modèle à partir de ses propres données (B). Le serveur agrège les résultats en effectuant la moyenne (pondérée par la taille des jeux de données locaux) des poids résultants de chaque entrainement indépendant (C). Il actualise ainsi les poids du modèle global. Ces étapes constituent un «tour de communication». Elles sont répétées itérativement afin de réaliser l'apprentissage du modèle.

Figure 1.4 Apprentissage fédéré d'un modèle de réseau de neurones tirée de Godard et al., (2019)

Le serveur forme le modèle initial sur les données disponibles au préalable. Le modèle initial est envoyé à un nombre sélectionné de périphériques clients éligibles. Le critère d'éligibilité permet de s'assurer que l'expérience de l'utilisateur n'est pas gâchée pour tenter de construire le modèle. Un nombre optimal de périphériques clients sont sélectionnés pour participer au processus d'apprentissage. Une fois les données utilisateur traitées, les mises à jour du modèle sont partagées avec le serveur. Le serveur agrège ces gradients et améliore le modèle global. Toutes les mises à jour de modèles sont traitées en mémoire et persistent pendant une très courte période sur le serveur. Le serveur renvoie ensuite le modèle amélioré aux périphériques clients participant à la prochaine session d'apprentissage. Après avoir atteint le niveau de précision souhaité, les modèles intégrés à l'appareil peuvent être ajustés pour la personnalisation de l'utilisateur. Ensuite, ils ne sont plus éligibles pour participer à l'apprentissage. Tout au long du processus, les données ne quittent pas le périphérique du client.

1.4.2 Approche décentralisée

Les approches classiques d'apprentissage automatique nécessitent la centralisation des données d'apprentissage sur une seule machine ou dans un centre de données. En effet, Google a mis en place l'une des infrastructures cloud les plus sécurisées et les plus robustes pour le traitement de ces données afin d'améliorer nos services. En revanche, l'apprentissage fédédré utilise une approche décentralisée pour former le modèle à l'aide des données de l'utilisateur (sensibles à la confidentialité). En bref, les méthodes d'apprentissage traditionnelles utilisaient l'approche consistant à «transformer les données en code» au lieu de «coder en données».

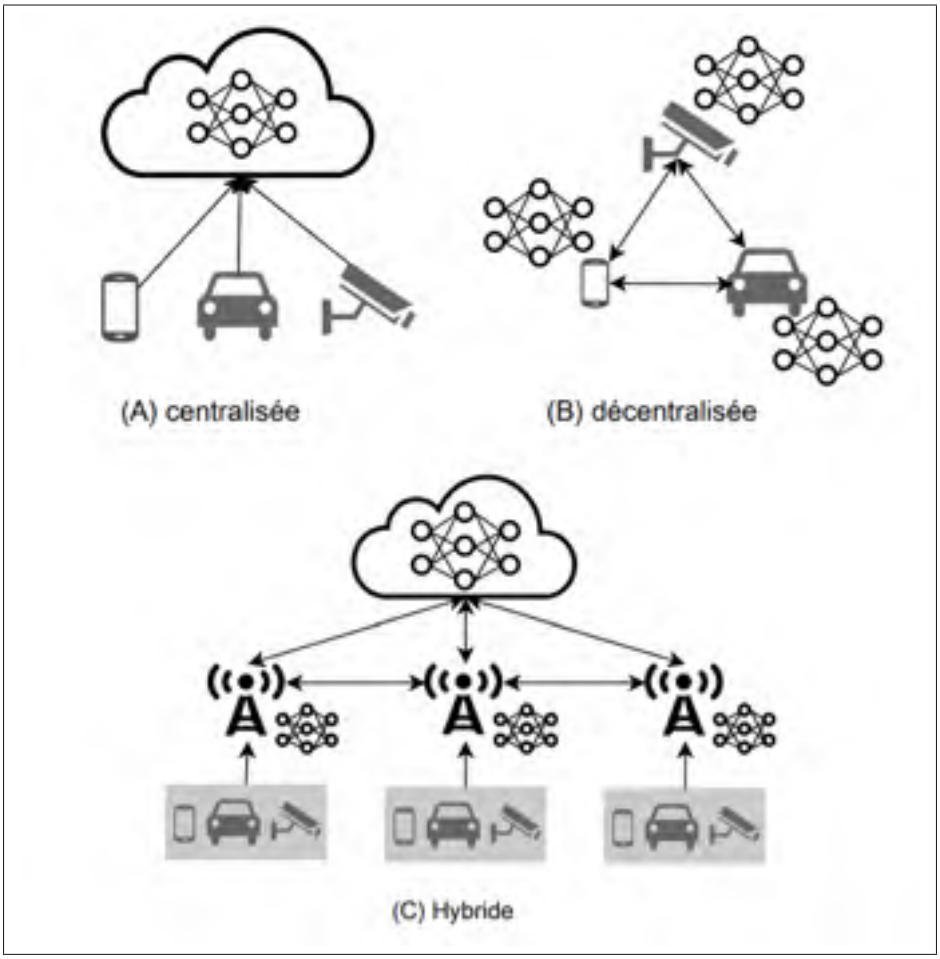


Figure 1.5 Les modes d'architecture de l'apprentissage distribué
tirée de Zhou et al., (2019)

l'apprentissage fédéré est tout au sujet de cette dernière approche est le mieux adapté au taches suivantes :

• Les étiquettes de tâches ne nécessitent pas d'étiquettes humaines, mais sont naturellement dérivées de l'interaction de l'utilisateur.

• Les données d'apprentissage sont sensibles à la confidentialité.

• Les données sont trop volumineuses pour pouvoir être collectées de manière centralisée et réalisable. Au début, l'entrainement des données avec l'approche décentralisée semble assez simple et similaire aux approches distribuées classiques. Cependant, il existe des différences majeures par rapport aux applications des centres de données où les données d'apprentissage sont réparties sur de nombreuses machines.

• Échantillons d'entraînement non équilibrés et non IID : Les données obtenues auprès de différents ensembles de données locaux d'utilisateurs ne seront pas représentatives de la répartition de la population et varieront d'un utilisateur à l'autre, créant ainsi des données de déséquilibre pour la phase d'apprentissage.

• Nombre élevé de clients : Comme les algorithmes d'apprentissage en profondeur exigent énormément de données, les applications qui utilisent l'apprentissage fédéré nécessitent beaucoup de clients. Les données de ces nombreux utilisateurs seront bien plus grandes que les données généralement stockées de manière centralisée.

• Connexions réseau lentes et non fiables : Les périphériques clients (tels que les smartphones) ont une bande passante réseau limitée.En raison de la vitesse variable de téléchargement entre les régions et les pays, les téléchargements requis pour l'apprentissage fédéré seront très lents par rapport à l'apprentissage traditionnel sur machine distribuée dans des centres de données où les communications entre les nœuds sont très rapides et les messages ne sont pas perdus.

• Les appareils clients ne sont pas toujours disponibles pour participer à une phase d'apprentissage. Les conditions optimales telles que l'état de charge, la connexion à un réseau Wi-Fi illimité, l'inactivité, etc. ne sont pas toujours réalisables.

• L'apprentissage fédéré intègre la préservation de la vie privée avec une agrégation réparties sur une large population.

1.4.3 Méthodes de compression

Il est clair que les communications entre le client et le serveur entraînent un surcoût, car la vitesse des connexions réseau est peu fiable et lente. Les RNA ont de millions de paramètres et l'envoi des mises à jour pour un aussi grand nombre de valeurs sur un serveur entraîne des coûts de communication énormes avec un nombre croissant d'utilisateurs et d'itérations. Afin de réduire les coûts de communication sur la liaison montante, McMahan et al., (2016), proposent deux méthodes décrites dans le présent document. Ce sont les techniques de compression qui codent les mises à jour avec moins de bits, car seules les mises à jour sont communiquées au serveur pour l'établissement de la moyenne.

• Mises à Jour esquissées : Dans cette méthode, chaque utilisateur calcule sa mise à jour après avoir appris ses données locales, puis avant d'envoyer les mises à jour au serveur, les mises à jour sont compressées à l'aide d'une combinaison de techniques de quantification d'un échantillonnage décrit dans l'article. En d'autres termes une mise à jour à partir d'un espace restreint de dimension inférieure.

• Mises à jour structurées : Ce deuxième type fait la compression en utilisant un masque aléatoire sur les mises à jour pour, et en envoyant uniquement les entrées non nulles. où nous apprenons une mise à jour complète du modèle, puis la compressons avant de l'envoyer au serveur.

Ces deux méthodes peuvent être utilisées pour réduire la surcharge de communication et réduire également la taille des mises à jour pour chaque partie, ce qui la rend fiable même à des vitesses de téléchargement réduites.

1.4.4 Aggrégation sécurisée

La question à laquelle nous voulons tous répondre est-ce que les données communiquées via un apprentissage fédéré sont vraiment anonymes et sécurisées ? Il existe principalement deux méthodes, à savoir l'agrégation sécurisée et la confidentialité différentielle, pour que les données communiquées restent anonymes. L' aggregation sécurisée utilise le protocole de calcul multi-parties sécurisé et le cryptage pour rendre les mises à jour des périphériques individuels non contrôlables par un serveur ou les mises à jour des clients sont résumées de manière sécurisée dans une seule mise à jour agrégée sans révéler aucun composant individuel du client, même au serveur en simulant de manière cryptographique un tiers de confiance.

L'aggregation sécurisée est un protocole interactif à quatre tours au cours de la phase de génération de rapports d'une série de données fédéré, ce qui signifie qu'il croîtra de manière quadratique avec le nombre d'utilisateurs, notamment le coût de calcul pour le serveur. À chaque tour de protocole, le serveur rassemble les messages de tous les périphériques du tour FL, puis utilise l'ensemble des messages de terminal pour calculer une réponse indépendante (agrégation finale) à renvoyer sur chaque terminal. Ce protocole est robuste pour une fraction importante d'abandons d'appareils, ce qui est peut-être le cas si la connexion réseau est mauvaise ou si le téléphone n'est plus inactif. Les deux premiers tours constituent une phase de préparation, dans lequel les secrets partagés sont établis et au cours de laquelle les périphériques qui abandonnent ne verront pas leurs mises à jour incluses dans le modèle globale.

Le troisième cycle constitue une phase de validation au cours de laquelle les appareils téléchargent des mises à jour de modèles masquées de manière cryptographique et le serveur accumule une somme des mises à jour masquées. La mise à jour de modèle de tous les appareils ayant terminé cette étape sera incluse dans la dernière mise à jour du protocole agrégé, sinon l'agrégation globale échouera. La dernière étape du protocole constitue une phase de finalisation, pendant lesquels les périphériques révèlent suffisamment de secrets cryptographiques pour permettre au serveur de démasquer la mise à jour de modèle agrégée. Tous les appareils engagés ne sont

pas obligés de terminer cette ronde. tant que le nombre de périphériques qui ont commencé à utiliser le protocole a survécu à la phase de finalisation, tout le protocole a réussi.

En utilisant des techniques de cryptographie, il est possible de s'assurer que les mises à jour des personnes ne peuvent être lues que lorsque suffisamment d'utilisateurs ont soumis des mises à jour. Cela rend les attaques de type Man in the middle beaucoup plus difficiles, car un attaquant ne peut pas tirer de conclusions sur les données d'entraînement en fonction de l'activité réseau interceptée d'un utilisateur individuel.

1.4.5 Étude de cas sur l'apprentissage fédéré : Gboard de google

Nous présentons une étude de cas d'amélioration des suggestions sur Gboard effectuée par Google. Yang et al., (2018) ont utilisé l'apprentissage fédéré pour les suggestions de requêtes de recherche sur Gboard. L'objectif était d'améliorer le taux de clics des requêtes en prenant en compte les suggestions du modèle de base et en supprimant les suggestions de faible qualité via le modèle.

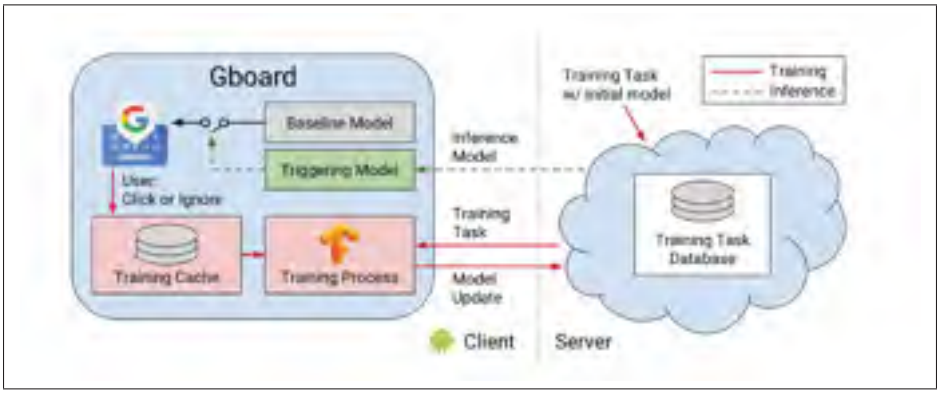


Figure 1.6 Une étude de cas pour l'amélioration des suggestions sur le Gboard de Google tirée de Yang et al., (2018)

Le cas d'utilisation consiste à former un modèle qui prédit si les suggestions de requête sont utiles, afin de filtrer les requêtes moins pertinentes. Les données d'apprentissage collectées pour ce modèle en observant les interactions de l'utilisateur avec l'application lors de la formulation d'une suggestion de requête à un utilisateur, un tuple (caractéristiques ; étiquette) est stocké dans

un cache d'apprentissage intégré, une base de données SQLite. Ici, les fonctionnalités sont une collection d'informations liées à la requête et au contexte et l'étiquette est une action utilisateur de clicked, ignored. Ces données sont ensuite utilisées pour l'apprentissage sur l'appareil et l'évaluation par les serveurs. Le modèle est généralement construit pendant la nuit lorsque le téléphone est en charge, en veille et connecté au réseau WiFi.

Le modèle de base est un apprentissage machine basé sur un serveur qui génère des suggestions de requêtes en faisant correspondre les données saisies par l'utilisateur à un sous-ensemble intégré du graphique Google Knowledge. Il note ensuite ces suggestions en utilisant un réseau récurrent (généralement appelés simplement «LSTM» capable d'apprendre des dépendances à long terme) construit sur un corpus hors ligne de données de discussion en vue de détecter les candidats potentiels à une requête. Ce LSTM est entraîné à prévoir la catégorie KG d'un mot dans une phrase et renvoie des scores plus élevés lorsque la catégorie KG du candidat à la requête correspond à la catégorie attendue. Le candidat qui obtient le score le plus élevé du modèle de base est sélectionné et affiché sous forme de suggestion de requête (une impression). L'utilisateur clique ou ignore alors la suggestion et l'interaction des utilisateurs sont stockées dans une mémoire cache intégrée pour que FL puisse les utiliser pour l'apprentissage.

La tâche du modèle fédéré est conçue pour intégrer la requête candidate suggérée dans le modèle de base et déterminer si la suggestion doit être présentée à l'utilisateur. Ce modèle FL déclenche le modèle. Le résultat du modèle est un score pour une requête donnée, les scores les plus élevés signifiant une plus grande confiance de suggestion.

Les étapes à suivre pour l'entraînement et la mise à jour du modèle FL global sont les suivants :

Les participants à la phase d'apprentissage sont les clients (ou appareils) et le serveur FL, qui est un service distribué basé sur un cloud. Les clients annoncent au serveur qu'ils sont prêts à exécuter une tâche FL pour une population FL donnée . Une population FL est spécifiée par un nom unique au monde qui identifie le problème d'apprentissage, ou l'application, sur lequel on travaille. Une tâche FL est un calcul spécifique pour une population FL, telle qu'une phase d'apprentissage à effectuer avec des hyperparamètres donnés ou une évaluation de modèles

construit sur des données de périphériques locales. Le serveur sélectionne un certain nombre de clients pour exécuter la tâche FL.

• Le serveur indique aux périphériques sélectionnés le calcul à exécuter avec un plan FL, une fois le processus établi, le serveur envoie ensuite à chaque participant les paramètres de modèle global actuel et tout autre état nécessaire en tant que point de contrôle FL.

• Chaque participant effectue ensuite un calcul local en fonction de l'état global et de son ensemble de données local, puis envoie une mise à jour au serveur sous la forme d'un point de contrôle FL.

• Le serveur qui incorpore ces mises à jour éphémères est agrégé à l'aide de l'algorithme Federated Averaging dans son état global et le processus est répété jusqu'à la convergence. Lors de la convergence, un point de contrôle qualifié est utilisé pour créer et déployer un modèle auprès des clients pour l'inférence.

1.5 Applications

En tant qu'un mécanisme de modélisation novateur qui pourrait former un modèle uni sur les données de plusieurs parties sans compromettre la vie privée et la sécurité de ces données, l'apprentissage fédérée a une application prometteuse des ventes, financiers, et bien d'autres industries, dont les données ne peuvent pas être directement agrégées pour la construction des modèles d'apprentissage de la machine en raison de la préservation de la vie privéeè. Les caractéristiques de données impliquées dans le e-commerce comprennent principalement le pouvoir d'achat, utilisateur de préférences personnelles et les caractéristiques du produit. Dans les applications pratiques, ces trois caractéristiques de données sont susceptibles d'être dispersées entre trois départements ou entreprises différentes. Par exemple, on peut déduire le pouvoir d'achat d'un utilisateur de son épargne bancaire et sa préférence personnelle peut être analysée de ses réseaux sociaux, alors que les caractéristiques des produits sont enregistrés par un e-shop. Dans ce scénario, nous sommes confrontés à deux problèmes. Tout d'abord, pour la protection de la confidentialité des données, des barrières de données entre les banques, les sites

de réseaux sociaux et les sites e-shopping sont difficiles à briser, par conséquent, les données ne peuvent pas être agrégées pour former directement un modèle.

En second lieu, les données stockées dans les trois parties sont généralement hétérogènes, et les modèles traditionnels d'apprentissage de la machine ne peut pas travailler directement sur des données hétérogènes. Pour l'instant, ces problèmes ne sont pas résolus de manière efficace avec des méthodes d'apprentissage des machines traditionnelles, qui font obstacle à la popularisation et l'application de l'intelligence artificielle dans plusieurs champs nous croyons que l'apprentissage fédéré et l'apprentissage de transfert sont la clé pour résoudre ces problèmes.

Tout d'abord, en exploitant les caractéristiques de l'apprentissage fédérée, nous pouvons construire un modèle d'apprentissage de la machine à trois parties sans exporter les données d'entreprise, qui protège non seulement la confidentialité des données, mais fournit également à ses clients des services personnalisés et ciblés pour permette d'obtenir benefices mutuels. Pendant ce temps, nous pouvons tirer parti de l'apprentissage transfert pour résoudre le problème de l'hétérogénéité des données et briser les limites des techniques d'intelligence artificielle traditionnelles. Par conséquent, l'apprentissage fédérée offre un bon support technique pour nous construire une entreprise, données croisées et écosphère inter-domaines pour les grandes données et l'intelligence artificielle.

Enfin, les soins de santé à puce est un autre domaine que nous attendons bénéficieront grandement de l'augmentation des techniques d'apprentissage fédérées. Les données médicales telles que les symptômes de la maladie, des séquences de gènes, les rapports médicaux sont très sensibles et des données privées, mais médicaux sont difficiles à recueillir et ils existent dans les centres médicaux et les hôpitaux isolés. L'insuffisance des sources de données et l'absence d'étiquettes ont conduit à une performance insatisfaisante des modèles d'apprentissage de la machine. Nous prévoyons que si tous les établissements médicaux sont unis et partager leurs données pour former un grand ensemble de données médicales, les performances des modèles d'apprentissage automatique formés sur cette grande base de données médicale serait nettement améliorée.

1.6 Disscussion

L'apprentissage fédéré est non seulement une norme technologique mais aussi un modèle d'affaires quand nous rendons compte des effets du big data, la première pensée qui se présente est d'agréger les données ensemble, calculer les modèles grâce à un processeur à distance, puis télécharger les résultats pour une utilisation ultérieure. Cependant, avec l'importance croissante de la confidentialité des données et la sécurité des données et une relation plus étroite entre les bénéfices d'une entreprise et ses données, le modèle de cloud computing a été contestée. Cependant, le modèle d'apprentissage fédéré a fourni un nouveau paradigme pour les applications de données importantes. Lorsque les données isolées occupées par chaque institution ne parvient pas à produire un modèle idéal, le mécanisme d'apprentissage fédérée permet aux institutions et aux entreprises de partager un modèle uni sans échange de données.

1.7 Conclusion

Ce chapitre nous a permis d'aborder les techniques de détections d'intrusion et leurs fonctionnement et capacités. Nous avons éventuellement abordés les approches actuelles d'apprentissage automatique qui nécessitent la disponibilité de grands ensembles de données. Celles-ci sont généralement créées en collectant d'énormes quantités de données auprès des utilisateurs ce qui a poussé les recherches vers une approche décentralisée plus souple applée Federated learning. Dans le chapitre suivant nous présentons un état de l'art des nouvelles techniques et méthodes FL existantes et les travaux connexes pour la détection d'intrusion sur le réseau.

