Machine learning-Enabled Network Traffic Analysis

Ons Aouedi

DOCTORAT BRETAGNE LOIRE / MATHSTIC

Nantes Université

THÈSE DE DOCTORAT DE

NANTES UNIVERSITÉ

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Ons AOUEDI

Machine Learning-Enabled Network Traffic Analysis

Thèse présentée et soutenue à Nantes, le 02 Décembre 2022
Unité de recherche : Laboratoire des Sciences de Numérique de Nantes (LS2N), UMR 6004

Rapporteurs avant soutenance :

Adlen KSENTINI	Professeur, Eurecom, Sophia Antipolis, France
Sonia BEN MOKHTAR	Directrice de Recherche CNRS/INSA LYON, France

Composition du Jury :

Président :	Yassine HADJADJ-AOUL	Professeur, Université de Rennes I, France
Examinateurs:	Adlen KSENTINI	Professeur, Eurecom, Sophia Antipolis, France
	Sonia BEN MOKHTAR	Directrice de Recherche CNRS/INSA LYON, France
	Yusheng JI	Professeure, National Institute of Informatics, Japon
Dir. de thèse :	Benoît PARREIN	Maître de Conférences HDR, Nantes Université
Co-encadrant. de thèse :	Kandaraj PIAMRAT	Maître de Conférences, Nantes Université

"Who fears climbing the mountains lives forever among the pits".

— Abou el Kacem Chebbi

Acknowledgments

This has been an enriching and tough journey, full of ups and down, full of joy and challenges, and especially full of wealthy lessons. I thank God, who gave me the positive energy and opportunity to carry out this work. Also, I was surrounded by many supportive peoples who have given me the strength to go through difficult times. So, it was a great pleasure to thank them all for their positive impact.

My initial gratitude goes to my supervisors Kandaraj PIAMRAT and Benoît PARREIN for giving me the chance to work on an interesting subject. I am grateful for their trust, technical, and emotional support during these three years. Their advice at the scientific and personal level has resulted in the production of this work. I would thank them again for having shown my weaknesses, and strengths, as well as for giving me the freedom to collaborate with other researchers around the world. I am really indebted to them.

I would also like to express my gratitude to the jury members of my defense. Many thanks to Adlen KSENTINI and Sonia BEN MOKHTAR who agreed to read and review my thesis. I want to thank Yassine HADJADJ-AOUL and Yusheng JI for agreeing to participate in the jury.

I am grateful to my colleague Alexis Bitaillou for all the technical discussion, the good mood in the team, his help, and for proofreading the abstract in French.

A special and deepest gratitude goes to my parents, my father Khmais, and my mother Faouzia. My parents have given me everything through their endless sacrifices for my success. My love and sincere thanks to my brothers Neder, Seifedine, and Aymen with his wife Maria, as well as my sister Arwa and her husband Ahmed for their love and support in the good and the bad times. Also, I cannot finish without expressing my thanks to my little nieces Awess and Alma for bringing joy to my soul even though we are far away.

Last but not least, I am thankful to Nantes University for awarding me the PhD research scholarship, and to all the people from the administration team who helped me many times.

Finally, I am grateful to all my loved ones, all everyone who helped me during the PhD and to whom I was able to meet during this trip.

Abstract

Recent development in network communication along with the drastic increase in the number of smart devices leads to an explosion in data generation. To this end, intelligent network traffic analysis can help to understand the behavior of connected smart devices and applications as well as provides defense against cyber-attacks. In this line, Machine Learning (ML) and Deep Learning (DL) models have the ability to model and uncover hidden patterns using training data or environment. Despite their benefits, major challenges need to be addressed such as model generalization (due to model overfitting), lack of label (due to the difficulty to label all the data), and privacy (due to recent regulations). In this thesis, new ML/DL-based models are proposed for tackling these challenges. The first contribution focuses on improving the generalization and classification performance by proposing an ensemble blending model. The simulation results show that the accuracy of the proposed ensemble model is 10%, better than some state-of-the-art models. Second, a semi-supervised model has been proposed and the experiment results show that unlabeled data boost the classification accuracy by 11% in comparison to its supervised version. Finally, a Federated Learning (FL) based Intrusion Detection System (IDS) has been proposed. It allowed the clients to learn an efficient intrusion detection model without the need to label their local data as well as to achieve high classification performance and improvement in terms of communication overhead (reduction by almost 75% in comparison to a centralized model).