CHAPITRE 2

REVUE DE LA LITTÉRATURE

2.1 Introduction

Bien que la recherche fonctionne dans l'application de l'apprentissage profond ont actuellement prospéré dans plusieurs domaines avec quelques recherches prometteuses travaux autour de la cybersécurité en utilisant l'approche de l'apprentissage fédéré. Ce chapitre présente un état de l'art des travaux antécédents reliés à l'optimisation distribuée dans un apprentissage fédéré avec des ressources hétérogènes. Dans cette partie du projet, nous visons à établir un cadre pour des mécanismes de sécurité adaptables où l'accent sera mis sur la réalisation de compromis entre la consommation de ressources et l'exactitude de la détection. Tout d'abord, nous passerons en revue les spécifications existantes pour les différentes méthodes distribuées pour la détection d'intrusion ce qui permettra d'identifier les principales classes de caractéristiques contextuelles à prendre en considération.

2.2 Parallelisme et mèthodes distribuées

Lors de la phase d'apprentissage de modèles profonds sur des jeux de données très volumineux, nous devons ajouter d'avantage de parallélisme à nos calculs. La distribution d'opérations d'algèbre linéaire sur des processeurs graphiques ne suffit plus, et les chercheurs ont commencé à explorer comment utiliser plusieurs machines. En effet, il existe deux approches pour paralléliser l'appretissage profond ; le modèle parallèle et les données parallèles. Le modèle parallèle consiste à distribuer le modèle d'apprentissage et à placer ces parties sur différents nœuds de calcul. Par exemple, nous pourrions placer la première moitié des couches sur un processeur graphique et l'autre moitié sur un second ou divisez les couches en leur milieu et affectez-les à des GPU distincts. Bien que séduisante, cette approche est rarement utilisée dans la pratique en raison de la latence de communication lente entre les périphériques. Le parallélisme des données est plutôt intuitif car les données sont partitionnées sur des périphériques de calcul, et chaque

périphérique contient une copie du modèle d'apprentissage, appelée réplique ou parfois worker. Chaque réplique calcule des gradients sur sa partition de données et les gradients sont combinés pour mettre à jour les paramètres du modèle. Différentes manières de combiner les gradients conduisent à différents algorithmes et résultats, examinons-les de plus près dans cette section.

2.2.1 Méthodes de base

Dans cette section, nous allons décrire plusieurs algorithmes de base fondamentaux en commençant par le plus ancient Gradient Descent est un algorithme d'optimisation utilisé pour minimiser la fonction coût dans divers algorithmes d'apprentissage automatique. Il est essentiellement utilisé pour mettre à jour les paramètres du modèle d'apprentissage, par modifier le modèle en le déplaçant le long d'une pente d'erreur vers une valeur minimale cela donne à l'algorithme le nom de «descente de gradient».

En théorie et en pratique, la descente de gradient peut être considérablement accélérée par l'ajout d'un terme de momentum . Les idées d'accélération pour les méthodes de gradient dans l'optimisation convexe peuvent être reliées aux travaux de Polyak [19] et Nesterov [68]. Il existe trois variantes principales de descente de gradient en fonction du nombre de modèles d'entraînement utilisés pour calculer l'erreur ; qui est à son tour utilisé pour mettre à jour le modèle soit la descente de gradient stochastique (SGD), le Batch,et le mini-batch.

2.2.1.1 Descente de gradient stochastique

La mise à jour du modèle pour chaque exemple d'apprentissage signifie que la descente de gradient stochastique est souvent appelée un algorithme d'apprentissage automatique en ligne.

Pour une analyse théorique pour les fonctions convexes, nous renvoyons le lecteur à [66, 64, 65] et [87, 81] pour les problèmes SVM. Dans une étude récente [12], les auteurs décrivent de nouvelles orientations de recherche. Pour une discussion plus pratique axée, voir la référence [11]. Dans le contexte des réseaux de neurones, le calcul des gradients stochastiques est appelé rétropropagation [49]. Au lieu de spécifier les fonctions et ses gradients explicitement, la

rétropropagation est une manière générale de calcul du gradient la Performance de plusieurs algorithmes compétitifs pour les réseaux de neurones profonds a été comparé dans [70].

Une astuce commune qui a été observé pratiquement pour fournir des performances supérieures, est de remplacer l'échantillonnage aléatoire à chaque itération en passant par toutes les fonctions dans un ordre aléatoire. Cette commande est remplacé par un autre ordre aléatoire après chaque cycle [56]. La compréhension théorique de ce phénomène a été un problème ouvert de longue date, compris le travail de [40] qui l'a resolu récemment.

2.2.1.2 Descente de gradient par lots

La descente de gradient par lots est une variante de l'algorithme de descente de gradient qui calcule l'erreur pour chaque exemple du jeu de données d'apprentissage, mais met à jour le modèle uniquement après évaluation de tous les exemples d'apprentissage.

Récemment, nous avons distingué une tendance constante vers l'apprentissage par lots de taille large, même dans le centre de données Goyal et al., (2017). L'algorithme Federated Averaging [12] adopte aussi une approche similaire. Liu Hsieh, (2018), ont montré de leur recherche [2] que la stratégie optimale pour atteindre le taux de convergence optimale par l'accès aux données est de toujours choisir la taille du batch.

2.2.1.3 Descente de gradient par mini-batch

La descente de gradient par mini-lot est la variante de la descente de gradient recommandée pour la plupart des applications, en particulier pour l'apprentissage profond.

La taille du lot est un curseur sur le processus d'apprentissage, les petites valeurs donnent un processus d'apprentissage qui converge rapidement au détriment du bruit dans le processus d'apprentissage, par contre, les grandes valeurs donnent un processus d'apprentissage qui converge lentement avec des estimations précises du gradient.

Dans un article récent [20] ils ont proposé mS2GD une version parallèle de l'algorithme S2GD [101], est une méthode qui intègre un schéma de mini-batch pour améliorer la complexité théorique et pratique de la performance de la descente de gradient. Dans une autre recherche récente de Liu Takác, (2016), une méthode de descente de gradient semi-stochastique abrégé PS2GD [55] a été projetée avec un mini-lot pour améliorer à la fois la complexité théorique et les performances pratiques de la méthode de descente de gradient stochastique (SGD) générale.

Bien que le SGD en mini-batch soit l'une des méthodes d'optimisation stochastique les plus répandues pour la phase d'apprentissage des réseaux profonds, il indique une vitesse de convergence lente en raison du bruit important dans l'approximation du gradient. Peng Li ,(2019) ont résolu ce problème récemment dans [79] en construisant plus de méthodes efficaces de sélection des lots basée sur un échantillonnage typique, ce qui réduit l'erreur d'estimation du gradient en mode conventionnel minibatch SGD.

2.2.1.4 Descente de gradient semi stochastique

Dans [101] une méthode de la descente du gradient semi-stochastique (S2GD) a été proposé et analysé qui permet de tirer parti des avantages des deux algorithmes GD et SGD, elle hérite de la stabilité et de la rapidité de GD tout en préservant l'efficacité de SGD. Une autre méthde a été testé applée S2GD+ [101] et que sa performance est supérieure à toutes les méthodes que nous avons cité auparavant, y compris S2GD. Cependant, la complexité de cette méthode reste un problème ouvert car celles-ci sont utilisées pour estimer le changement de gradient, et cette direction combine l'ancien gradient et les nouvelles informations de gradient stochastique utilisées dans la mise à jour.

Parmi les méthode hybride entre S2GD et CD, S2CD [84] peut être vu comme une méthode semi stochastique à coordonées aléatoires, se comporte alors comme une descente de gradient avec un pas fixe proche de l'optimum.

2.2.2 Algorithmes distribués aléatoires

Ces dernières années ont vu une explosion de nouvelles méthodes aléatoires qui, dans une première approximation, combinent les avantages de SGD avec une convergence rapide de GD. La plupart de ces méthodes peut dire d'appartenir à l'une des deux classes les méthodes duales de la variété de descente à coordonnées aléatoires, et les méthodes primitives de la descente de gradient stochastique avec une variété de réduction de la variance.

2.2.2.1 Méthodes de descente à coordonnées aléatoires (RCD)

Bien que l'idée de coordonner la descente a été autour depuis plusieurs décennies dans divers contextes , les travaux de Nesterov [67] ont doté une stratégie de randomisation, acquis une importance particulière dans l'apprentissage machine et l'optimisation. De nombreux travaux de suivi étendu le concept de réglage proximative[9], au parallélisme d'un processeur unique [24] et de développer efficacement une accélération réalisable [6]. Toutes ces trois propriétés ont été connectées en un seul algorithme [5], auquel nous renvoyons le lecteur à un examen des premiers développements dans le domaine du RCD.

2.2.2.2 Méthodes stochastiques ascendente à coordonnées doubles

Lorsqu'un régularisateur est ajouté à la perte moyenne, il est possible d'écrire ses (Fenchel) variables duales qui réside dans l'espace de n dimension. Cette méthode [13] a acquis une large popularité auprès des praticiens, probablement en raison du fait que, pour un certain nombre de fonctions de perte, la méthode est sans la nécessité d'accorder des hyper-paramètres. Le travail [16] fut le premier à montrer que en appliquant le RCD [14] au problème duale, on résout aussi le problème primordial. Pour une comparaison théorique et computationnelle de l'application du RCD au primaire par rapport aux problèmes double, voir [39].

Une méthode qui fait parti des méthodes de la descente stochastique à coordonnée aléatoire appelé Quartz, a été développé dans [43]. Il a été récemment montré dans SDNA [14] que l'intégration des informations de courbure faible contenues dans les sous-espaces de dimension aléatoires

enjambés par quelques coordonnées peut parfois conduire à des accélérations spectaculaires. Des travaux [26, 38] interprètent la méthode SDCA dans l'établissement primaire seulement, c'est pourquoi cette méthode fonctionne comme méthode SGD avec une version de la propriété de réduction de la variance.

2.2.2.3 Descente stochastique à variance réduite

Nous passons maintenant à la deuxième classe des algorithmes probabilistes qui peuvent être généralement interprété comme des variantes de SGD, Avec une tentative de réduire la variance inhérente au processus d'estimation de gradient. Le premier algorithme notable de cette classe est le Gradient moyen stochastique (SAG) [18]. De plus, [45,44] ont modifié l'algorithme SAG pour obtenir SAGA, une estimation non biaisée des gradients. L'exigence de mémoire est toujours présente, mais la méthode simplifie considérablement l'analyse théorique, et donne une garantie de convergence un peu plus forte.

Un autre algorithme de la classe SGD des méthodes est la variance stochastique à faible gradient appelée SVRG [8]. Le SVRG a l'avantage qu'il ne possède pas les exigences de mémoire supplémentaires de SAG et SAGA, mais il a besoin de traiter tout l'ensemble de données. En effet, la comparaison SGD, ce qui rend généralement des progrès significatifs dans le premier passage à travers les données, SVRG ne fait aucune mise à jour que ce soit, car il a besoin de calculer le gradient complet. Ceci et plusieurs autres questions pratiques ont été abordés dans [41], ce qui rend l'algorithme concurrentiel avec SGD, et supérieur dans les versions ultérieures.

Les expériences de l'algorithme vanilla dans [101] suggèrent que SVRG correspond à SGD de base, et même le dépasse dans le sens où la variance des itérations semble être significativement plus faible pour SVRG. Toutefois, afin de tirer des conclusions significatives, il faudrait procéder à de nombreuses expériences et comparer avec les méthodes état de l'art en général équipés de nombreux heuristiques.

Il existe déjà des tentatives de combiner des algorithmes de type SVRG avec descente à coordonnées aléatoires [84]. Bien que ces travaux mettent en évidence certaines propriétés

théoriques intéressantes, les algorithmes ne semblent pas être pratique au moment dont plus de travail est nécessaire dans ce domaine. La première tentative d'unifier les algorithmes tels que SVRG et SAG / SAGA est apparu déjà dans le document de SAGA [45], où les auteurs interprètent SAGA comme un point médian entre la SAG et SVRG. Des travaux récents [4] présente aussi un algorithme général, qui récupère SVRG, SAGA, SAG et GD comme des cas particuliers, et obtient une variante asynchrone de ces algorithmes ce qui conduit à un nouveau procédé de SVRG accéléré connu sous le nom Katyusha et une verson accéléré de SVRG [17]. Dans l'étude récente de [2], ils ont claasée SAGA et d'autres existants qui ont utilisés des lots aléatoires sur des ensembles de données réelles. De plus, ils ont effectué également une analyse précise de comparer les règles de mise à jour différentes pour les méthodes de réduction de la variance, montrant que SAGA++ converge plus vite que SVRG en théorie.

2.2.2.4 Quasi-Newton Stochastic

Une troisième classe d'algorithmes sont les méthodes quasi-Newton Stochastic [26, 23] qui tentent d'imiter la méthode de BFGS de mémoire limitée (L-BFGS) [63], mais modéliser l'information de courbure locale en utilisant des gradients inexactes provenant de la procédure de SGD. Nous trouverons une récente tentative de combiner ces méthodes avec SVRG dans [63]. Les auteurs de [67] utilisent des progrès récents dans le domaine de l'inversion de la matrice stochastique [69] révélant de nouvelles connexions avec des méthodes quasi-Newton, et de concevoir une nouvelle méthode BFGS mémoire limitée stochastique travaillant avec SVRG.

2.3 Optimisation Fédérée

Dans l'optimisation fédérée où les données définissant l'optimisation sont inégalement répartis sur un très grand nombre de noeuds. Dans ce contexte, l'efficacité de la communication est de la plus haute importance et de minimiser le nombre de tours de communication sera l'objectif principal.

La plupart des recherches sur l'apprentissage distribué a toujours été fait dans le contexte d'un cadre central cluster / données hautement contrôlé, par exemple, avec un jeu de données divisé uniformément de façon équilibré (IID). Le multi-core et l'entrainement de multi-GPU distribué a été spécialement étudié dans le cadre de la reconnaissance vocale dans [33]. Les efforts en matière d'apprentissage décentralisé hautement distribuée, les données non équilibrées et non-IID est relativement récente ont été posées dans [32] avec une introduction de l'algorithme fédérée (FedAvg) et son application à un ensemble de vision par ordinateur (MNIST, CIFAR-10) et les tâches de modélisation linguistique (appliquée aux données de Shakespeare et Google messages). Il y a pour l'instant très peu d'expériences de la vie réelle que nous connaissons à l'exception du clavier de Google Gboard pour Android [36] et plus récemment la barre de suggestion d'URL de Mozilla [37].

Le problème d'optimisation fédérée dans le cadre des fonctions convexes a été étudié dans [3]. Les auteurs ont proposé une procédure d'optimisation descente de gradient qui réduit la variance stochastique SVRG avec mise à l'échelle du gradient par coordonnée à la fois local et global pour améliorer la convergence. Leur stratégie globale moyenne du gradient par coordonnée repose sur une mesure de la sparsification de coordonnées des jeux de données locaux des utilisateurs et est applicable uniquement dans le contexte des rares modèles linéaires-in-the-fonctions. Cette dernière hypothèse ne tient pas dans le contexte des réseaux de neurones pour les applications basées sur la parole.

Plusieurs améliorations de l'algorithme de FedAvg initial ont été proposées en mettant l'accent sur la sélection des clients [7], l'optimisation sous contrainte budgétaire [30] et la réduction des coûts de communication entre clients. Une stratégie de la moyenne dynamique d'un modèle robuste dérivée d'un concept basé sur un critère de divergence du modèle local a été récemment introduit dans [28].

Ce travail est aussi considéré parmi les contributions actuelles qui utilisent des stratégies efficaces pour réduire les coûts de communication inhérents à l'optimisation fédérée, en rajoutant un module de compression du modèle, par conséquent faisant baisser le nombre d'itérations durant

la phase d'apprentissage ce qui influt sur la performance du modèle pour atteindre des résultat plus élevés par rapport aux recherches existantes.

2.3.1 Catégorisation d'un apprentissage fédéré

Dans cette section, nous discutons de la façon de catégoriser l'apprentissage fédéré en fonction des caractéristiques de distribution des données. Étant donné que les données ne quittent jamais leurs locaux d'origine, l'apprentissage fédéré offre la possibilité à différents propriétaires de collaborer et de partager leurs données. Dans un article récent [27], les chercheurs Yang et al., (2019) ont envisagé les différentes configurations dans lesquelles cela peut se produire. Ils ont classé l'apprentissage fédérée à trois catégories « apprentissage fédéré horizontal», «apprentissage fédéré vertical» et « apprentissage transfer learning ». Nous prenons le cas de cadres d'apprentissage fédéré pour un scénario à deux entités soit deux banques régionales, bien qu'ils aient une clientèle ne se partagentt pas leurs données auront des espaces de fonctionnalités similaires car ils ont des modèles de gestion très similaires. Ils pourraient s'unir pour collaborer à un exemple d'apprentissage fédéré horizontal.

Dans le cadre de l'apprentissage fédéré vertical, deux entreprises offrant des services différents (banque et commerce électronique, par exemple), mais ayant une grande intersection de clientèle, pourraient trouver la possibilité de collaborer sur les différents espaces de fonctions qu'elles possèdent, ce qui se traduirait par de meilleurs résultats. Dans les deux cas, les propriétaires de données peuvent collaborer sans avoir à sacrifier la vie privée de leurs clients. En outre, un autre secteur d'activité qui pourrait en bénéficier est le secteur de la santé. Les hôpitaux et autres prestataires de soins de santé ont tout intérêt à pouvoir partager les données des patients à la fin de l'entrainement du modèle dans le cadre de la préservation de la vie privée.

2.3.2 Confidentialité de l'apprentissage fédéré

La vie privée est l'une des propriétés essentielles de l'apprentissage fédéré. Cela exige des modèles de sécurité et d'analyse pour fournir des garanties significatives de la vie privée. Dans

cette section, nous comparons les différentes techniques de confidentialité pour l'apprentissage fédérée. Nous identifions également des approches et des défis potentiels pour protéger la vie privée [27].

2.3.3 Secure Multiparty Calcul (SMC)

Le calcl u multipartie sécurisé est l'un des modèles de sécurité concernent de nombreuses parties et fournir une preuve de sécurité dans un cadre de simulation bien défini pour garantir la connaissance zéro complète, ce qui est, chaque partie ne sait rien, sauf son entrée et de sortie. Zéro connaissance est hautement souhaitable, mais cette propriété désirée nécessite généralement des protocoles de calcul complexes et ne peut être réalisé efficacement.

Il est possible de construire un modelde sécurité avec SMC en vertu des exigences de sécurité plus bas en échange d'efficacité [16]. Récemment, une étude [46] a utilisé le cadre SMC pour les modèles d'apprentissage automatique avec deux serveurs et des hypothèses semi-honnêtes. Les protocoles MPC sont utilisés dans [33]pour l'entrainement de modèle sans que les utilisateurs révélant des données sensibles. L'un de l'état de l'art des cadres SMC est Sharemind [8]. Les auteurs de [44] ont proposé un modèle [5, 21, 45] avec une majorité honnête en considérant la sécurité dans les deux hypothèss semi-honnêtes et malveillants. Ces travaux exigent que les données des participants soient implicitement partagées entre les serveurs.

2.3.4 Confidentialité différentielle

Une autre ligne de travail utilise les techniques de confidentialité différentielle [18] pour la protection des données personnelles [1, 12, 42, 61]. Les procédés de la vie privée différentielle, k-anonymat, et la diversification [3] Consiste à ajouter du bruit aux données, ou en utilisant les methodes de généralization pour masquer certains attributs sensibles jusqu'à ce que le tiers ne peut pas distinguer l'individu, therebymaking les données impossibles à restaurer pour protéger la confidentialité des utilisateurs. Cependant, la racine de ces méthodes exige toujours que les données sont transmises ailleurs, ce qui implique généralement un compromis entre la

précision et la vie privée. Dans [23], Les auteurs ont introduit une approche différentielle pour l'apprentissage fédéré afin d'ajouter une protection aux données côté client en se cachant les contributions du client pendant l'apprentissage.

2.3.5 Le chiffrement homomorphique

Le chiffrement homomorphique est également adoptée pour protéger la confidentialité des données utilisateur par l'échange de paramètres dans le cadre du mécanisme de chiffrement lors de l'apprentissage automatique [24, 26, 48]. Contrairement à la protection de la vie privée différentielle, les données et le modèle lui-même ne sont pas transmis, ils ne peuvent pas être devinés par les données de l'autre partie. Par conséquent, il y a peu de possibilité de perte au niveau des données brutes. Des travaux récents ont adopté le chiffrement homomorphique pour la centralisation des données et la formation sur le nuage [75 , 76]. Dans la pratique, le cryptage additivement homomorphic [2] est largement utilisé et les approximations polynomiales doivent être prises pour évaluer les fonctions non-linéaires dans les algorithmes d'apprentissage automatique, ce qui entraîne des compromis entre la précision et la préservation de la vie privée [35].

2.3.6 Perte d'information indirecte

Travaux pionniers de l'apprentissage fédérée exposent les résultats intermédiaires tels que les mises à jour des paramètres à partir d'un algorithme d'optimisation comme SGD [41]. Toutefois, aucune garantie de sécurité est fourni et la perte de ces gradients peuvent effectivement influencent une perte importantes d'information[51] quand il est exposé en même temps que la structure de données, comme dans le cas des pixels d'image. Les chercheurs ont examiné la situation dans laquelle un des membres d'un système d'apprentissage fédéré attaque par une action malveillance les autres en permettant l'insertion d'une porte dérobée pour en apprendre d'avantage sur les données des autres. Dans [6], les auteurs démontrent qu'il est possible d'insérer des portes arrière cachées dans un modèle global commun et proposent une

nouvelle méthodologie d'empoisonnement par modèle «contrainte-à l'échelle» pour réduire l'empoisonnement des données.

Dans [43], les chercheurs ont identifié des lacunes potentielles dans les systèmes de collaboration, où les données d'apprentissage utilisées par les différentes parties dans l'apprentissage collaboratif est vulnérable à l'inférence des attaques. Ils ont montré qu'un participant peut déduire l'adhésion contradictoire, ainsi que les propriétés associées à un sous-ensemble des données de formation. Ils ont également discuté des défenses possibles contre ces attaques.

Dans [62], les auteurs exposent un problème de sécurité potentiel associé à des échanges de gradient entre les différentes parties et proposent une variante sécurisée de la méthode de descente de gradient. Ils montrent qu'il tolère jusqu'à une fraction constante des travailleurs byzantins.

Les chercheurs ont également commencé à envisager une blockchain comme plateforme pour faciliter l'apprentissage fédérée. Dans [34], les chercheurs ont examiné un blockchain fédérée (BLOCKFL), où les mises à jour des modèles d'apprentissage local appareils mobiles sont échangés et vérifiés en tirant parti d'un blockchain. Ils ont considéré une génération de bloc optimale, l'évolutivité du réseau et des problèmes de robustesse.

2.4 Travaux Connexes

L'apprentissage fédéré permet à plusieurs parties de construire en collaboration un modèle d'apprentissage de la machine tout en gardant leur données d'apprentissage privées. En tant que nouvelle technologie, l'apprentissage fédéré a plusieurs fils ted d'originalité, dont certains sont ancrés sur les champs existants. Nous expliquons la relation entre l'apprentissage fédérée et d'autres concepts connexes à partir de plusieurs points de vue.

2.4.1 Apprentissage machine préservant la vie privée

l'apprentissage fédéré peut être considéré comme un apprentissage collaboratif décentralisé étroitement lié à la préservation la vie privée. De nombreux efforts de recherche ont été consacrés à ce domaine dans le passé, en citant [17, 67] des algorithmes d'arbre de décision multipartite sécurisé ont été proposés pour les données verticales. Vaidya et Clifton ont proposé des règles d'association sécurisée [65] et k-means sécurisés [66], un classificateur bayésien [64] pour les données partitionnées verticalement. Les travaux [25, 72] qui ont utilisé surtout le calcul multi-parties (SMC) pour garantir la confidentialité.

Mohassel Zhang, (2017), ont abordé le problème du gradient stochastique et ils ont également proposé des protocoles pour la préservation de la vie privée de régression logistique et les réseaux de neurones. Récemment, un travail de suivi avec un modèle à trois serveurs a été élaboré [44]. Aono, Hayashi, Phong, Wang, (2016), ont proposé un protocole de régression logistique sécurisée utilisant le cryptage homomorphique. Dans [51] les chercheurs ont utilisé le chiffrement homomorphique pour préserver la vie privée et d'améliorer la sécurité du système.

Avec les progrès récents en matière d'apprentissage profond, préservant la vie privée, l'inférence des réseaux de neurones reçoit également beaucoup d'intérêts de recherche [1, 22,77, 28, 40, 52, 54]. Shokri Shmatikov, (2015), ont constuit des réseaux de neurones pour les données partitionnées horizontalement avec des échanges de paramètres mis à jour.

Truex et al., (2019), ont présenté une approche hybride fédéré qui utilise à la fois la confidentialité différentielle et le SMC pour équilibrer ces compromis. La combinaison de la confidentialité différentielle avec le calcul multipartite sécurisé ce qui de réduire la croissance de l'injection de bruit à mesure que le nombre de parties augmente sans sacrifier la confidentialité, tout en maintenant un taux de confiance prédéfini. Il est considéré comme un système évolutive qui protège contre les menaces d'inférence et produit des modèles avec une grande précision.

2.4.2 Machine learning distribué

L'apprentissage fédéré à première vue est un peu similaire à l'apprentissage de la machine distribuée, car il couvre de nombreux aspects, y compris le stockage distribué des données d'apprentissage, le fonctionnement distribué des tâches informatiques, la distribution des résultats du modèle, etc. Le paramètre du serveur [50] est un élément typique dans l'apprentissage de la machine distribuée. En tant qu'outil pour accélérer le processus d'apprentissage, le serveur stocke des paramètres de données sur des nœuds distribués, qui alloue les ressources de données et de calcul par un noeud de planification centrale, de manière à construire un modèle plus efficace. Pour l'apprentissage fédéré horizontal, le nœud représente le propriétaire des données. Il a une autonomie totale pour les données locales, et peut décider quand et comment joindre l'apprentissage fédéré. Dans les paramètres serveur, le noeud central prend toujours le contrôle, si l'apprentissage fédéré est confronté à un environnement d'apprentissage plus complexe. Dans le domaine d'apprentissage automatique distribué, l'approche fédérée devra également traiter des données non-IID [71] dont les performances peuvent être considérablement réduits pour l'apprentissage fédéré. Les auteurs en réponse fourni une nouvelle méthode pour résoudre le problème similaire pour le transfert d'apprentissage.

2.4.3 Edge computing

l'apprentissage fédéré peut être considéré comme un système d'exploitation pour le edge computing, car il fournit le protocole d'apprentissage pour la coordination et la sécurité. Dans [73], les auteurs considérés comme classe générique des modèles de l'apprentissage machine qui ont été formés en utilisant des approches basées sur les gradients de descente. Ils analysent la convergence liée de descente de gradient distribué à partir d'un point de vue théorique, sur la base duquel ils proposent un algorithme de contrôle qui détermine le meilleur compromis entre la mise à jour locale et l'agrégation des paramètres globaux pour minimiser la fonction de perte en vertu d'un nombre important de ressources des données.

2.4.4 Les systèmes de base de données

Systèmes de base de données fédérée [57] sont des systèmes qui intègrent des unités de base de données multiples et gèrent le système intégré dans son ensemble. Le concept de base de données fédérée est proposée pour réaliser l'interopérabilité avec de multiples bases de données indépendantes. Il utilise souvent le stockage distribué pour les unités de base de données, et dans la pratique, les données de chaque unité de base de données est hétérogène. Par conséquent, il a beaucoup de similitudes avec l'apprentissage fédérée en termes de type et de stockage des données. Cependant, le système de base de données fédérée ne comporte aucun mécanisme de protection de la vie privée dans le processus d'interaction, et toutes les unités de base de données sont entièrement visibles au système de gestion. En outre, la mise au point du système de base de données fédérée est sur les opérations de base de données, y compris l'insertion, la suppression, la recherche et la fusion, etc.

2.5 Compromis entre la consommation de ressources et l'exactitude de la détection

Les approches existantes d'apprentissage fédéré ne tiennent pas compte des ressources et des données hétérogènes[31,10,12], ni de leur prise en compte par le trafic. Dans ce contexte, [91] a montré que les réponses retards ne sont pas considérés par les techniques existantes à travers une étude préliminaire qui consiste à quantifier l'impact potentiel des resources et des données hétérogène sur le temps d'apprentissage fédéré. En particulier, il y a deux approches principales pour entraîner un modèle FL soit le FL synchrone et le FL asynchrone. En FL synchrone, un certain nombre de données des clients sont interrogées à chaque époque d'apprentissage pour assurer la performance et la confidentialité des données.

Les récents algorithmes synchrones de FL se concentrent sur la réduction du temps total d'entraînement sans tenir compte des parties déja entrainées. Par exemple,[12] propose de réduire les coûts de communication réseau en effectuant de multiples mises à jour SGD localement et en mettant en lots les parties de données. [31] a réduit la consommation de bande

passante de communication par des mises à jour structurées et esquissées. De plus,[9] exploite une technique aléatoire pour réduire les itérations de communication.

La plupart des algorithmes FL asynchrones ne fonctionnent que pour les fonctions de perte convexes et ne permettent pas aux utilisateurs d'abandonner. Par exemple,[15] fournit une garantie de performance uniquement pour les fonctions de perte convexe avec hypothèse de retard limité. De même, des recherches [77,10] permettent un échantillonnage uniforme de données des clients et fournissent une garantie de performance pour les fonctions convexe. La comparaison des méthodes synchrones et asynchrones de descente de gradient distribué [78] suggère que FL devrait utiliser l'approche synchrone, car elle est plus efficace que les approches asynchrones [12,13].

FedCS [7] propose de résoudre la question de la sélection des clients par une approche fondée sur les échéances qui permet d'éliminer les entités qui répondent lentement. Toutefois, FedCS ne tient pas compte de la façon dont cette approche influe sur les facteurs contributifs du groupe des clients retardataires dans la phase d'apprentissage.

De même, un algorithme FL [73] a été proposé pour le cas d'utilisation sur des dispositifs à ressources limitées, mais il ne vise pas à traiter les parties qui traitent les autres parties comme étant des ressources limitées. En revanche, une autre recherche [2] a concentré sur des scénarios où des dispositifs à ressources limitées sont jumelés à des dispositifs à ressources élevées pour effectuer l'approche FL.

Dans ce contexte, une étude récente a montré que les réponses retards ne sont pas considérés par les techniques existantes à travers une étude préliminaire qui consiste à quantifier l'impact potentiel des resources et des données hétérogène sur le temps d'apprentissage fédéré [91]. Dans une approche hybride récente [92], les auteurs ont conçu un algorithme heuristique permettant de résoudre les problèmes de sélection des clients pour construire un bon modèle sur le serveur sous la limitation de la bande passante et le temps.

2.6 Comparaison et discussion

Cette section présente une étude comparative entre notre solution et les articles références de la littérature. Le tableau 2.1 compare notre approche FIDS à celles existantes par rapport à leurs objectifs de performances ciblés, à savoir,la sélection des clients, la gestion de transfert de données qui prend en considération la réduction de tours communication en assurant un nombre minimal des clients communiqués dans chaque tour, et le problème de resources limitées. Aucune des solutions existantes ne peut à la fois assurer une bon modèle global tout en utilisant des resources disponible efficacement. Dans notre approche, nous visons à atteindre simultanément ces objectifs afin d'optimiser notre FIDS et respecter la qualité de service exigée par le client tel que la minimisation du temps de réponse.

Tableau 2.1 Comparaison entre notre approche FIDS et les approches existantes

Approches	**Hétéro**	**Max Parallélisme**	**Min du Temps**	**NIDS**	**Resources limitées**
McMahan et al., (2017)	vrai	Vrai	Faux	Faux	Faux
Diro & Chilamkurti, (2017)	Faux	Faux	Faux	Vrai	Faux
Godard et al., (2019)	Vrai	Faux	Faux	Faux	Faux
Nishio & Yonetani, (2018)	Vrai	Faux	Vrai	Faux	Vrai
Wang et al., (2018)	Vrai	Vrai	Faux	Faux	Vrai
Chai et al., (2019)	Vrai	Faux	Vrai	Faux	Faux
Truex et al., (2019)	vrai	Vrai	Faux	Faux	Faux
Notre FIDS	Vrai	Vrai	Vrai	Vrai	Vrai

2.7 Conclusion

Les travaux ci-dessus tout se concentrer sur l'apprentissage fédéré dans lequel les interactions distribués par l'utilisateur, le coût de la communication dans une distribution massive, et la distribution des données hètérogènes et la fiabilité de détection sont quelques-uns des principaux facteurs d'optimisation. En outre, les données sont partitionnées horizontalement dans l'espace de données par les ID utilisateur ou ID de dispositif, par conséquent, ils font une ligne de recherche très liée à la préservation de la vie privée. Dans le chapitre suivant nous allons aborder la partie d'implémentation de notre solution proposée au cadre d'une approche décentralisé fédéré appliqué à un apprentissage supervisé en utilisant un type des réseaux de neurones artificiels très comnnu dans la litérature nommés réseau de neurones convolutifs.

CHAPITRE 3

SOLUTION PROPOSÉE

3.1 Introduction

Bien que l'algorithme d'apprentissage automatique a été utilisé de manière distribuée, l'apprentissage fédéré est différent de la façon dont l'apprentissage automatique est utilisé dans les centres de données. Beaucoup de garanties sur les distributions ne peuvent être faites et la communication est souvent lente et instable. Pour pouvoir effectuer efficacement l'apprentissage fédéré, les algorithmes d'optimisation peuvent être adaptés et divers schémas de compression peuvent être utilisés. La confidentialité peut être abordée à l'aide de la confidentialité différentielle et du cryptage. Comme le système est en général assez flexible, il peut être adapté pour permettre des modèles local. Bien qu'il y ait eu plusieurs articles sur Federated Learning, il est encore assez récent et l'industrie n'en a pas encore beaucoup utilisé.

En effet, l'apprentissage fédéré demeure aujourd'hui au stade expérimental et n'a encore été appliqué que dans des problèmes d'apprentissage supervisé. néanmoins, les expérimentations réalisées dans le cadre de cette étude n'ont permis pas d'aborder seulement l'aspect distribué de l'apprentissage fédéré mais une comparaison avec l'approche centralisée sera faite en utilisant plusieurs scénarios.

3.2 Module de prétraitement

Lors de l'évaluation de l'IDS, l'un des défis auxquels sont confrontés les chercheurs est de trouver un ensemble de données appropriées. L'acquisition d'un ensemble de données réelles du monde qui représente le trafic passant à travers le réseau sans aucune sorte d'anonymisation ou modification est un problème qui a été continuellement rencontré par la communauté de la recherche en matière de cybersécurité [43]. Même dans les cas où les données sont autorisés à être publié ou partagé pour un usage public, il sera fortement anonymisées ou gravement altérée. Cela entraînera un grand nombre de composants de données essentielles qui sont considérées

comme essentielles pour les chercheurs à perdre ou ne sont plus fiables. Pour cette raison, de nombreux chercheurs ont décidé d'utiliser des ensembles de données simulées comme l'ensemble de données KDDCup'99 bien connu [44], ou l'un de ses contemporains l'ensemble de données NSL-KDD [45]. Récemment il y a eu un effort significatif pour essayer de développer des ensembles de données qui reflètent le monde réel, l'Institut canadien pour la cybersécurité (CIC) a publié un ensemble de données de détection d'intrusion nommé CIC-IDS2017 [46], qui ressemble à la vraie capture des paquets (PCAPs) du monde réel. Les données réalistes provient généralement de plates-formes hétérogènes et peuvent être bruyantes, redondantes, incomplètes et incohérentes [50]. Ainsi, il est important de transformer les données brutes en un format adapté à l'analyse et la découverte de connaissances. Dans cette recherche, l'étape de pré-traitement consiste à enlever les valeurs aberrantes, les instances redondantes et transformation des données, ainsi que la normalisation des données.

3.2.1 Analyse des données

Dans le présent projet, les échantillons sont prélevés à partir de deux ensembles de données bien connues NSL-KDD et IDS2017 en passant par la phase de pré-traitement des données qui est de la plupart du temps une étape essentielle dans l'exploration de données.

3.2.1.1 KDDCup'99

L'ensemble de données KDDCup'99 a été généré par Lincoln Labs [82] basé sur une simulation d'un typique US Air Force LAN, y compris plusieurs semaines de données de vidage TCP brutes avec des activités normales et divers types d'attaques. Chaque connexion est décrite par 41 caractéristiques discrètes et continues qui peuvent être essentiellement regroupées en trois catégories soient les fonctions de base de connexion individuelle, les caractéristiques de contenu au sein d'une connexion, et les caractéristiques du trafic qui sont calculés en utilisant des fenêtres temporelles de 2s.

Le trafic simulé contenu dans cet ensemble de données comprend une variété d'intrusions sous différentes distributions de probabilité, et ils peuvent être divisés en quatre grandes catégories :

• DoS : Déni de service.

• PROBE : Sonder - Surveillance pour la collecte de l'information ou des vulnérabilités connues.

• R2L : Local à distance - Accès non autorisé à partir d'une machine distante.

• U2R : Utilisateur à Root - Gain privilèges super-utilisateur local (root).

Même si cet ensemble de données est ancient et que plusieurs recherches aient signaler ses défaults[45], il est toujours considéré comme standard et utilisé par des études récentes [18], [48] dans divers domaines.

3.2.1.2 NSL-KDD

L'ensemble de données NSL-KDD a été proposé en 2009 [80] comme une nouvelle version révisée du KDDCup'99 ensemble de données d'origine. La Figure 3.1 illustre les données attaques de NSL-KDD.

Attack category	Description	Data instances - 10 % data			
		KDDCup 99		NSL-KDD	
		Train	Test	Train	Test
Normal	Normal connection records	97,278	60,593	67,343	9,710
DoS	Attacker aims at making network resources down	391,458	229,853	45,927	7,458
Probe	Obtaining detailed statistics of system and network configuration details	4,107	4,166	11,656	2,422
R2L	Illegal access from remote computer	1,126	16,189	995	2,887
U2R	Obtaining the root or superuser access on a particular computer	52	228	52	67
Total		**494,021**	**311,029**	**125,973**	**22,544**

Figure 3.1 Déscription du jeux de données NSL-KDD tirée de Vinayakumar et al,.(2019)

Alors que NSL-KDD a gardé les caractéristiques avantageuses et stimulantes de KDDCup'99, il adressa quelques inconvénients hérités de l'ensemble de données d'origine, comme la suppression

des enregistrements redondants, l'inclusion d'un nombre plus raisonnable de cas, et le maintien de la diversité des échantillons sélectionnés.

En particulier, la caractéristique importante de NSL-KDD est qu'il a été compilé afin de maximiser la difficulté de prédiction. Les données initial ont évalué à l'aide de plusieurs classifieurs référence et chaque instance a été annotées avec le nombre de ses prédictions groupe succès que pour finalement les grouper en cinq niveaux de difficultés [49]. Dans chaque groupe, le nombre d'enregistrements sélectionnés est inversement proportionnel aux pourcentages d'enregistrements de l'ensemble de données KDDCup'99 original.

3.2.1.3 CIC-IDS2017

Nous avons choisis un deuxième ensemble de données CIC-IDS2017 publié par l'Institut canadien pour la cybersécurité (CIC) [93], il contient des attaques bénignes à jour, qui ressemble à la vraie capture des paquets (PCAPs) du monde réel. Il comprend également les résultats de l'analyse trafic réseau à l'aide de CICFlowMeter avec des flux étiquetés en fonction de time stamp, source et destination IP, les ports source et destination, les protocoles et les attaques (CSV files) qui a été créé avec le système B-Profile qui décrit le comportement abstrait d'interactions humaines et génère un trafic de fond naturel bénin.

Ceci est l'un des plus récent jeu de données de détection d'intrusion marquée, qui couvre tous les onze critères nécessaires avec des attaques mises à jour courantes telles que DDoSS, BRUTE FORCE, XSS, SQL Injection, Infiltration, Port Scan et Botnet. Dans le détail, cet ensemble de données contient les enregistrements 2,830,743 conçus sur 8 fichiers et chaque enregistrement comprend 78 primitives différentes avec son étiquette,nous avons utilisé les fichiers CSV qui sont disponible en ligne pour aider les chercheurs. Nous avons utilisé CICIDS2017 pour répondre à l'une des questions de notre recherche car l'ensemble de données de référence de NSL-KDD n'a que 45 attributs pour déterminer s'il s'agit d'une attaque, ce qui rend la détection des attaques très difficile et la mise à niveau de la fonctionnalité du réseau de chaque paquet a plus d'attributs

à considérer. Nous avons analysé 85 attributs de paquets de l'ensemble de données CIC-IDS2017, ce qui est très favorable pour la prise de décision.

Tableau 3.1 Déscription des données CIC-IDS2017.