Résumé

L'Internet des Objets entraînent par son nombre de terminaux une explosion du trafic de données. Pour augmenter la qualité globale de réseau, il est possible d'analyser intelligemment le trafic réseau afin de détecter d'éventuel comportement suspect ou malveillant. Les modèles d'apprentissage automatique et d'apprentissage profond permettent de traiter ce très grand volume de données. Néanmoins, il existe certaines limites dans la littérature, notamment la confidentialité des données, le surapprentissage (manques de diversité dans les données) ou tout simplement le manque de jeu de données labélisées. Dans cette thèse, nous proposons de nouveaux modèles s'appuyant sur l'apprentissage automatique et l'apprentissage profond afin de traiter une grande quantité de données tout en préservant la confidentialité. Notre première approche utilise un modèle d'ensemble. Les résultats montrent une diminution du surapprentissage, tout en augmentant de 10% la précision comparé à des modèles de l'état de l'art. Notre seconde contribution s'attache aux problèmes de disponibilité des données labélisées. Nous proposons un modèle d'apprentissage semi-supervisé capable d'améliorer la précision de 11% par rapport à un modèle supervisé équivalent. Enfin, nous proposons un système de détection d'attaque s'appuyant sur l'apprentissage fédéré. Nommé FLUIDS, il permet de réduire la surcharge réseau de 75% (comparé à son équivalent centralisé) tout en préservant de très haute performance et la confidentialité.

Contents

List of Figures

List of Tables

List of Abbreviations

AE AutoEncoder

AI Artificial Intelligence

ANN Artificial Neural Networks

Bi-LSTM Bidirectional-LSTM

CFS Correlation-based Feature Selection

CM Confusion Matrix

CML Centralized ML

CNN Convolution Neural Networks

CPS Cyber physical Systems

DDoS Distributed Denial of Service

DL Deep Learning

DoS Denial of Service

DPI Deep Packet Inspection

DT Decision Tree

EFB Exclusive Feature Bundling

FedAvg Federated Averaging

FL Federated Learning

GBDT Gradient Boosting Decision Tree

GDPR General Data Protection Regulation

GOSS Gradient-based On Side Sampling

GRU Gated Recurrent Unit

HIDS Host-based IDS

IDS Intrusion Detection Systems

IG Information Gain

IIoT Industrial IoT

IoT Internet of Things

ISP Internet Service Providers

JAP Joint Announcement Protocol

KNN K-Nearset Neighbour

LSTM Long Short-Term Memory

ML Machine Learning

MLP Multi-Layer Perceptron

NB Naive Bayes

NIDS Deep Packet Inspection

PC Principal Component

PCA Principal Component Analysis

QoS Quality of Service

ReLU Rectified Linear Unit

RF Random Forest

RFE Recursive Feature Elimination

RL Reinforcement Learning

RNN Recurrent Neural Networks

SAE Stacked AutoEncoder

SDN Software Defined Networking

SSAE Stacked Sparse AutoEncoder

SVM Support Vector Machine

XAI eXplainable AI

General Introduction

1.1 Motivation

Recent development in network communication along with the drastic increase in the number of smart devices as well as the Internet of Things (IoT) leads to an explosion in data generation and heterogeneity. For example, by 2023, 5G will generate nearly 3× more traffic than 4G. Also, according to the latest Cisco forecast, by 2030 the number of connected IoT devices will surpass 500 million [1]. Meanwhile, mobile data traffic will be 77 Exabytes per month, which is 7 times that in 2017 [2]. At the same time, the heterogeneous traffic coming from smart vehicular, mobile, and Industrial IoT (IIoT) requires efficient network resources and architecture in order to maintain the Quality of Service (QoS) to the end-user. These made the network architecture highly resource-hungry, calling for ultra-efficient, fast, and autonomous network traffic analysis approaches. In addition, security and privacy concerns are becoming more stringent for 5G and beyond networks.

In this context, accurate traffic analysis helps to understand the behavior of connected smart devices and applications as well as provides defense against security attacks. In particular, major problems in traffic analysis can be broadly divided into two categories, which are (i) traffic classification based on flow and packet-based features (attribute), and (ii) traffic prediction using time series data [3]. The purpose of traffic classification is to understand the type of traffic carried on the Internet [4] [5]. It aims to identify the application (YouTube, Netflix, Twitter, etc.) or detect network attacks. On the other hand, traffic prediction aims to forecast the status of network links or the total amount of traffic expected based on historical data [6]. It is often faced as a time-series forecasting problem [7].

Indeed, network traffic classification poses significant challenges in computation time, complexity, and data privacy. Combined with the increasing and heterogeneous traffic, Machine Learning (ML) based approaches are opening the ways to model, learn, and recognize the complex patterns within the network traffic behavior using training data or environment [8]. Besides, key technological advances in networking, such as Software-Defined Networking (SDN), promote the use of ML in networking. Recent research has demonstrated the benefits of ML with traffic classification and intrusion detection systems (IDS). It presents a key advantage in managing network traffic and in turn, makes the network self-managed,

and self-adaptive. Also, it provides the network operators with more intelligence, autonomy, and less human intervention as maximum as possible. In the same direction, ML can be a promising solution for data processing in a (near)-real-time manner. Despite these benefits, there are some major challenges that need to be addressed such as model generalization, partially labeled data, privacy preservation of the user data, etc.