Taille(GB)	**Déscription**
Lundi 11 G	Benigne
Mardi 11 G	Attaques et Benigne
Mercredi 13 G	Attaques et Benigne
Jeudi 13 G	Attaques et Benigne
Vendredi 8.3 G	Attaques et Benigne

Tableau 3.2 Comparaison entre le jeu de données
générés et publics basés sur un cadre d'évaluation des IDS

	Traffic	**Normal**	**Meta**	**Format**	**Network**	**Splits**	**Hetero**	**Labeled**
[93]	vrai	vrai	vrai	Packet	vrai	faux	vrai	vrai
[82]	vrai	vrai	faux	Autre	vrai	faux	vrai	vrai
[80]	vrai	vrai	faux	Autre	vrai	vrai	vrai	vrai

3.2.2 Filtrage des données

En raison de l'hétérogénéité des plates-formes, les données brutes contiennent inévitablement des cas anormaux et redondants, ce qui peut avoir un effet négatif sur la précision de la classification. Pour résoudre ce problème, ces dossiers doivent être retirés de l'ensemble de données au début de nos expériences. Par exemple, la fonction «Flow Packets» dans l'ensemble de données CIC-IDS2017 comprend des valeurs anormales telles que «Infinity» et « NAN ». De plus, l'ensemble de données CIC-IDS2017 contient 8 entités non valides qui ont la même valeur pour tous les enregistrements.

3.2.3 Transformation des données

Les ensembles de données utilisées contiennent des valeurs symboliques, continues et binaires. Par exemple, la fonction « type de protocole » dans les ensembles de données KDDCup'99 et NSL-ECD comprend des valeurs symboliques telles que « tcp », « udp » et « ICMP ». Comme

beaucoup de classifieurs n'acceptent que des valeurs numériques, le processus de conversion est considéré comme vital et a un impact significatif sur la précision d'un système IDS. Dans cet cette expérience, nous remplaçons chaque valeur unique avec un nombre entier afin de gérer les fonctions symboliques. De plus, différentes échelles entre les caractéristiques peuvent dégrader performances de la classifications, par exemple, les caractéristiques qui prennent de grandes valeurs numériques, par exemple, pour l'ensemble de données CIC-IDS2017, «Flow Duration» peut dominer le modèle par rapport aux caractéristiques des valeurs numériques relativement petites comme «Total Fwd Packets». En conséquence, la normalisation est une transformation mise à l'échelle avec une méthode simple et rapide dite One Hot Encoder(OHE) [ ] est utilisé dans notre expérimentation.

3.2.4 Création de données équilibrées

Nous utilisons l'ensemble de données NSL-KDD, cet ensemble de données est une référence pour la détection d'intrusion basée sur l'apprentissage de la machine, cependant, il souffre de plusieurs inefficacités comme le déséquilibre des classes, où, par exemple, dans l'ensemble de données de formation NSL-KDD seulement 0,04% des échantillons appartiennent à le type d'attaque U2R rendant sévèrement sous-représentés, le cas est similaire pour les types d'attaque r2l et de la sonde alors que la majorité des dossiers d'attaque sont représentant le type d'attaque DDOS, ce fait rendu difficile pour les classificateurs de détecter ces types sous-représentés entraînant une mauvaise précision. Un autre problème est que cet ensemble de données est irréaliste, en réalité, la plupart du trafic dans un réseau est bénin et seul un faible pourcentage pourrait être malveillant, alors que dans la formation NSL-KDD mis par exemple ; l'échantillons d'attaque de 80% composent l'ensemble des données qui rend les modèles formés en utilisant cet ensemble de données inefficaces dans des situations de la vie réelle. Afin d'éviter le déséquilibre des échantillons représentant chaque type d'attaque dans les données d'entraînement de NSL-KDD, et d'éviter l'incapacité du modèle à apprendre de nouveaux types d'attaque en observant les types existants, nous présentons une approche qui utilise des codeurs automatiques et la reconstruction d'erreur pour détecter les anomalies.

3.3 Module de reduction de la dimension

Il existe quelques moyens de réduire les dimensions de grands ensembles de données pour assurer l'efficacité du calcul, comme la sélection en arrière, l'élimination des variables présentant une corrélation élevée, un nombre élevé de valeurs manquantes. L'autocodeur est une méthode relativement nouvelle de réduction de la dimensionnalité. Les autoencodeurs sont une branche de réseau neuronal qui tente de compresser les informations des variables d'entrée dans un espace dimensionnel réduit, puis de recréer le jeu de données d'entrée. Généralement, l'auto-codeur est entraîné sur le nombre d'itérations en utilisant une descente de gradient, ce qui minimise l'erreur quadratique moyenne. Le composant clé est la couche cachée. C'est ici que l'information de l'entrée a été compressées. En extrayant cette couche du modèle, chaque noeud peut être traité comme une variable de la même manière que chaque composant principal choisi est utilisé comme variable dans les modèles suivants.

3.3.1 Sélection des caractéristiques

Les encodeurs automatique en quelque sorte sont similaires aux techniques de réduction de la dimensionnalité comme l'analyse des composantes principales. Ils créent un espace dans lequel les parties essentielles des données sont conservés, tandis que des parties non essentielles (bruyante) sont enlevés. Cela peut sembler comme les méthodes de sélection de caractéristiques ou de compression de modèle, mais la plus grande différence entre un algorithmes de compression et usage autoencoder général est qu'en cas de autoencoders, la compression est obtenue par l'apprentissage sur un ensemble de données pour éviter plusieurs inefficacités comme le déséquilibre des classes dans les données NSL-KDD ce qui a une influence majeure par rapport à la réduction de temps d'apprentissage.

3.4 Méthodologie

Pour atteindre notre objectif, nous élaborons dans ce qui suit les étapes à suivre dans le but de réaliser un framework fédéré en considération de plusieurs facteurs tel que :

• La gestion efficace de la communication : Nous étudierons comment la taille du modèle communiqué entre les clients et les impacts du serveur le compromis entre la coût de la communication et l'exactitude. Nous allons effectuer des expériences approfondies comparer la corrélation entre le coût de la communication et la précision. Atteindre cela, nous allons faire varier la taille des modèles mis à jour renvoyés par le client au serveur.

• Étudier comment la distribution de données asymétriques des clients affecte la précision de détection : Chaque client aura un ensemble unique de données variant dans une certaine degré en ce qui concerne les données d'apprentissage, la taille et l'étiquette en fonction du comportement du client pour déterminer comment ces paramètres affectent l'exactitude des modèle généré.

• Examiner l'agrégation des modèles locaux sur le serveur.

• Enfin, pour mettre en oeuvre la détection d'intrusion, nous adapterons les cadres mentionnés ci-dessus en modifiant le code source pour mieux répondre aux exigences de l'IDS, qui comprend les algorithmes, paramètres, ensembles de données, la structure du modèle, ... etc.

3.4.1 Architecture du modèle centralisé

Il s'agit d'un encodeur automatique qui se compose d'une couche d'entrée de 122 neurones du fait que le nombre de caractéristiques pour chaque échantillon est de 122, suivie par une couche Dropout et une couche cachée de 8 unités neuronales afin que la représentation cachée d'autoencodeur ait un taux de compression de 122/8, ce qui le force à apprendre des motifs intéressants et des relations entre les caractéristiques. Enfin il ya une couche de sortie de 122 unités, et l'activation de la couche cachée et la couche de sortie est la fonction «ReLU».

Le modèle est formé pour 10 époques à l'aide d'un optimiseur d'Adam [89] avec une taille de lot de 100, de plus nous avons distribué 20% des échantillons d'apprentissage pour valider le modèle.

Les contraintes de régularisation appliquées sur l'autoencodeur l'empêchent de simplement copier l'entrée à la sortie et de surajuster les données, en outre le décrochage présenté sur les

entrées fait de l'autoencoder un cas particulier d'un autoencodeur de dénoisation, ce genre des autoencodeurs est formé pour reconstruire l'entrée d'une version corrompue déformée de lui-même, forçant l'autoencoder à apprendre encore plus de propriétés des données.

3.4.2 Distribution des données

Nous avons utilisé divers modes de répartition des données afin d'étudier l'influence de l'hétérogénéité de la distribution sur l'apprentissage global. Nous intéressons à l'hétérogénéité des données non-IID, et au déséquilibre de différentes répartitions en termes de quantité. Pour réaliser une répartition équilibrée, les données sont réparties en quantités équitables entre les différents clients, une deuxième répartition est déséquilibrée et enfin, une distribution mixte qui combine les deux distributions précedentes a été expérimenté dans le but d'évaluer les avantages de la manière de répartitions. Pour se faire, la quantité de données affectées à chaque client pour une répartition équilibrée (au sens uniforme) en quantité consiste à affecter le même nombre de données à chaque client, contrairement à une répartition déséquilibrée (unbalanced) pour laquelle cette quantité est distribuée aléatoirement.

3.4.3 La référence centralisée

Pour avoir une valeur de référence des résultats qui sont utilisés avec le modèle où les données sont distribuées nous avons entrainé un modèle classique basé sur les SVM avec des données centralisées de CIC-IDS2017 et un modèle neuronal convolutif avec les données NSL-KDD. Nous nous référerons donc les résultats de cette approche afin de comparer les performances observées avec l'approche d'apprentissage fédéré.

3.4.4 Outils et matière

Conformément à la construction de l'environnement et mettre en oeuvre les tests, une gamme de différents outils doit être utilisés et combinés comme l'installation et la configuration de

l'environnement, aussi bien pour le développement des algorithmes. Après avoir examiné les outils de mise en oeuvre, la liste suivante a été choisis :

• Tensorflow : Calcul numérique utilisant des graphes de flux de données pour la construction de réseaux de neurones convolutifs [83]. La partie expérimentale présentée plus bas dans le chapitre suivant a été intégralement réalisée en langage Python avec TensorFlow [83].

• Keras : API Keras intègre très facile avec les flux de travail de tensorflow. Il est très utile pour l'apprentissage profond.

• Scikit-learn : La fameuse librairie est un module d'apprentissage automatique et un outil efficace pour l'exploration et l'analyse des données [88].

• Python : Comme un langage de script pour automatiser des expériences et l'extraction de données.

3.4.5 Implementation des algorithmes

Comme déjà mentionnés précédemment, notre étude porte sur trois algorithmes pour la détection d'anomalie basés sur deux approches différentes soit l'approche centralisée et l'approche décentralisé fédéré. Il existe de grandes variétés d'algorithmes d'apprentissage automatique ont été largement utilisés pour la détection des anomalies, parmi ceux nous avons choisi ces modèles pour leur faible coût tel que les réseau de neurones artificiels et les SVM (Support Vector Machine). Dans cette section, nous allons présenter les algorithmes proposés avec l'implémentation. En fait, il s'agit d'une modélisation simple de la structure des modèles proposés avec les différentes metriques associées au niveau des données.

3.4.5.1 Algorithme distribué fédéré

Cette approche est basée sur le travail de McMahan et al., (2017) qui font parti de l'équipe de chercheurs de Google AI. Pour que l'apprentissage fédéré soit possible, nous avons dû surmonter de nombreux problèmes techniques et algorithmiques ; un algorithme d'optimisation

tel que la descente de gradient stochastique (SGD) illustré dans la Figure3.2, s'exécute sur un grand jeu de données partitionné de manière homogène sur des serveurs. De tels algorithmes hautement itératifs nécessitent des connexions à haut débit avec une latence faible et des données d'apprentissage de grande dimension. Cependant, dans le cadre de l'apprentissage fédéré, les données sont réparties de manière très inégale sur des millions d'appareils. En outre, ces périphériques disposent de connexions à débit plus lent et à latence nettement supérieure et ne sont disponibles que par intermittence d'apprentissage.

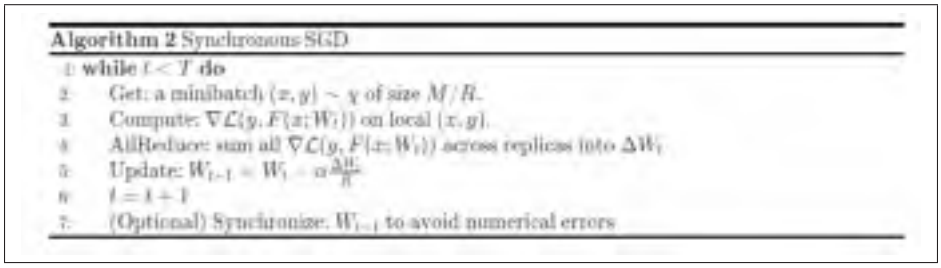


Figure 3.2 La déscente du gardient stochastique synchrone tirée de (Arnold, 2016)

Ces limitations de bande passante et de latence motivent notre algorithme de calcul de la moyenne fédérée qui peut former des réseaux profonds en utilisant 10 à 100 fois moins de communication par rapport à une version naïvement fédérée de SGD. Notre idée principale est d'utiliser des processeurs puissants modernes pour calculer des mises à jour de meilleure qualité, comme il nous faut moins d'itérations de mises à jour de haute qualité pour produire un bon modèle, l'apprentissage peut utiliser beaucoup moins de communication.

Dans ce cadre, ils ont conçu Federated Averaging afin que le serveur de coordination n'ait besoin que de la mise à jour moyenne ce qui permet d'utiliser de l'aggregation sécurisée. Cependant, le protocole est général et peut également être appliqué à d'autres problèmes. L'algorithme de moyenne fédérée abrégé FedAVG introduit dans le document d'origine de Google est présenté dans la Figure 3.3.

Un cycle d'apprentissage typique d'un algorithme fédéré comprend la séquence suivante :

• Un sous-ensemble aléatoire membres de la fédération (appelés clients) est sélectionné pour recevoir le modèle global de manière synchrone à partir du serveur.

• Chaque client sélectionné calcule un modèle mis à jour à l'aide de ses données locales.

Les mises à jour du modèle sont envoyées par les clients sélectionnés au serveur.

• Le serveur agrège ces modèles (généralement par une moyenne) pour construire un modèle global amélioré.

Contrairement à Google qui avait initialement appliqué l'algorithme FedAVG sur des données collectées via des millions de combinés dans son écosystème Android, nous allons appliquer cet algorithme à des paquets réseau pour la détections d'intrusion. L'ensemble des paramètres du modèle fédéré sont dictés par la partie centralisée du système d'apprentissage global. L'algorithme ci dessous montre comment ces paramètres sont utilisés au sein d'une structure algorithmique globale telle que nous l'avons utilisée dans nos expériences.

Paramètres du modèle :

T : nombre de tours de communication (t = 1 ; 2 ; : : : ; T).

K : nombre total de clients.

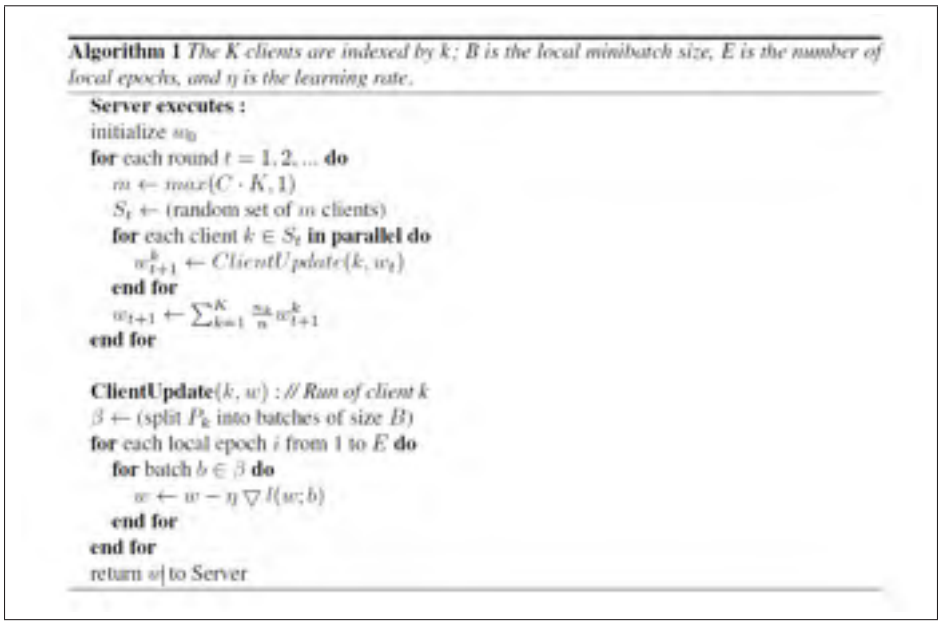
C : fraction de clients à sélectionner lors de chaque tour t.

B : taille des batchs locaux (B =1 correspond à un batch intégral).

E : nombre d'itérations effectuées par les modèles locaux.

η : taux d'apprentissage local (learning rate).

L'algorithme FedAvg fonctionne bien et stable à travers de nombreuses séries de tests et il surpasse beaucoup d'autres approches décentralisées de la littérature, y compris d'autres approches centralisées que nous avons tenté dans le cadre de notre étude. Notre modèle effectuera



Algorithm 1 *The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.*

Server executes :
initialize w_0
for each round $t = 1, 2, \ldots$ **do**
  $m \leftarrow max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
  end for
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

ClientUpdate(k, w) : *// Run of client k*
$\beta \leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \beta$ **do**
    $w \leftarrow w - \eta \bigtriangledown l(w; b)$
  end for
end for
return w to Server

Figure 3.3 Algorithme de moyenne fédéré tirée de Mcmahan et al., (2017)

une classification binaire des données à deux classes indiquant si le trafic est normal ou une attaque.

3.4.5.2 Réseaux de neurones convolutifs

Réseau de neurones convolutifs est très similaire à celle du réseau de neurones qui reçoit une entrée et la transformer en plusieurs couches cachées. Chaque couche cachée est un ensemble combiné de neurones et tous les neurones sont entièrement connectés à tous les neurones de la couche précédente, par contre, dans un réseau de neurones à une seule couche ils sont independents. Nous appelons la couche de sortie qui est la dernière couche, une couche fortement connectée [6].

Avec des couches plus complexes et abstraites qu'un modèle typique, les RNA sont capables d'apprendre les caractéristiques de haut niveau ce qui en fait le plus populaire pour les tâches de vision par ordinateur, par exemple, la classification d'images ce qui permet l'inférence de haute précision dans les tâches, parmi ceux nous avons choisis le réseau de neurones convolutif

qui est conçu pour la classification d'images, comme le premier CNN qui a remporté le Défi IMAGEnet en 2012 [59]. Compte tenu d'une série d'attaques à partir d'un monde réel avec l'utilisation de CNN, le système d'intelligence artificielle apprend à extraire automatiquement les caractéristiques de ces entrées pour accomplir une tâche spécifique, par l'application de différents filtres convolutifs.

Nous avons 121 entrées différentes traversent trois couche cachée de notre modèle décentralisé proposé. Toutes les couches sont entièrement connectées les unes aux autres et deux sorties obtenues soit une attaque ou bénigne de l'ensemble NSL-KDD.

Dans notre travail, nous avons mis l'accent sur deux modèles profonds, le modèle shallow qui signifie un modèle simple à une seule couche comme celui de l'encodeur automatique versus notre modèle fédéré plus profonds à trois couches pour comparer leur performance.

3.4.5.3 Algorithme OCSVM

Dans notre nous allons intérésser à l'algorithme OCSVM ou encore les SVM non supervisés tant qu'une approche centralisée pour la détection d'IDS basé sur le travail de [90].

Le but d'un classificateur OCSVM est identique à celui d'un classificateur binaire supervisé car OCSVM est un P-classificateur, à savoir le modèle de classification est construit avec des échantillons normaux seulement. Contrairement au classificateur supervisé [74], les données d'apprentissage de ce classificateur contiennent des échantillons étiquetés uniquement de la classe d'intérêt. Dans notre cas d'un classificateur binaire l'autre classe maligne ou la classe négative doit être représenté par l'ensemble d'apprentissage. La collecte d'un ensemble d'apprentissage automatique représentative fixée pour la classe négative peut être très coûteuse en terme du temps en raison du fait que la classe négative est l'agrégation de toutes les autres classes sans la classe normale.

3.5 Discussion

Les chercheurs ont contribué à une meilleure compréhension des problèmes à prendre en compte pour faire passer l'apprentissage fédéré du concept au déploiement. Bien que des travaux aient été réalisés sur l'efficacité et la précision de l'apprentissage fédéré, à notre avis les défis les plus importants sont ceux liés à la sécurité.

L'apprentissage fédéré reste toutefois confronté notamment l'incapacité à inspecter les exemples d'apprentissage, les problèmes de bande passante, la nécessité d'une connexion WiFi et la nécessité de déduire naturellement de l'étiquetage des interactions de l'utilisateur.

L'appretissge d'un modèle local supervisé nécessite des données étiquetées qui ne sont pas disponibles ou difficiles à produire dans la plupart des cas. Un bon moyen de relever ce défi consiste à définir le problème de l'apprentissage fédéré et à concevoir un pipeline de données tel que les libellés soient capturés de manière implicite, par exemple, les interactions des utilisateurs, les commentaires sur les réponses du modèle en fonction de certaines actions entreprises ou d'événements déclenchés, etc.

Dans le monde réel, le temps de convergence du modèle est plus élevé dans la configuration fédérée que dans l'approche d'apprentissage centralisée traditionnelle. Il peut y avoir des problèmes de fiabilité si tous les clients ne participent pas au processus d'apprentissage fédéré en raison de problèmes de connectivité, de modèles d'utilisation des applications et de temps d'apprentissage des modèles différents, de mises à jour irrégulières ou manquantes, etc. L'apprentissage fédéré ne doit être pris en compte que lorsque la taille des données et leur coût d'agrégation à partir de sources distribuées est très élevé. Des versions de modèle incohérentes entre les clients n'affectent pas trop l'expérience pour un laps de temps significative et le modèle central peut converger avec une participation minimale d'un client.

Enfin, l'agrégation centralisée des données et la création de silos par les grandes entreprises pour un avantage concurrentiel constitueraient un défi majeur pour l'adoption de l'apprentissage fédéré. Des politiques efficaces de protection des données, ainsi que des incitations appropriées

et des modèles commerciaux axés sur la décentralisation des données peuvent résoudre ces problèmes et développer l'écosystème fédéré.

3.6 Conclusion

Dans cette approche, nous avons tenté de surmonter les problèmes qui existent dans les ensembles de données KDD99 et NSL-KDD, à savoir la question du déséquilibre des classes et les données irréalistes, ce qui rend notre recherche plus utile dans les applications du monde réel et plus viable pour une utilisation dans les réseaux réels. Pour traiter un réseau de grande dimension avec des données hétérogènes déséquilibrée, nous proposons un IDS basé sur une approche décentralisé fédéré. Ensuite, pour construire le modèle nous allons utilisé FedAvg avec une distributions aléatoire locales sur les données NSL-KDD pour optimiser notre solution. Dans le but d'évaluer notre approche par rapport à une approche centralisé nous proposons l'algorithme autoencodeur avec les meme données de NSL-KDD ainsi que l'algorithme OCSVM d'apprentissage machine classique sur les données de CIC-IDS2017.

CHAPITRE 4

EXPÉRIMENTATION ET RÉSULTATS

4.1 Intoduction

Peu d'aspects devraient avoir un impact sur la conception d'un modèle distribué fédéré. Le principe ici est d'envisager de manière appropriée les réseaux de neurones convolutifs car ils font un calcul parallèle très coûteux en ce qui concerne le nombre de paramètres qu'ils contiennent. Il s'agit d'une propriété favorable du réseau, car nous souhaitons limiter le temps passé en communication tant qu'une surcharge par opposition au calcul. En plus d'être particulièrement performantes avec les données corrélées, les couches convolutifs y parviennent parfaitement puisqu'elles multiplient les caractéristiques sur toutes les entrées. Un autre point à considérer est l'utilisation d'optimiseurs basés sur la quantité de mouvement avec des résidus et des poids quantifiés. Nous allons explorer une vaste évaluation empirique de l'approche proposée selon les différents scénarios qui ont été effectués pour avoir une meilleire optimisation de l'algorithme FedAVG ce qui permettent de bien estimer la performance de notre FIDS.

Dans le but de fournir une évaluation de la performance des algorithmes proposés, nous avons considéré aussi la consommation de ressources en terme de CPU, RAM, etc.

4.2 Étude comparative

Dans cette étude, la performance du modèle profond a été comparée la détection d'attaques distribuée et a été évaluée par rapport au système de détection centralisé. Nous allons examiner les différentes mesures et techniques d'apprentissage automatique et le temps nécessaire pour transférer des paramètres de modèle avec une consommation efficace des ressources. Cela comprend plusieurs expériences pour une analyse approfondie de l'envoi des mises à jour du modèle global au serveur, la surconsommation des dispositifs en terme de CPU/RAM , et la fréquence des modèles locales. Une vaste évaluation empirique de notre approche a été proposée, plus concrètement, nous présentons l'algorithme de calcul de la moyenne fédérée, qui combine

la descente de gradient stochastique local (SGD) sur chaque client avec un serveur qui effectue la moyenne globale.

4.2.1 Métriques d'évaluation

Pour évaluer la performance des deux approches proposées, nous avons utilisé un ROC curve propsé par Hanley et McNeil,(1982) qui est une courbe d'un modèle représentant ses vrais positifs par rapport à ses faux positif. L'aire sous cette courbe représente la performance du modèle.

Concrètement, la détection efficace des fausses alarmes ne signifie pas une bonne estimation de la performance du modèle. Pour cette raison, nous avons aussi mesuré le nombre des attaques non détectées en calculant le taux d'exactitude présenté dans la formule (4.1).

$$Accuracy = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}} \tag{4.1}$$

4.3 Test d'évaluation

Cette section couvre l'ensemble des tests de détection faites avec les trois algorithmes que nous allons investiguer dans nos expériences avec divers scénarios et plusieurs configurations. Nous avons réalisé nos expérimentations en utilisant deux modèle centralisé avec des données normal qui sont formé avec NSL-KDD et des données réelles de CIC-IDS2017.

Le premiers test effectue la détection d'anomalies par le calcul de l'erreur de reconstruction d'échantillons, étant donné que NSL-KDD ont une distribution asymétrique des données, cette intuition nous permettrons de détecter des attaques par fixer un seuil pour l'erreur de reconstruction, si un échantillon de données contient une erreur de reconstruction supérieure au seuil, l'échantillon est classé comme une attaque, sinon il est considéré comme un trafic normal.

Pour le deuxième modèle nous avons utilisé des données de test de CIC-IDS2017 pour évaluer sa performance avec quatre scénarios selon les jours de semaine.

4.3.1 Paramètres d'évaluation

L'optimisation des poids pour effectuer une tâche donnée est essentielle à tout modèle d'apprentissage en profondeur. Cet ensemble de classes fournit des options pour la sélection de l'algorithme d'optimisation approprié. Étant donné les paramètres θ le taux d'apprentissage, α les gradients calculé sur les données du minibatch x, SGD met à jour les paramètres par la formule suivante :

$$\theta' = \theta - \alpha \nabla J(\theta; x)) \tag{4.2}$$

Nous mettons en oeuvre SGD avec Momentum qui suit l'historique des mises à jour de gradient pour aider le système à accélerer plus rapidement au niveau des points d'équilibre. Compte tenu des paramètres supplémentaires : moment γ, perte de poids λ et la vitesse actuelle υ, nous utilisons les équations de mise à jour présentées dans la Figure 4.1.

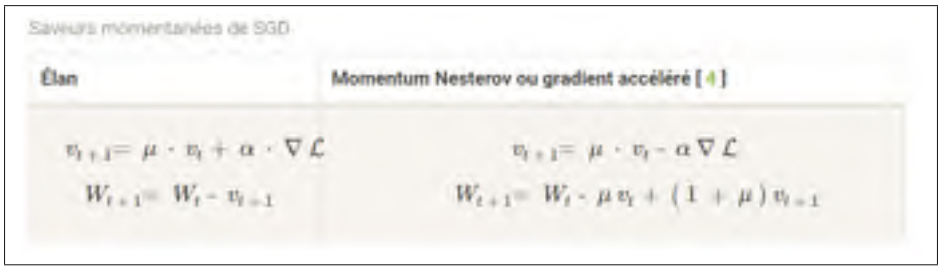


Figure 4.1 SGD avec Momentum tirée de Arnold, 2016

Dans ces formules , μ est le paramètre momentum qui signifie combien de mises à jour précédentes nous voulons inclure dans la mise à jour actuelle. La pente accélérée de Nesterov [95] donne un nouvel moment à Momentum pour tenter d'envisager ce qui est à venir.

4.3.2 Mise à l'échelle adaptative du taux d'apprentissage

Trouver de bons taux d'apprentissage peut être un processus coûteux et une compétence souvent considérée comme une boite noire. Les métriques que nous avons utilisé sont présentés dans la Figure 4.2, elles tentent de résoudre ce problème en définissant automatiquement le taux d'apprentissage ou par paramétrage.

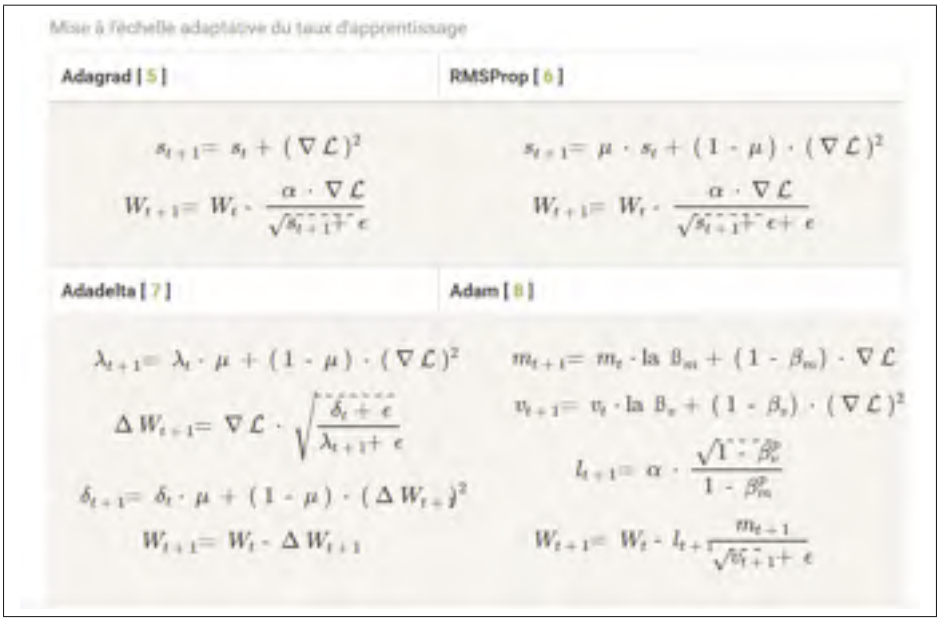


Figure 4.2 Paramètres d'optimisation tirée de (Arnold, 2016)

Dans les formules au dessus ϵ est une constante pour assurer la stabilité numérique, et μ est la constante de désintégration de l'algorithme, à quelle vitesse nous diminuons le taux d'apprentissage lorsque nous convergeons, où p est l'époque actuelle, c'est-à-dire un pas plus le nombre de passages dans l'ensemble de données.

Nous intéressons à deux optimiseurs pour calculer les gradients, avec une régularisation L2 qui est la pénalité pour la complexité d'un modèle et elle aide à éviter le surapprentissage. Les mises à jour de SGD a été appliquées sur le modèle locale avec un réglage d'hyperparamètres accéléré inclut la prise en charge de Momentum, de la décélération du taux d'apprentissage et le moment de Nesterov.

La valeur de Momentum est un paramètre qui accélère SGD dans la direction correspondante et atténue les oscillations. Pour le Nesterov est un booléen qui prend la valeur vrai que ce soit pour appliquer le moment Nesterov.

Le deuxième paramètre ajusté est appellé Adadelta [86] qui est une extension plus robuste d'Adagrad [96] qui adapte les taux d'apprentissage en fonction de mises à jour de gradients, au lieu de cumuler tous les gradients passés. Pour cette raison, nous avons appliqué ce paramètre au niveau du serveur pour contrôler la mise à jour des paramètres du modèle global. Adadelta continue à apprendre même lorsque de nombreuses mises à jour ont été effectuées par les nouveaux poids du modèle global.

Par rapport à Adagrad, dans la version d'origine d'Adadelta nous n'avons pas à définir de taux d'apprentissage initial. Dans notre implémentation, nous avons initialisé le taux d'apprentissage et le facteur de décroissance comme dans la plupart des autres optimiseurs de Keras. Nous avons fixé le taux d'apprentissage à 0.01 avec un facteur de décroissance appelé rho ou encore un decay d'Adadelta de 0.95 qui correspond à la fraction de gradient à conserver à chaque pas de temps.

4.3.3 Minimisation de la fonction de perte

La descente de gradient stochastique (SGD) consiste à calculer la direction du gradient de la fonction de perte que nous essayons de minimiser par rapport aux paramètres actuels du modèle. Une fois que nous connaissons la direction du gradient, c'est-à-dire la direction de la plus grande augmentation, nous ferons un pas dans la direction opposée puisque nous essayons de minimiser l'erreur finale. plus formellement, nous pouvons représenter notre jeu de données comme une distribution à partir de laquelle nous ferons des échantillons des tuples N d'entrées et d'étiquettes présentées par la formule 4.1.

$$W_{opt} = argminE_{(x,y)}[\![L(Y, F(X; W))]\!] \tag{4.3}$$

Ensuite, étant donné une fonction de perte L notre choix comprend la fonction de l'erreur moyenne quadratique, nous voulons trouver l'ensemble optimal de poids Wop de notre modèle neuronal profond. Dans ce cas, SGD mettra à jour itérativement les poids Wt en timestep t avec :

$$W_{t+1} = W_t - \alpha \cdot \nabla W_t L\left(Y, F(X; W)\right) \tag{4.4}$$

Dans cette formule, le taux d'apprentissage peut être interprété comme l'ampleur de l'étape que nous prenons dans la direction du gradient négatif. Comme nous le verrons, il existe des algorithmes d'optimisation liée à l'apprentissage profond qui tentent de définir de manière adaptative le taux d'apprentissage, mais en général, il doit être choisi par l'expérimentateur humain.

Il est important de noter que, le gradient est évalué sur un ensemble d'échantillons de minibatch qui varie d'un scénario à un autre et ceux sont effectué en faisant la moyenne de la perte du gradient pour chaque échantillon du minibatch ce qui aide à deux aspects :

• Il peut être correctement calculé par un calcul vectoriel.

• Il nous permet d'obtenir une meilleure approximation du gradient réel de L(y,F(x ; W)) au-dessus de x, et nous fait donc converger plus vite.

En outre, le choix d'une fonction de perte appropriée peut être utiles pour guider le processus d'estimation de taux d'erreur de modèle d'apprentissage et sur les données de validation, nous allons entamer ces expériences dans la section suivante.

4.4 Scénarios de test

Dans cette expérience, nous utilisons un environnement contrôlé qui convient à notre expérimentation, mais qui aborde toujours les questions clés de l'approche fédéré soient la disponibilité des clients et la nature déséquilibrées des données de NSL-KDD. Nous partons du principe d'un système de mise à jour synchrone qui procède des tours de communication où il y a un

ensemble fixe des clients K, chacun avec un ensemble de données local fixe en mettant l'accent sur les propriétés non-IID car la répartition déséquilibrée et non IID de NSL-KDD est beaucoup plus représentatif du type de distribution de données que nous attendons pour les applications du monde réel. Au début de chaque tour, une fraction aléatoire C égale à 0.1 de clients est sélectionnée, le serveur envoie l'état de l'algorithme global courant à chacun de ces clients (par exemple, les paramètres du modèle en cours).

Dans ce cadre, notre objectif est de diminuer le nombre de tours de communication nécessaires pour construire le modèle. Il y a deux façons principales que nous pouvons rajouter :

• Une augmentation du parallélisme : où plus de clients travaillant localement entre chaque cycle de communication.

• Une augmentation de calcul sur chaque client : comme un calcul de gradient, chaque client effectue localement un calcul plus complexe entre chaque cycle de communication. Nous examinons ces deux approches, mais les accélérations que nous avons obtenu sont surtout dus à l'ajout de plus de calculs sur chaque client, une fois un niveau de parallélisme minimal sur les clients est assuré.

4.4.1 Scénario 1

Pour prouver l'efficacité de l'optimisation fédérée, nous avons également besoin de préciser comment les données sont réparties sur les clients. Nous étudions deux façons de partitionner les données NSL-KDD sur les clients selon [12], les données IID, où les données sont mixé aléatoirement, puis ils sont divisé en 100 clients chacun recevant 600 exemples, et Les non-IID sont diviseé en 200 sous ensemble de taille 300, et affecter chacun des 100 clients deux fragments. Ceci est une partition non-IID pathologique des données. Ainsi, cela nous permet d'explorer la mesure dans laquelle notre algorithmes persistent sur des données hautement non-IID, où l'une de ces partitions sont équilibrés, et l'autre est déséquilibré. Nous avons effectué des expériences supplémentaires sur les versions asymétriques de ces ensembles de données, et les a en fait plus facile avec FedAvg.

Nous partons nos expériences avec une fraction du client C, qui contrôle le parallélisme multi-client. Nous rapportons le nombre de tours de communication nécessaires pour obtenir une précision cible de l'ensemble de test. Pour calculer cela, nous construirons une courbe d'apprentissage pour chaque combinaison de paramètres, avec l'optimisation de puis nous amélioration monotone de chaque courbe en prenant la meilleure valeur de la précision des ensembles de test obtenues au cours de tous les tours précédents. Avec l'augmentation de B il n'y a qu'un petit avantage à augmenter la fraction du client, par contre l'utilisation d'une taille plus petite de minibatch B montre une amélioration significative dans l'utilisation de C 0,1, en particulier dans le cas de non-IID.

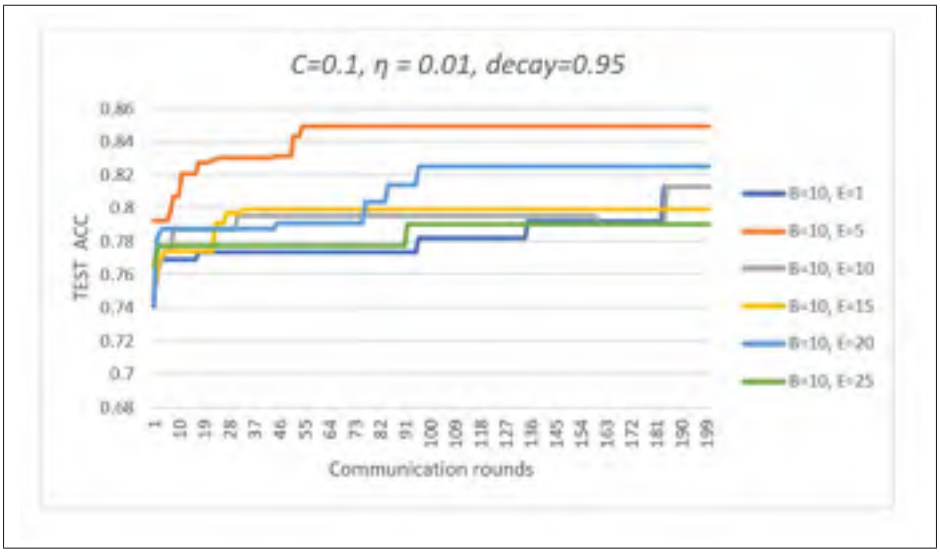


Figure 4.3 Acc du modèle fédéré NSL-KDD (scénario 1)

Tableau 4.1 Comparaison entre les strategies de la moyenne globale pour atteindre une précision cible du scénario 1

AVG strategy	50 rounds	200 rounds
ηglobal= 1.0	75%	78%
Adadelta ηglobal= 0.01	82%	85%

Sur la base de ces résultats, pour la plupart du reste de nos expériences, nous fixons C = 0,1, ce qui représente un bon équilibre entre le taux de calcul et de la convergence. En comparant le

nombre de tours pour plusieurs configurations de nombre d'époques avec B = 10 dans la Figure 4.3 qui montre une accélération remarquable de précision sur l'ensemble de validation.

Quand B = ∞ (C'est à-dire la taille du lot est égal à la taille du jeu de données locale) et E = 1, une seule mise à jour de gradient est performé sur les données de chaque utilisateur. Il est strictement équivalent à faire un seul calcul de gradient sur un lot comprenant tous les points de données utilisateur. Ce cas spécifique est appelée FedSGD, par exemple la descente du gradient stochastique avec chaque lot étant les données de la fédération d'utilisateurs sélectionnés à un tour donné. FedAvg (Moyenne fédéré) est le cas générique lorsque plus d'une mise à jour est effectuée localement pour chaque utilisateur. L'étape globale de la moyenne peut être écrit comme suit, en utilisant un taux de mise à jour globale $\eta global$:

$$W_t \leftarrow W_{t-1} - \eta global \sum_{k \epsilon S_t} \frac{n_k}{n} (w_{t-1} - w_{t,k}) \tag{4.5}$$

Le réglage de la vitesse de mise à jour globale global mis à 1 est équivalent à un cas de calcul de moyenne pondérée sans moyenne mobile. L'équation 4.5 met en évidence le parallélisme entre la moyenne globale et la mise à jour du gradient.

Ce parallélisme motive l'utilisation de mises à jour de coordonnées adaptatives Gt sous la formule 4.6 qui se sont avérés efficaces pour l'optimisation des problèmes non convexes tel que les réseaux de neurones profonds [104].

$$Gt = \sum_{k \epsilon S_t} \frac{n_k}{n} (w_{t-1} - w_{t,k}) \tag{4.6}$$

La moyenne sur le moment(Moment-based averaging) permet d'affiner le modèle en tenant compte les mises à jour des tours précédents qui ont été calculé sur différents sous-ensembles d'utilisateurs. Nous supposons que l'on a exponentiellement effectué le même type de régularisation du decay (decay rate) qui se produit dans la descente du gradient avec un mini-batch, où

Adadelta a fait ses preuves su un large éventail de tâches avec diverses architectures basées sur les réseaux neuronaux [105][91].