1.2 Research questions

To formulate the scope of this thesis, we pose several research questions as follows:

- *What are the most suitable ML-based models for network traffic classification?*
  First, we need to study the performance of different ML models for network traffic classification. Among such models applied to network traffic classification so far, no model outperforms all the others. More specifically, each model has its advantages and weaknesses, thus it is a risky and difficult task to find the best model.

- *How to deal with unlabeled data?*
  Second, as new applications and attacks emerge every day, it is not possible to have all the flow labeled in a real-time manner. At the same time, labeling all the traffic requires a huge effort from human annotators sometimes with a specific domain of expertise. In this context, the authors in [9] indicate that labeling all the traffic is a hard task and thus one of the most obvious obstacles to progress on traffic classification. On the other hand, since the unlabeled data provide informative characteristics, we can use them to improve the performance of the ML-based models.

- *How to build an ML model under the privacy concerns?*
  Finally, despite the profitable use of ML models, these data-driven methods are facing issues such as the scarcity and privacy of user data. For example, strict laws such as the General Data Protection Regulation (GDPR) in European Union [1] completely redefined the data management policy. Also, the increase in the amount of network traffic could decrease the scalability of the model, bottleneck the whole network, and cause an extra computational cost for both storage and processing. Altogether, this research question corresponds to the network traffic processing under privacy constraints as well as reducing the communication overhead.

1.3 Thesis Contributions

The purpose of this thesis is to answer the above research questions. In particular, this thesis aims to investigate the application of ML/DL-based models for intelligent traffic analysis

[1] https://gdpr-info.eu/issues/data-protection-officer/

including traffic classification and IDS. The proposed solutions demonstrate the ability of these models to improve performance in terms of accuracy, complexity, the communication overhead. It is important to note that since the contributions of this thesis focus on ML/Deep Learning (DL)-based models, the conventional and traditional techniques applied to network traffic analysis will not be considered. The following items highlight our contributions to applying ML for network traffic analysis.

- **Improving the generalization of traffic classification:** Despite the performance of ML and DL models for network traffic classification, they experience overfitting and low-bias problems. In particular, such models perform well on the training set, whereas this is not the case with the unseen data (i.e. test data). A straightforward solution to this issue is to propose a blending ensemble through the combination of different DL and tree-based models. The tree-based models are used as base classifiers and DL has been used as a meta-classifier in order to correct the errors that occur during the learning process of the base classifiers as well as learn the non-linear relationship among the base classifiers. We show that the proposed ensemble prevents overfitting and reduces bias simultaneously to some extent, in addition, to achieve good results on both non-encrypted and encrypted network traffic.

- **Enhancing model performance with unlabeled data:** As new types of traffic emerge every day and are generally partially labeled, this opens the question of how to accurately classify traffic using a limited amount of labeled data or partially labeled data. Based on these reasons, we reformulate traffic classification into semi-supervised learning where both supervised learning (using labeled data) and unsupervised learning (unlabeled data) are combined. The main motivations of this approach are: (i) unlabeled data is often abundantly and easily available; (ii) classification performance of the whole model can be greatly improved when a large amount of unlabeled traffic is included in the training process; (iii) there is a limit to how much human effort can be thrown at the labeling problem. In particular, we study how unlabeled data can impact the performance of the whole model. The proposed approach shows the advantage of the unlabeled data helps to extract high-level feature representations through the pre-trained strategy and in turn, boosts the traffic classification.

- **Providing traffic privacy:** Network traffic can contain private data and sending them to a central entity may affect user privacy. Also, IDS requires fast analysis while centralized processing is time-consuming. To cope with the above limitations, a semi-supervised, Federated-Learning (FL), based IDS has been proposed, called `FLUIDS`. More specifically, the clients train an unsupervised model using unlabeled data and the FL server is not only used for the model aggregation task, but also for supervised learning using a few amounts of labeled data. The experimental results demonstrate

that FLUIDS with limited labeled data can achieve competitive results as well as decrease the communication overhead.

1.4 Thesis Structure

The remainder of this thesis is organized as follows.

The first part of our thesis focuses on the background and state-of-the-art. Chapter 2 provides background on ML/DL models as well as FL concepts. In this chapter, we detail the reasons behind the use of each model as well as their strengths and weaknesses. In Chapter 3, we review the application of ML-based solutions for network traffic analysis. In particular, we study the application of ML and DL models for traffic classification and intrusion detection systems. Then, we outline the shortcomings of the proposed solutions such as generalization capability, lack of labeled data, as well as privacy preservation of the end-user data.