4.4.2 Scénario 2

Dans cette section, nous fixons C à 0,1 en rajoutant plus de calcul par client sur chaque tour, ce qui diminue soit B, soit l'augmentation de E, ou les deux. La Figure 4.4 montre que l'ajout de mises à jour de SGD locales et globale avec decay par tour peut produire une diminution spectaculaire des coûts de communication.

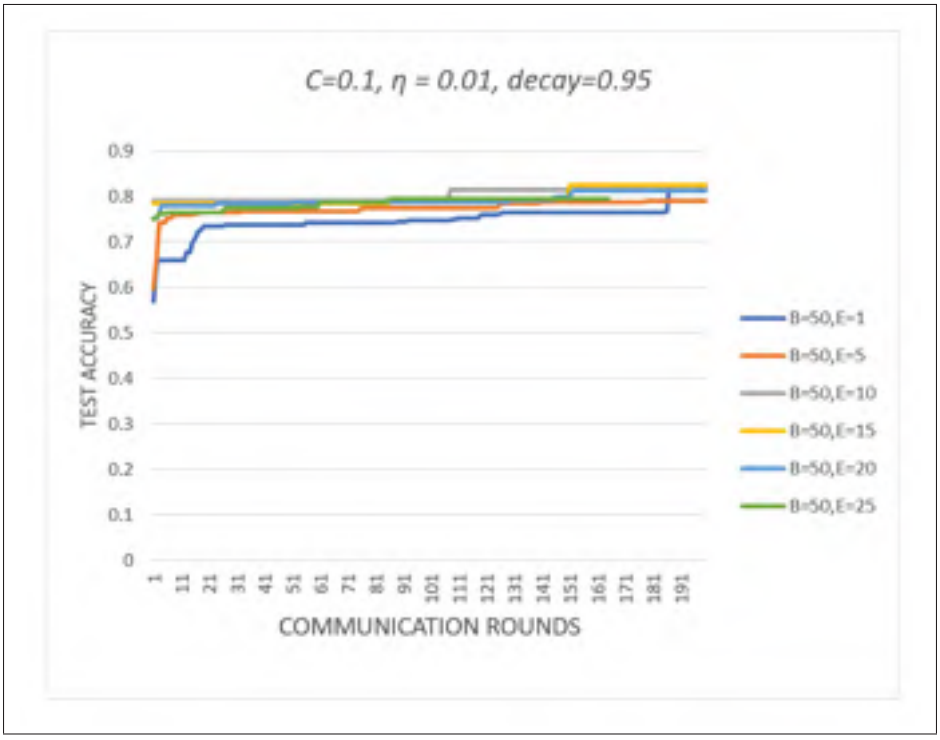


Figure 4.4 Acc du modèle fédéré NSL-KDD (scénario 2)

Tableau 4.2 Comparaison entre les strategies de la moyenne globale pour atteindre une précision cible du scénario 2

AVG strategy	50 rounds	200 rounds
ηglobal= 1.0	77%	80%
Adadelta ηglobal= 0.01	80%	82%

Le nombre prévu de mises à jour locales des clients par tour est μ = (E [ηk] / B)E = ηE / (KB), où l'attente est supérieure à la fraction d'un client k aléatoire. Nous voyons que l'ajout de mises à jour SGD locales par tour peut entraîner une réduction spectaculaire des coûts de communication, et le tableau 4.1 quantifie ces accélérations par l'augmentation de μ en variant à la fois E et B est efficace.

Tableau 4.3 Nombre des tours de communication pour atteindre une précision cible avec FedAVG

CNN	E	B	μ	NON-IID
FedAVG scénario 1	1	10	6	181
	5	10	30	15
	10	10	60	160
	15	10	180	30
	20	10	720	73
	25	10	3600	90
FedAVG scénario 2	1	50	1.2	191
	5	50	6	30
	10	50	12	19
	15	50	36	10
	20	50	144	8
	25	50	720	5

Tableau 4.4 Optimisation du nombre des tours de communication avec FedAVG en comparaison avec la méthode FedSGD pour atteindre une précision cible

ACC	80%	82%	85%
FedSGD	250 (4.8X)	370 (4.7X)	N/A (—)
FedAVG	20	50	65

Nous avons calculé la colonne μ qui présente le nombre prévu de mises à jour par tour ce qui influ sur le nombre de tour de communication que nous voulons le diminuer pour une précision cible. Tant que B=50 est assez grand pour profiter pleinement du parallélisme disponible du client, il n'y a aucun coût dans le temps de calcul pour l'abaisser, et donc dans la pratique cela devrait être parmi les paramètres ajustés.

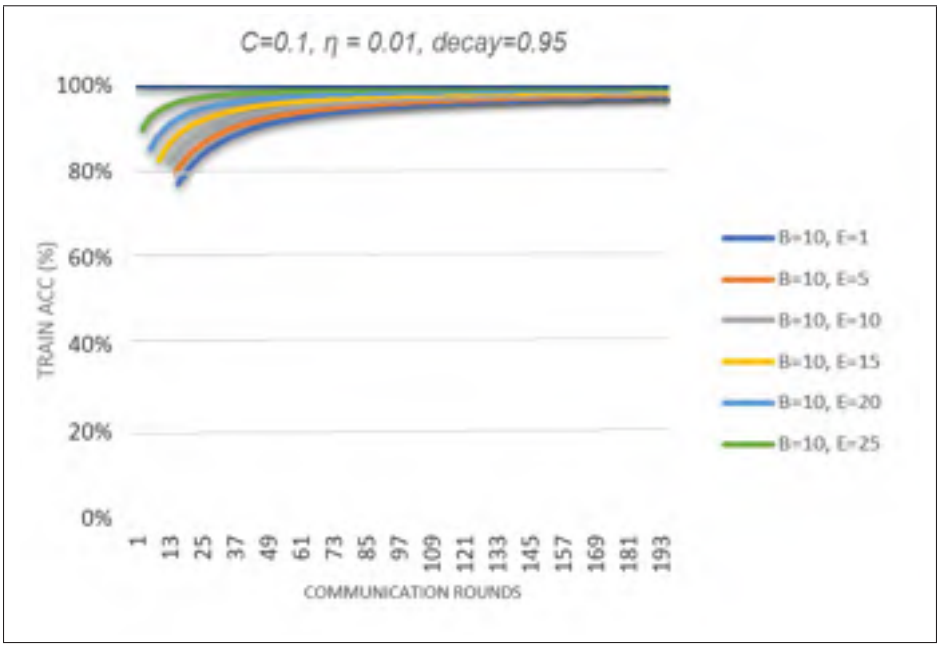


Figure 4.5 Acc du modèle locale NSL-KDD (scénario 1)

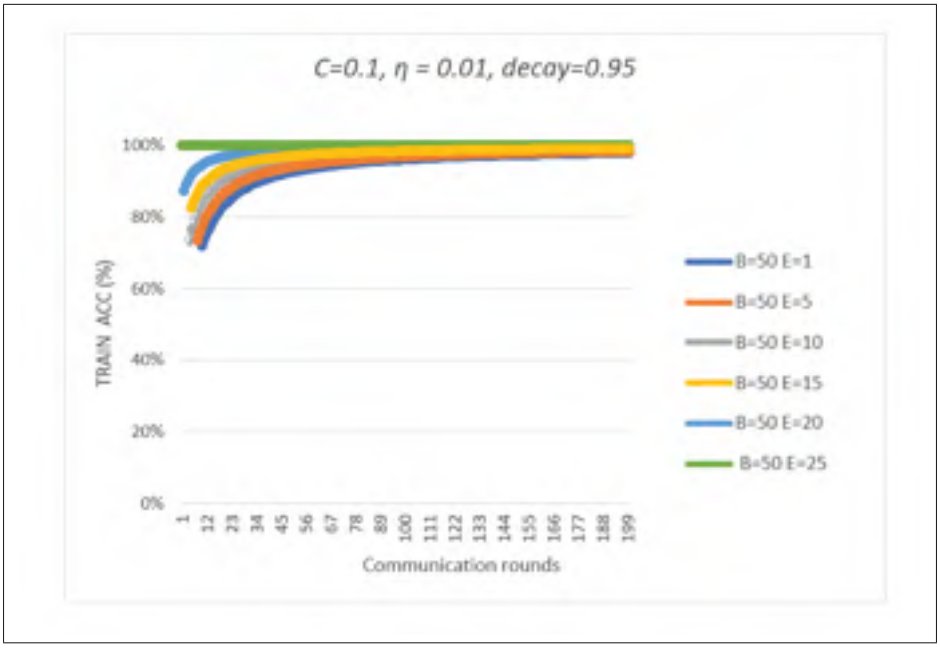


Figure 4.6 Acc du modèle locale NSL-KDD (scénario 2)

Nous rapportons le nombre de communications nécessaire pour atteindre une performance cible de l'ensemble des tests. Pour se faire, nous construisons une courbe sur l'ensemble

d'apprentissage pour chaque combinaison des paramètres, l'optimisation du modèle locale telle que décrite dans les figures au dessous, en prenant la meilleure valeur de la précision des ensembles de test obtenue sur tous les tours précédents.

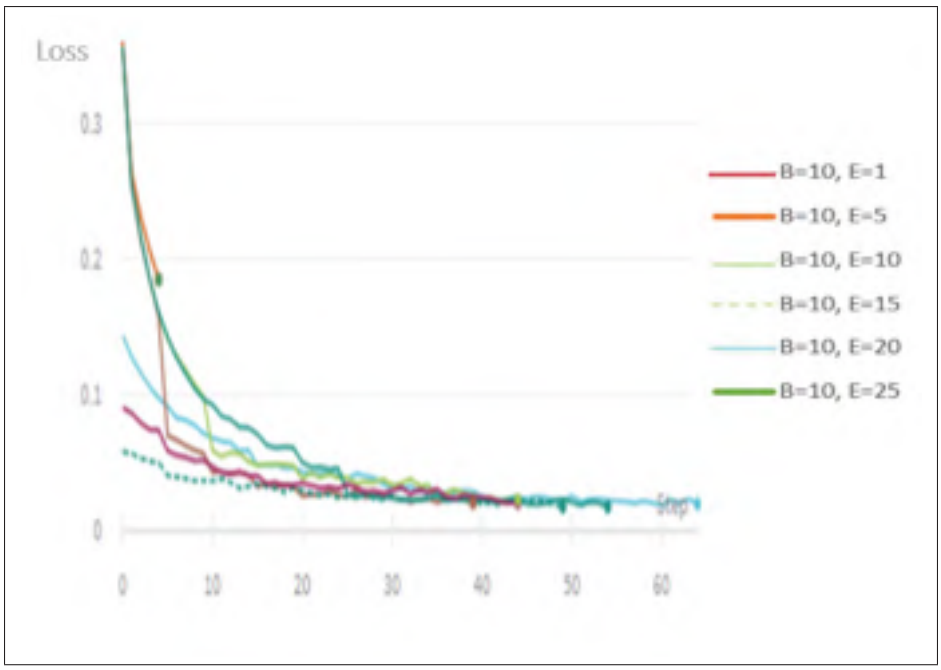


Figure 4.7 Taux d'erreur du modèle fédéré NSL-KDD (scénario 1)

Nous remarquons selon les deux figures 4.5 et 4.6 que dans les deux scénarios identifiés notre solution a atteint une précision de 100% sur le modèle locale après avoir effectuer des nouvelles mises à jours SGD par tour en fonction du nombre d'époques et la fraction du client, par conséquent en diminuant le nombre de tours de communication.

L'estimation de la fonction de perte qui est influencé directement par la valeur du taux d'apprentissage globale, était trés utiles pour guider le processus de minimisation de la fonctions d'erreur hautement non convexe du modèle sur les données de test NSL-KDD. Le taux d'apprentissage est un hyper-paramètre qui contrôle combien nous ajustons les poids de notre réseau en respectant le gradient de perte. Plus la valeur ηglobal est faible, plus nous nous déplaçons lentement le long de la pente descendante. Bien que cela puisse être une bonne idée (en

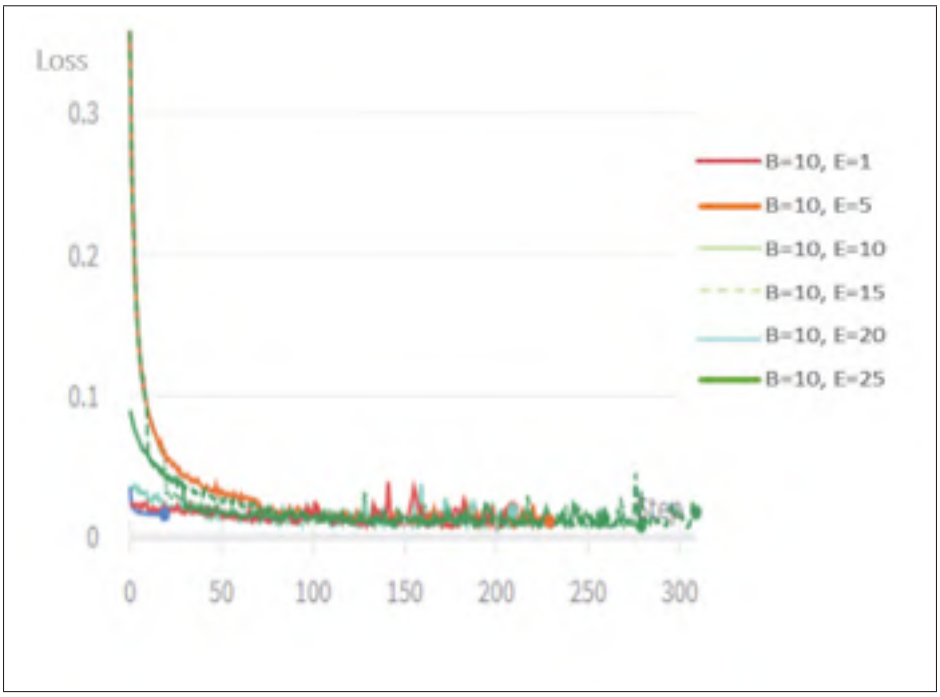


Figure 4.8 Taux d'erreur du modèle fédéré NSL-KDD (scénario 2)

utilisant un faible taux d'apprentissage) pour s'assurer que nous ne manquons aucun minimum local, cela pourrait également signifier que nous mettrons beaucoup de temps à converger.

Nous avons constaté par expérience que les deux modèles dans les scénario 1 et scénario 2 produisent une réduction significative de la perte sur l'ensemble de test de NSL-KDD.

4.4.3 Scénario 3

Dans ce test, nous voulons identifier les paramètres optimaux qui permettent d'augmenter la performance de l'algorithme d'encodeur automatique proposé. Pour tester la fiabilité de ce modèle centralisé sur l'ensemble de NSL-KDD, nous avons utilisé la courbe ROC AUC pour estimer le taux de détection du classificateur, en d'autres termes s'il peut détecter parfaitement des évènement anormaux en tant qu'un évènement anormaux et des évènement normaux en tant qu'un évènement anormaux.

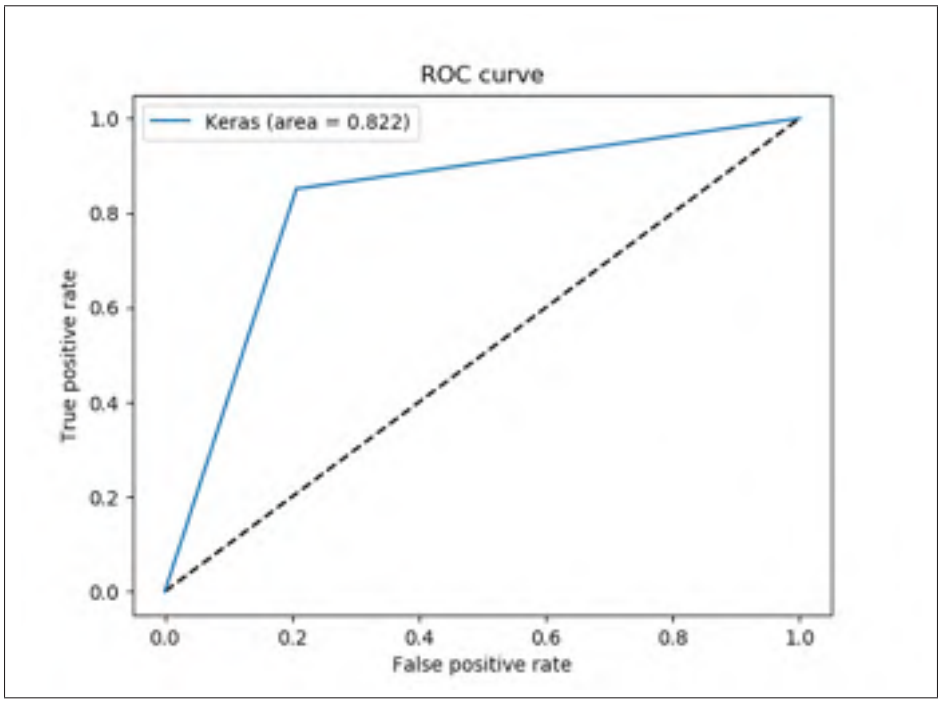


Figure 4.9 Roc curve du modèle NSL-KDD

Nous pouvons constater que la valeur auc=0.82 du classificateur keras est très proche à la valeur 1 ce qui signifie que l'algorithme d'auto-encodeur est robuste et il détecte parfaitement les anomalies sans fausses alarmes.

Lors de la phase d'évaluation de la performance sur l'ensemble de données de test qui accepte les caractéristiques d'origine et les caractéristiques prédites ou encore les sorties de l'autoencodeur, le modèle renvoie la perte de chaque échantillon de données, et enfin chaque échantillon de données est classé en fonction de son erreur de reconstruction avec un seuil prédéterminé.

La Figure 4.10 montre avec la parcelle de violon la distribution des valeurs d'erreur de reconstruction pour tous les échantillons de données dans l'ensemble de test, il est clair que les valeurs de perte des attaques sont pour la plupart plus élevées que la valeur seuil et l'inverse est vrai pour les échantillons normaux.

Pour mieux comprendre ce processus, nous avons présenté le taux d'erreur sur les deux ensembles d'apprentissage et de validation dans la Figure 4.11. Nous avons constaté une minimisation

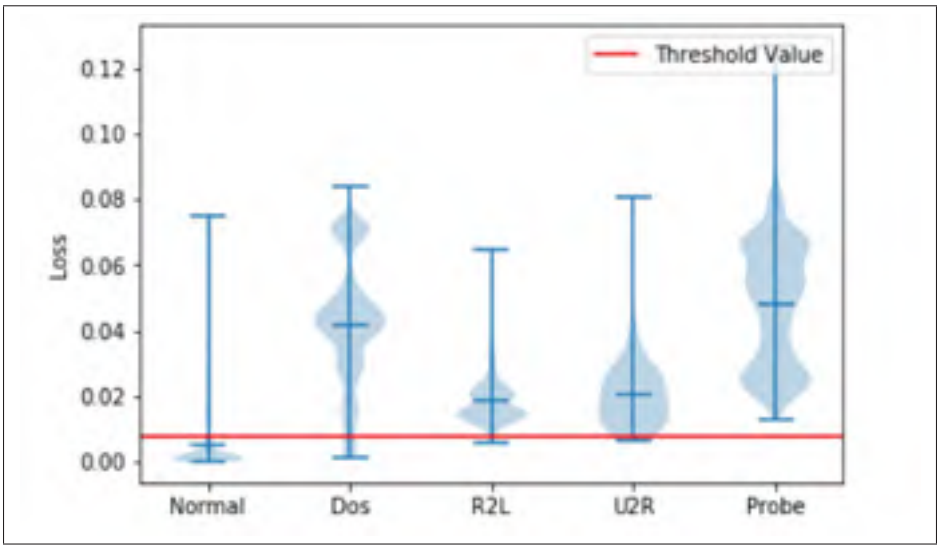


Figure 4.10 Reconstruction d'erreur avec un seuil de détection du modèle NSL-KDD centralisé

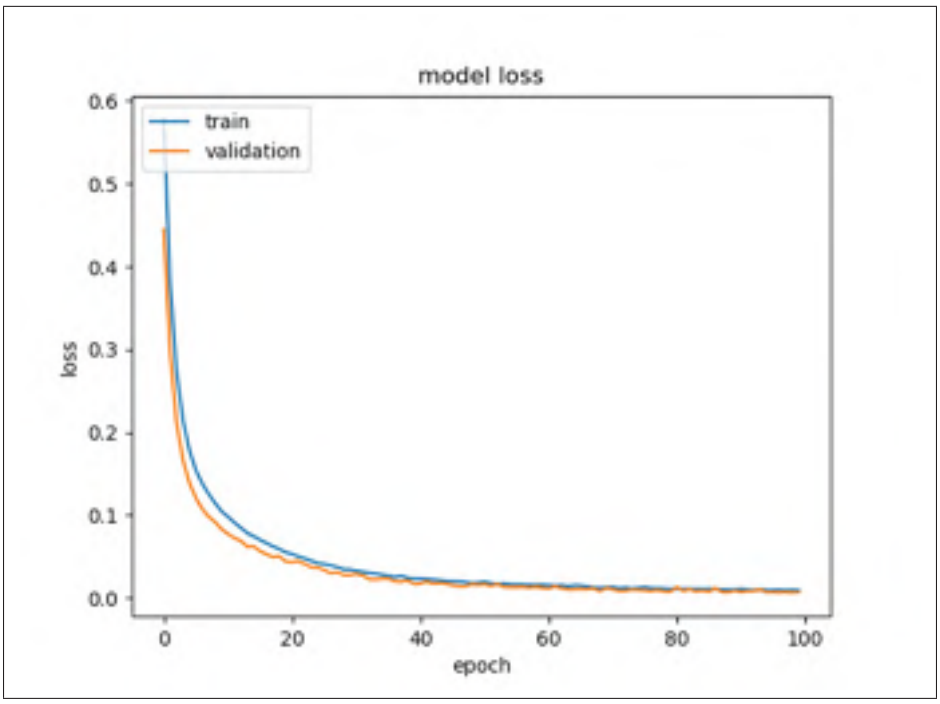


Figure 4.11 Taux d'erreur du modèle d'apprentissage NSL-KDD centralisé

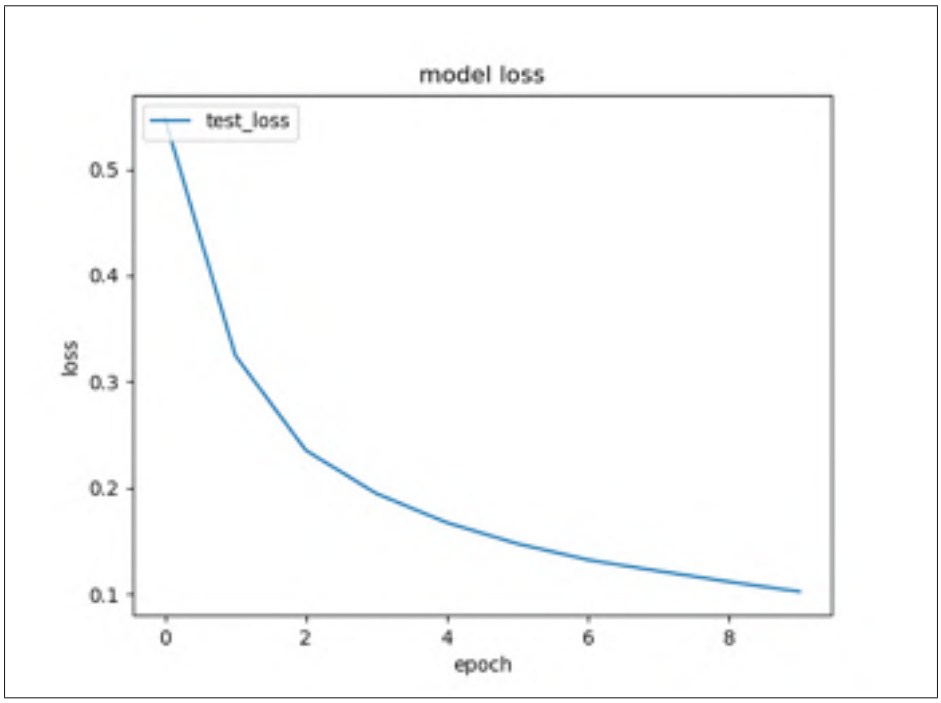


Figure 4.12 Taux d'erreur du modèle de test NSL-KDD centralisé

significative du taux d'erreur d'apprentissage avec des données normaux versus le taux d'erreur de validation testé sur 20 % de l'ensemble de validation NSL-KDD en utilisant le seuil de détection.

Les résultats concernant le seuil de détection sur l'ensemble de test sont performants et stables selon la Figure 4.12. Nous pouvons dire qu'ils surpassent de nombreuses autres approches, y compris d'autres approches que nous avons tentées dans le cadre de notre étude.

4.4.4 Scénario 4

La détection des anomalies est une forme de classification et une mise en œuvre sous forme de classification d'une classe, car une seule classe est représentée dans les données d'apprentissage. Un modèle de détection d'anomalies permet de prédire si un point de données est typique d'une distribution donnée ou non. Un point de données atypique peut être soit aberrant, soit un exemple d'une classe inédite. Normalement, un modèle de classification doit être formé

sur des données qui comprennent à la fois des exemples et des contre-exemples pour chaque classe afin que le modèle puisse apprendre à les distinguer. Comme nous avons déja précisé dans le chapire précédent, l'algorithme OCSVM comprend seulement les données normales sur l'ensemble d'apprentissage, c'est pour cela nous allons identifier quatres scénarios, en conservant le jeu de données Monday qui contient uniquement des données normales comme un ensemble d'entraînement et le reste des fichiers csv comme un ensemble de test. Nous avons donc effecctué les quatres scénario comme suit : Monday/Tuesday, Monday/Wednesday, Monday/Thursday, Monday/Friday.

Ce test nous permet d'appliquer les configurations optimales identifiés à partir de ces quatres scénarios afin de correctement détecter les attaques avec les données réelles de CIC-IDS 2017.

Nous avons évaluer ces modèles en utilisant les formules suivantes :

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4.7}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4.8}$$

$$F1 - score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{4.9}$$

Nous avons tracé les courbes de précision et de rappel résument le compromis entre le vrai taux positif et la valeur prédictive positive pour un modèle prédictif utilisant différents seuils de probabilité. Les courbes ROC sont appropriées lorsque les observations sont équilibrées entre chaque classe, tandis que les courbes précision-rappel conviennent aux ensembles de données déséquilibrés. De plus OCSVM est idéal pour utilisée roc curve et roc prc recall car on a une classe majeure ou positive (0 ou normale) en faisant predire la classe negative.

Nous avons tracé la courbes ROC (AUC) pour l'algorithme OCSVM, selon la Figure 4.13 il possède une valeur de AUC maximal égal à 0.74 ce qui signifie que ce classificateur détecte efficacement les anomalies.

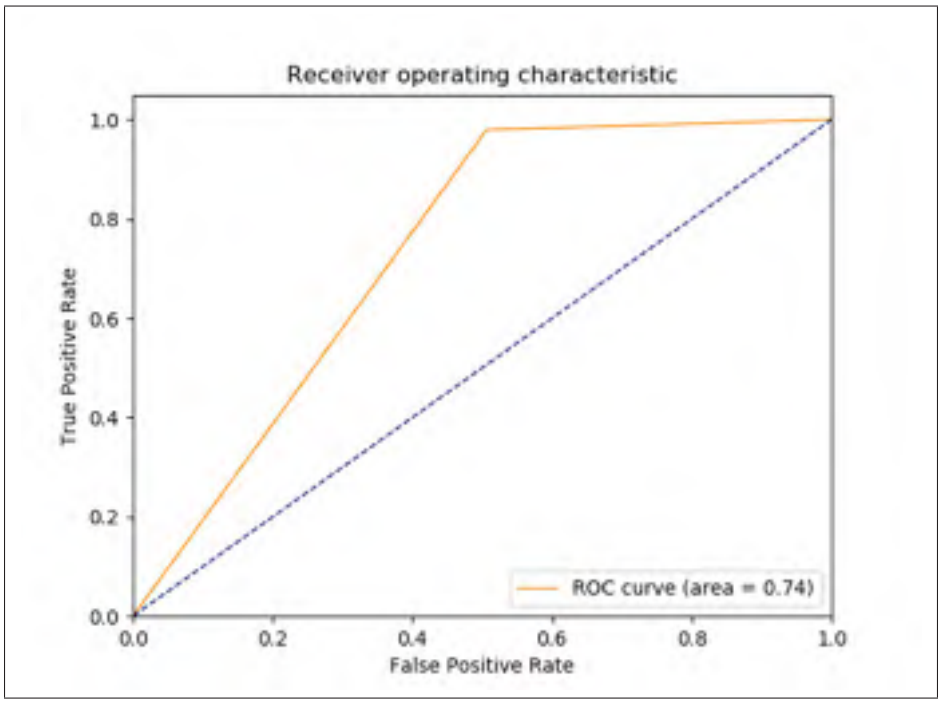


Figure 4.13 Roc curve du modèle CIC-IDS2017

Nous avons évalué la précision sur les quatre scénarios d'attaques en prenant en considération deux paramètres de controle nu et gamma qui comprennent plusieurs valeurs de test afin d'atteindre des résultats optimaux.

Selon la Figure 4.14, il est clair que les modèles avec les scénarios Monday/Wednesday et Monday/Friday atteignent la meilleure performance pour n'importe quelle valeur de nu et gamma respectivement. Les résultats étaient plus que satisfaisants, étant donné qu'un compromis optimal entre le taux de détection et le faux positif était enregistré.

De plus, nous remarquons que le modèle OCSVM offre une précision maximale avec des petites valeurs de nu et gamma allant de 0.01 à 0.07. Cependant avec des valeurs supérieures à 0.07, le modèle OCSVM devient beaucoup moins précis.

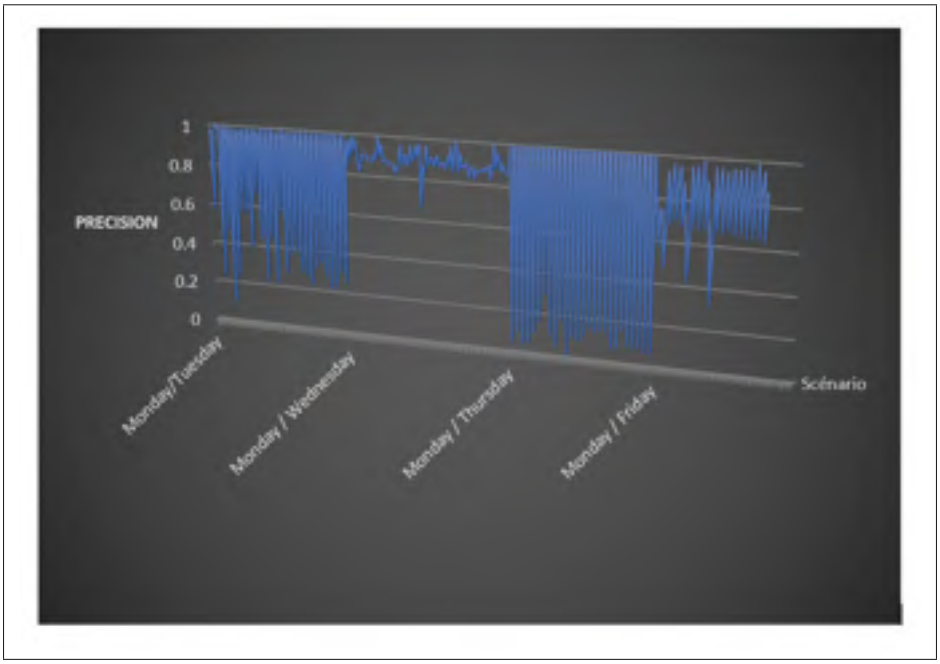


Figure 4.14 Test de la précision du modèle CIC-IDS2017

Nous avons testé les scénarios précédents basant sur plusieurs configurations pour arriver à des résultats optimaux, et par conséquent une détection efficace de l'algorithme OCSVM sur les données CIC-IDS2017 ce qui implique une influence directe sur la consommation de ressources.

4.5 Test de surconsommation

Les tests de surconsommation ont été faites avec les scénarios abordés précédèment pour les trois algorithmes proposés, dont nous avons estimé la surcharge en terme du processeur, de la RAM, disque et réseau. Pour se faire nous avons exploré l'outil Weights Biases l'un des outils de développement pour l'apprentissage en profondeur qui sert à garder trace d'hyperparamètres et métriques de système afin que nous puissions comparer les expériences et visualiser des graphiques d'apprentissage en temps réel.

4.5.1 Scénario 1 versus scénario 2

Dans cette partie est dédiée aux tests de surconsommaton de notre solution basée sur l'approche fédérée les résultas obtenus sont illustrés dans les deux figures 4.15 ET 4.16.

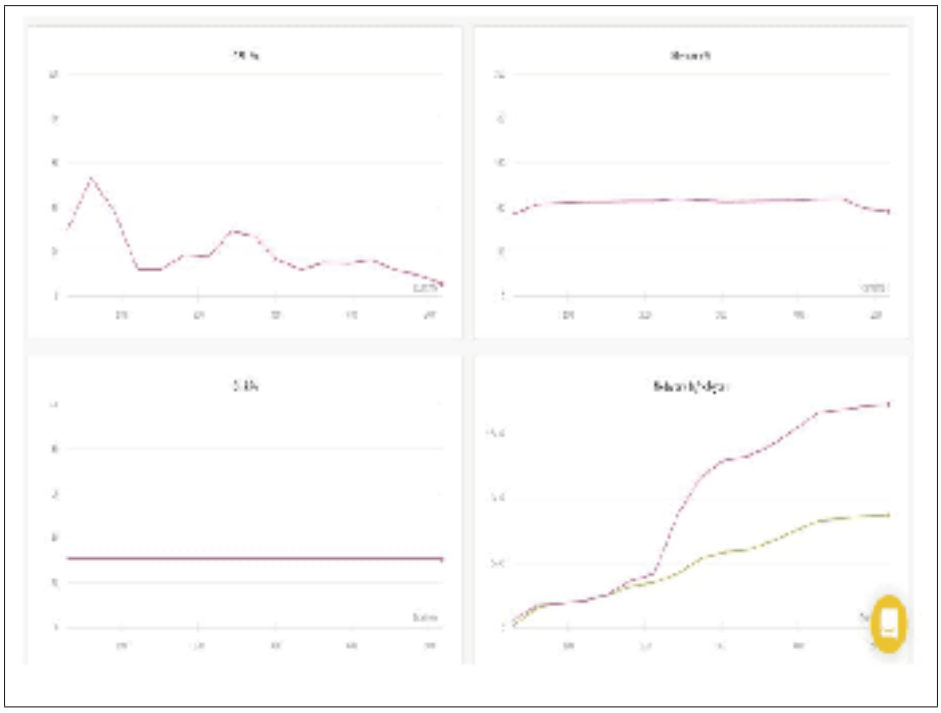


Figure 4.15 Surconsommation du modèle fédéré NSL-KDD (scénario 1)

Ces deux tests permet de mesurer la consommation de ressources du modèle proposé de NSL-KDD avec les différentes configurations (La taille du batch, nombre d'époques, la fraction du client) qui ont une influence directe sur la surconsommation.

Nous constatons que le modèle fédéré du scénario 1 est le moins coûteux en espace mémoire et en temps pour les configurations expérimentées par rapport au modèle fèdèré avec le scénario 2 qui est relativement plus coûteux surtout en terme de mémoire. En conséquence, la surconsommation augmente avec l'augmentation de la fraction des clients et en diminuant la taille des minibatch locale dont nous avons enregistré la consommation allant jusqu'à 54 % de CPU, 40 % de RAM et un mémoire de stockage de 30%.

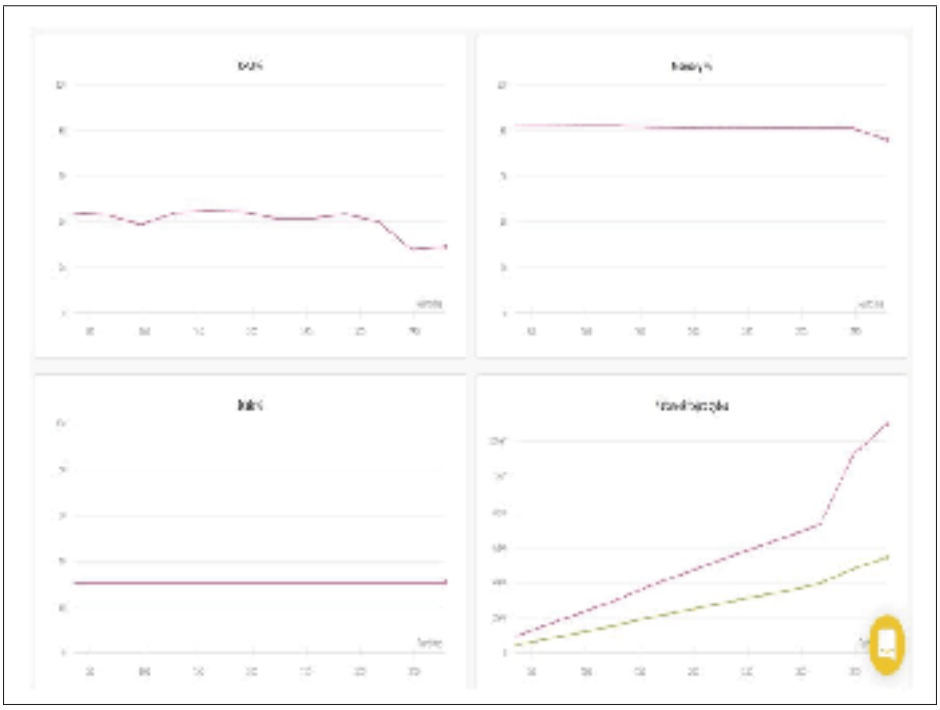


Figure 4.16 Surconsommation du modèle fédéré NSL-KDD(scénario 2)

La surconsommation du réseau a été mesuré en temps réel en calculant le total des octets envoyés et reçus par le serveur. RX Bytes(La ligne rouge) indique le volume de données en octets, reçus par le serveur , et Tx (la ligne verte) indique le nombre de communications transmises par le serveur comme le montre la Figure 4.15.

4.5.2 Scénario 3

Dans ce scénario, nous avons mesuré les ressources utilisées lors de la création du modèle NSL-KDD centralisé dans la Figure 4.17.

La surconsommation mesurée en CPU et mémoire pour les données de NSL-KDD centralisé augmente par rapport à taille du batch et le nombre d'époques effectué dont nous avons marqué une consommation de réseau allant jusqu'à 40 % de CPU et 20 % de RAM avec un batch égale à 100 en dix époques pour atteindre de taux de détection de 99%.

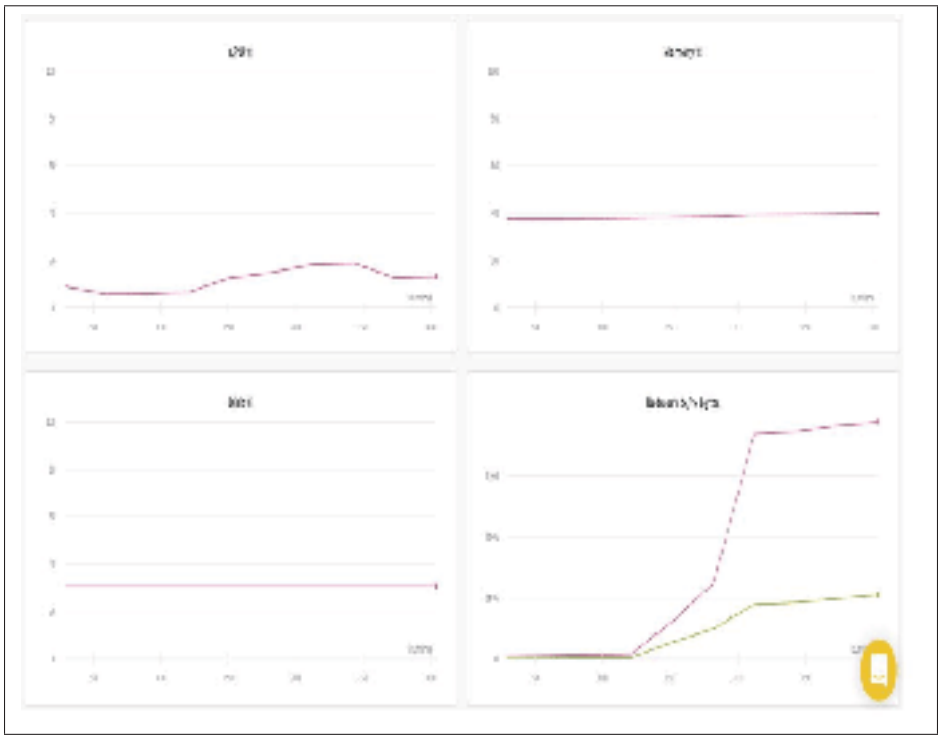


Figure 4.17 Surconsommation du modèle NSL-KDD centralisé

4.5.3 Scénario 4

Dans cette partie est dédiée aux tests de surconsommaton basée sur l'approche centralisée avec les données CIC-IDS2017.

Les résultas obtenus sont illustrés dans les deux figures 4.15 ET 4.16.

Nous avons mesuré les ressources consommées du modèle CIC-IDS2017 avec les paramètres optimales sélectionné selon les quatres scénarios lors de la phase du test de la section précédentes. Ce modèle permettent d'offrir une précision maximale de 100 %, cependant il est relativement coûteux en ressources à cause du nombre des données réelles élevées de ce modèle.

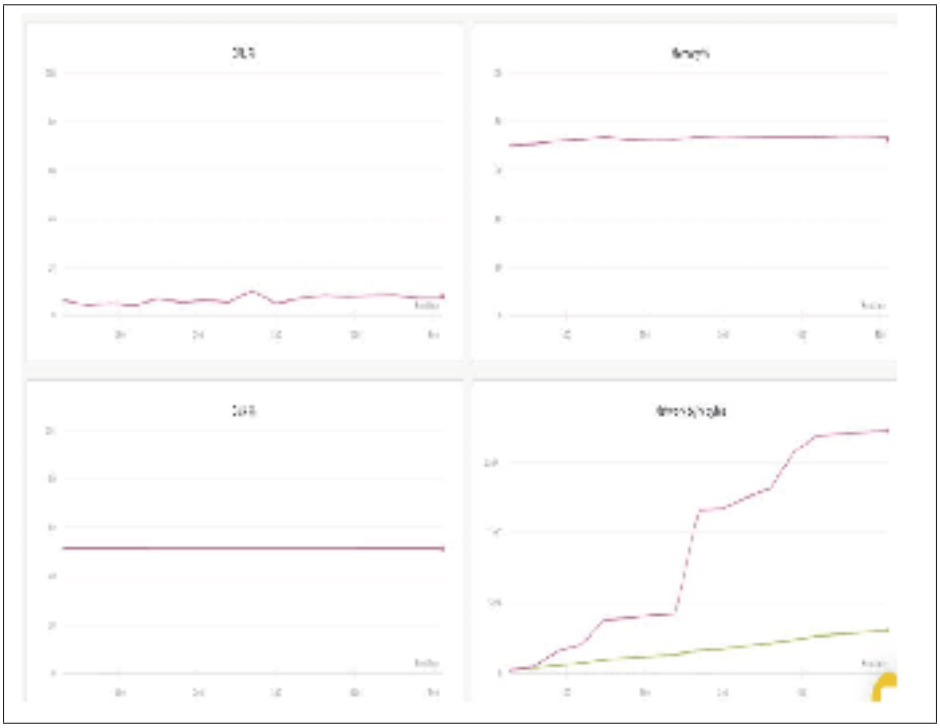


Figure 4.18 Surconsommation du modèle CIC-IDS2017

4.6 Conclusion

Les expériences ont montré que notre système de détection d'attaque distribuée et fédéré du scénario 1 est plus efficace par rapport au scénario 2 en appliquant l'algorithme FedAVG avec minibatch. Il a également été démontré que le modèle profond est plus efficace dans la détection des attaques que ses contre-parties peu profondes des deux approches centralisées étudiées de l'algorithme d'encodeur automatique et même pour l'algorithme OCSVM de machine learning traditionnelles. La surconsommation de ressources a été estimée en terme de CPU, mémoire, disque, et RAM pour comparer les différentes scénarios proposés dont l'approche décentralisée a surpassé les autres méthodes avec le scénario 2 avec un 40% de CPU et 32% de mémoire.

CONCLUSION ET RECOMMANDATIONS

Nous avons élaboré une étude comparative efficace entre l'approche centralisée en effectuant les techniques traditionnelles de machine learning et l'approche décentralisée basée sur l'apprentissage fédéré avec des données hétérogènes de la base de donées réference dansle domaine de la cybersécurité NSL-KDD pour la préservation de la vie privée.

Nos expériences montrent que l'apprentissage fédéré est trés pratique tant que l'optimisation de l'algorithme FedAvg forme un modèle robuste avec une réduction importante de tours de communication controlé par le parallélisme multi-clients comme nos résultats le montrent sur une variété de scénarios.

En comparaison avec les résultats obtenus de l'algorithme basique de FedSGD, nous avons constaté que l'optimisation des paramètres globaux de l'algorithme FedAVG est plus performante pour une distribution non-IID, et cela également avec la descente du gradient par minibatch sans et avec le moment Nesterov pour le modèle locale, tandis que Adadelta est prometteur en dépit de son meilleur performance sur entièrement des données non-IID du modèle globale du scénario 1.

Nous avons pu constater que notre solution pouvait contraindre l'apprentissage du modèle global, nécessitant d'avantage de temps et de ressources pour une fédération de clients plus élevée ce qui peut dégrader les performances prédictives. Toutefois, sauf dans les cas les plus extrêmes, des performances de classification restent satisfaisantes dont nous pouvons dire que notre plateforme est le premier IDS fédéré qui permet un compromis entre l'efficacité de détection et la consommation des ressources.

Lors de l'évaluation de l'approche centralisée avec l'algorithme auto-encodeur, nous avons évité la manipulation humaine du seuil afin d'obtenir des résultats reproductibles sans une intervention humaine. En revanche dans les réseaux réels lorsque le système est déployé l'administrateur

réseau peut régler manuellement le seuil qui permet un compromis entre la sensibilité et la spécificité en fonction des besoins du réseau qui est un énorme avantage par rapport à d'autres approches existantes. En terme de taux de détection et du temps d'apprentissage notre approche surpasse les deux approches centralisées étudiées et toutes les méthodes fédéré référence existante.

La qualité de la solution de notre algorithme dépend non seulement de la qualité de l'optimisation effectuée, mais également de la fonction de coût et de l'architecture utilisées. Une autre force de cette approche décentralisée est sa simplicité, il se compose de trois couches cachées seulement qui le rend très facile à contruire et particulièrement adapté pour l'apprentissage en ligne.

La limitation évidente de notre FIDS est qu'il ne peut pas différencier les différents types d'attaques, les travaux futurs peut être fait pour surmonter cette limitation en construisant un modèle qui se prolongent sa fonctionnalité afin d'obtenir la classification de cinq classes. Nous prévoyons d'appliquer notre approche sur des données non supervisées en utilisant les auto-encodeurs avec des données réelles comme une ligne de recherche intéressantes qui n'a pas été encore explorée dans la cybersécurité.

BIBLIOGRAPHIE

[1] F. Bourse, M. Minelli, M. Minihold, and P. Paillier. (2017). Fast homomorphic evaluation of deep discretized neural networks. CRYPTO 2017. 483-512.

[2] Xuanqing Liu, and Cho-JuiHsieh. (2018). Fast Variance Reduction Method with Stochastic Batch Size. arXiv :1808.02169v1 [cs.LG], 7 Aug 2018.

[3] Rakesh Agrawal & Ramakrishnan Srikant. (2000). Privacy-preserving data mining. ACM SIGMOD International Conference on Management of Data.

[4] Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. (2016). Stochastic variance reduction for nonconvex optimization.

[5] Robin C. Geyer, Tassilo Klein, and Moin Nabi.(2017). Differentially private federated learning : A client level perspective. NIPS. vol2.

[6] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. (2018). How To Backdoor Federated Learning.

[7] Takayuki Nishio and Ryo Yonetani. (2018). Client selection for federated learning with heterogeneous resources in mobile edge.

[8] Rie Johnson, and Tong Zhang. (2013). Accelerating stochastic gradient descent using predictive variance reduction. NeurIPS. Vol-1.

[9] Guanghui Lan & Yi Zhou. (2017). An optimal randomized incremental gradient method. Mathematical programming. pages 1–49.

[10] Guanghui Lan and Yi Zhou. (2018). Random gradient extrapolation for distributed and stochastic optimization. SIAM. 28. 4. 2753–2782.

[11] L.Bottou. (2012). Stochastic gradient descent tricks. Neural Networks :Tricks of the Trade.

[12] H.Brendan McMahan, E.Moore, Daniel Ramage, S.Hampson, B.Agüera y Arcas. (2017). Communication efficient learning of deep networks from decentralized data.

[13] Shai Shalev-Shwartz and Tong Zhang. (2013). Stochastic dual coordinate ascent methods for regularized loss. Journal of Machine Learning Research,14(1) :567–599.

[14] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. (2016). Applying Neural Networks Encrypted Data with High Throughput and Accuracy.

[15] Virginia Smith, Chao-Ka Chiang, Maziar Sanjabi, and Ameet Talwalkar. (2017). Federated multi-task learning. Neural Information Processing Systems. vol-2 .

[16] Peter Richtárik & Martin Takáč. (2016). Stochastic reformulation of linear systems and fast stochastic iterative methods. Technical report, 2016.

[17] Wenliang Du and Zhijun Zhan. (2002). Building decision tree classifier on private data. IEEE. Volume 14.

[18] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, and Rui Zhang. (2018). A hybrid trust model for distributed differential privacy.

[19] B T. Polyak. (1964). Some methods of speeding up the convergence of iteration methods. Computational Mathematics and Mathematical Physics. 4. 5. 1–17.

[20] Jakub Konečný, Jie Liu, Peter Richtárik, and Martin Takáč. (2015). MINI BATCH SEMI STOCHASTIC GRADIENT DESCENT IN THE PROXIMAL SETTING.

[21] Jun Furukawa, Y. Lindell, Ariel Nof, and Or Weinstein. (2016). High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority.

[22] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, Emmanuel Prouff. (2017). Privacy preserving classification on deep neural network.

[23] Herbert Robbins & Sutton Monro. (1951). A stochastic approximation method. The Annals of Probability and The Annals of Statistics. 22. Volume-3. 400–407.

[24] Irene Giacomelli, Somesh Jha, Marc Joye, C. David Page, and K. Yoon. (2017). Privacy preserving ridge regression with only linearly homomorphic encryption.

[25] O. Goldreich, S. Micali, and A. Wigderson. (1987). How to play any mental game. ACM.

[26] Rob Hall, Stephen E. Fienberg, and Yuval Nardi. (2011). Secure multiple linear regression based on homomorphic encryption. Journal of Official Statistics. 669–691.

[27] Q.Yang, Y.Liu, & T.Chen. (2019). Federated Machine Learning :Concept and Applications.

[28] J. Sicking, F. Hger, Peter Schlicht, Tim Wirtz, M. Kamp, L Adilova, Stefan Wrobel. (2018). Efficient decentralized deep learning by dynamic model averaging.

[29] T Yang, G Andrew, H Eichner, H. Sun, W Li, N Kong, D Ramage, and F Beaufays. (2018). Applied federated learning : Improving Google keyboard query suggestions.

[30] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan.(2018). When edge meets learning : Adaptive control for resource-constrained distributed machine learning, CoRR, vol. abs/1804.05271.

[31] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, A Suresh Dave Bacon. (2016). Federated learning : Strategies for improving communication efficiency.

[32] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Aguera y Arcas. (2016). Federated learning of deep networks using model averaging.

[33] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. (2015). Parallel training of DNNs with Natural Gradient and Parameter Averaging. ICLR. v8.

[34] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong- Lyun Kim. (2018). On-Device Federated Learning via Blockchain and its Latency Analysis.

[35] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, & Xiaoqian Jiang. (2018). Secure logistic regression based on homomorphic encryption.

[36] H. Brendan McMahan, and Daniel Ramage. (2017). Federated learning : Collaborative machine learning without centralized training data.

[37] Florian Hartmann. (2018). Federated learning, https ://florian.github.io/federated-learning/.

[38] Dominik Csiba, and Peter Richtárik. (2015). Primal method for ERM with flexible mini-batching schemes and non-convex losses.

[39] Dominik Csiba and Peter Richtárik. (2016). Coordinate descent face-off : primal or dual ? arXiv :1605.08982, 2016.

[40] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. (2017). Oblivious neural network predictions via MiniONN transformations.

[41] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. (2016). Federated learning of deep networks using model averaging.

[42] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. (2017). Learning differentially private language models without losing accuracy.

[43] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. (2018). Inference attacks against collaborative learning.

[44] P Mohassel & P Rindal. (2018). ABY3 :A mixed protocol framework for machine learning.

[45] A. Defazio, F. Bach, S. L. Julien. (2014). SAGA :A fast incremental gradient method with support for non-strongly convex composite objectives.

[46] Seb Arnold. (2016). An Introduction to Distributed Deep Learning.

[47] Payman Mohassel and Yupeng Zhang. (2017). SecureML : A system for scalable privacy - preserving machine learning. IACR Cryptology. IEEE. pages 19–38.

[48] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, & Nina Taft. (2013). Privacy preserving ridge regression on hundreds of millions of records.

[49] Yann A LeCun, Léon Bottou, Genevieve Borr, and Klaus-Robert Muller. (2012). Efficient backprop. In Neural networks : Tricks of the trade. Springer. 9–48.

[50] Qirong Ho, James Cipar, Henggang Cui, Jin Kyu Kim, Seunghak Lee, Phillip B. Gibbons, Garth A. Gibson, Gregory R. Ganger, & E P. Xing. (2013). MORE EFFECTIVE DISTRIBUTED ML VIA A STALE SYNCHRONOUS PARALLEL SERVER.

[51] Le T. Phong, Y Aono, T Hayashi, L Wang, & S Moriai. (2018). Privacy-preserving deep learning via additively homomorphic encryption. IEEE. 1333–1345.

[52] M. S Riazi, C Weinert, O Tkachenko, E M. Songhori, T Schneider, & F Koushanfar. (2018). Chameleon : A hybrid secure computation framework for ML applications.

[53] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos (1978). On data banks and privacy homomorphisms. Academic Press, Inc. pages 169–179.

[54] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. (2017). DeepSecure : Scalable provably-secure deep learning. CoRR abs/1705.08963.

[55] Jie Liu, Martin Takáč. (2017). Projected Semi-Stochastic Gradient Descent Method with Mini-Batch Scheme under Weak Strong Convexity Assumption.

[56] Léon Bottou. (2009). Curiously fast convergence of some stochastic gradient descent algorithms. The Symposium on Learning and Data Science. vol-1.

[57] Amit P. Sheth & James A. Larson. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases.

[58] Reza Shokri & Vitaly Shmatikov. (2015). Privacy-preserving deep learning. ACM SIGSAC.

[59] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, & K. He. (2018). Accurate large minibatch SGD :Training ImageNet in 1 hour.

[60] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang.(2016). Scalable and secure logistic regression via homomorphic encryption. ACM. 142–144.

[61] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. (2013). Stochastic gradient descent with differentially private updates. IEEE, 245–248.

[62] Lili Su and Jiaming Xu. (2018). Securing distributed machine learning in high dimensions.

[63] Philipp Moritz, Robert Nishihara, Michael Jordan. (2016). A linearly convergent stochastic LBFGS algorithm. Artificial Intelligence and Statistics.

[64] Jaideep Vaidya, and Chris Clifton. (2004). Privacy preserving naïve Bayes classifier for vertically partitioned data. SIAM Conference on Data Mining. 330–334.

[65] Jaideep Vaidya, and Chris Clifton. (2002). Privacy preserving association rule mining in vertically partitioned data. Knowledge Discovery and Data Mining(KDD'02).

[66] Jaideep Vaidya, and Chris Clifton. (2003). Privacy- preserving K-means clustering over vertically partitioned data. Knowledge Discovery and Data Mining (KDD'03).

[67] Jaideep Vaidya and Chris Clifton. (2005). Privacy-preserving decision trees over vertically partitioned data. Springer. 139–152.

[68] Yurii.E. Nesterov. (2004). Introductory Lectures on Convex Optimization : A Basic Course.

[69] Robert Mansel Gower and Peter Richtárik. (2016). Randomized quasi-Newton updates are linearly convergent matrix inversion algorithms. arXiv :1602.01768.

[70] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc Viet Le, and Andrew Yan-Tak Ng. (2011). On optimization methods for deep learning.

[71] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, & Vikas Chandra. (2018). Federated Learning with Non-IID Data. arXiv :1806.00582[cs.LG].

[72] Andrew C. Yao. (1982). Protocols for secure computations. IEEE.

[73] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. (2018). When edge meets learning : Adaptive control for resource-constrained distributed machine learning. CoRRabs/1804.05271.

[74] Divya Rana. (2013). One Class SVM Vs SVM Classification. IJSR. 2319-7064.

[75] Jiawei Yuan & Shucheng Yu. (2014). Privacy preserving back-propagation neural network learning made practical with cloud computing. IEEE. 25. Vol-1. 212–221.

[76] Q Zhang, Laurence T. Yang, & Zhikui Chen. (2016). Privacy preserving deep computation model on cloud for big data feature learning. IEEE. 65. 5. 1351–1362.

[77] Inci M. Baytas, Ming Yan, Anil K. Jain, and Jiayu Zhou.(2016). Asynchronous Multi-Task Learning. IEEE International Conference on Data Mining. 11–20.

[78] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz.(2016). Revisiting distributed synchronous stochastic gradient descent.

[79] Xinyu Peng, Li Li, & Fei-Yue Wang. (2019). Accelerating Minibatch Stochastic Gradient Descent using Typicality Sampling. IEEE trans on NNLS. 1-11

[80] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. (2009). A detailed analysis of the KDD CUP 99 data set. IEEE Symposium. pp.1–6.

[81] Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro. (2013). Minibatch primal and dual methods for support vector machines.

[82] S. Stolfo. (2018). http ://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[83] Martin Abadi et al. (2016). TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems, Google Research. arXiv :1603.04467v2 [cs.DC].

[84] Jakub Konečný, Zheng Qu, and Peter Richtárik. (2014). Semi-stochastic coordinate descent.

[85] Stefan Axelsson. (2000). The Base-Rate Fallacy and the Difficulty of Intrusion Detection. Ericsson Mobile Data Design AB. ACM. 2. 3. 186–205.

[86] Zeiler, M.D. (2012). ADADELTA : An adaptive learning rate method.

[87] S. S Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter Pegasos. (2011). Primal estimated sub-gradient solver for SVM. Mathematical programming.

[88] scikit-learn : machine learning in python — scikit-learn 0.20.0 documentation. [online]. available : http ://scikit-learn.org/stable/. [accessed : 12-oct-2018].

[89] Kingma, D. and Ba, J. (2014) Adam : A method for stochastic optimization.

[90] Mennatallah Amer, Markus Goldstein, Slim Abdennadher. (2013). Enhancing one-class Support Vector Machines for unsupervised anomaly detection.

[91] Zheg Chai, H. Fayyaz, Z. Fayyaz, Ali Anwar, Yi Zhou, Nathalie Baracaldo, Heiko Ludwig , & Yue Cheng. (2019). TOWARDS TAMING THE RESOURCE AND DATA HETEROGENEITY IN FEDERATED LEARNING..

[92] S Truex, N Baracaldo, A Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, & Yi Zhou. (2019). A Hybrid Approach to Privacy-Preserving Federated Learning.

[93] Iman Sharafaldin, Arash Habibi Lashkari and Ali A. Ghorbani. (2017). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization.

[94] Arif Yulianto, Parman Sukarno, & Novian Anggis Suwastika. (2019). Improving AdaBoost based Intrusion Detection System. IEEE trans on cybernetics. 38(2) :577-83.

[95] Yurii. E . Nesterov. (1983). A method of solving a convex programming problem with convergence rate O(1/k²). Soviet Mathematics Doklady. No. 2. volume. 27.

[96] Duch, Hazan, and Singer. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research.

[97] Y Xiao, H Wang, L Zhang, and W Xu. (2014). Two methods of selecting Gaussian kernel parameters for one-classSVM and their application to fault detection.

[98] Sébastien Godard, Nicolas Voisine, Tanguy Urvoy, Vincent Lemaire. (2019).Apprentissage fédératif pour la prédiction du churn : une évaluation. EGC. 35. 141-152.

[99] Z. Zhou, X. Chen, E. Li, L. Zeng, Ke Luo, & Junshan Zhang. (2019). Edge Intelligence : Paving the Last Mile of Artificial Intelligence with Edge Computing.

[100] Hanley, James A, & Barbara J McNeil. (1982). The meaning and use of the area under a receiver operating characteristic. Radiology. no 1. vol.143. 29-36.

[101] Jakub Konečný & Peter Richtárik. (2015). Semi-Stochastic Gradient Descent Methods.

[102] A. A. Diro, and N. Chilamkurti. (2017). Distributed attack detection scheme using deep learning approach for Internet of Things. Elsevier. Vol-82. 761-768

[103] R Vinayakumar, M Alazab, K P.Soman, P.Poornachandran, A Al-Nemrat, S Venkatraman. (2019). Deep Learning Approach for Intelligent Intrusion Detection System.

[104] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau Snips. (2019). FEDERATED LEARNING FOR KEYWORD SPOTTING.

[105] Vukasin Felbab, Péter Kiss & Tomáš Horváth. (2019). Optimization in Federated Learning.