Then, in the second part of this thesis, we present our contributions. More specifically, in Chapter 4, we present a blending-based model in order to improve the generalization capability on the training set. However, before proposing the new ensemble, we evaluate the performance of different decision tree-based models. Finally, we demonstrate the effectiveness of the blending ensemble on both non-encrypted and encrypted traffic as well as its performance against state-of-the-art models.

Chapter 5 is devoted to the semi-supervised model for traffic classification. We demonstrate the importance of unlabeled data during the model training. Also, we study the proposed model complexity and classification performance compared to different machine learning models as well as state-of-art schemes.

In Chapter 6, we concentrate our focus on the intrusion detection systems, and we present an FL-based semi-supervised model, called FLUIDS. This model tries to take advantage of the unlabeled and labeled data for intrusion detection and attack classification while preserving privacy.

Finally, a summary of this thesis and some future directions are presented in Conclusion and Future Work 6.3.

1.5 Publications

The fruit of my Ph.D. and the research collaboration resulted in several international publications, which are listed below.

1.5.1 Journal Papers

1. **Ons Aouedi**, Kandaraj Piamrat, Benoît Parrein. "Ensemble-based Deep Learning model for network traffic classification", IEEE Transactions on Network and Service Management (TNSM), 2022 [10].

2. **Ons Aouedi**, Kandaraj Piamrat, Guillaume Muller, Kamal Singh. "Federated Semi-Supervised Learning for Attack Detection in Industrial Internet of Things", IEEE Transactions on Industrial Informatics, 2022 [11].

3. **Ons Aouedi**, Kandaraj Piamrat, Benoît Parrein. "Intelligent Traffic Management in Next-Generation Networks", MDPI Future Internet, 2022 [8].

4. Shaashwat Agrawal, Sagnik Sarkar, **Ons Aouedi**, Gokul Yenduri, Kandaraj Piamrat, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, Thippa Reddy Gadekallu. "Federated learning for intrusion detection system: Concepts, challenges and future directions", Elsevier Computer Communications, 2022 [12].

5. **Ons Aouedi**, Kandaraj Piamrat, Dhruvjyoti bagadthey. "Handling partially labeled network data: a semi-supervised approach using stacked sparse autoencoder", Elsevier Computer Networks, 2021 [13].

1.5.2 Conference and Workshop Papers

1. **Ons Aouedi**, Kandaraj Piamrat, Guillaume Muller, Kamal Singh. "Intrusion detection for Softwarized Networks with Semi-supervised Federated Learning", 2022, IEEE International Conference on Communications (ICC) May 2022, Seoul, South Korea - ***Best Paper Award*** [14].

2. **Ons Aouedi**, Kandaraj Piamrat, Benoît Parrein. "Decision tree-based blending method using deep-learning for network management", 2022, IEEE/IFIP Network Operations and Management Symposium (NOMS) April 2022, Budapest, Hungary [15].

3. **Ons Aouedi**, Kandaraj Piamrat, Benoît Parrein. "Performance evaluation of feature selection and tree-based algorithms for traffic classification", 2021 IEEE International Conference on Communications Workshops (ICC Workshops), - June 2021, Montreal, Canada [16].

4. **Ons Aouedi**, Kandaraj Piamrat, Dhruvjyoti Bagadthey. "A semi-supervised stacked autoencoder approach for network traffic classification", 2020 IEEE 28th International Conference on Network Protocols (ICNP Workshops) October 2020, Madrid, Spain [17].

Part I
Background & State of the Art

Part I consists of two main chapters in order to help understand the proposed contributions. The first Chapter will introduce the background and basic concepts. These concepts are shallow Machine Learning (ML) models, Deep Learning (DL), Federated Learning (FL), as well as dimensionality reduction techniques. Besides, a summary of several shallow ML and DL models' strengths and weaknesses has been presented. Since research in ML for network traffic analysis has been extensively studied in recent years and many schemes have been proposed, a survey of representative approaches will be given in the second Chapter of Part I. In this Chapter, a literature review on the application of such models for network traffic analysis will be given along with the motivation behind such applications. In particular, we will study ML/DL-based approaches for traffic classification and intrusion detection systems. Besides, we will highlight the research gaps encountered in such proposed approaches. Therefore, the reader will be introduced to the motivation behind our contributions.

Chapter 2

Machine Learning Introduction

Introduction

As discussed in the General Introduction, our objective is to propose a new ML/DL-based solution for network traffic analysis. Traffic analysis aims to understand the traffic carried on the Internet. Consequently, it has significance in a variety of network-related activities, from security monitoring (e.g. detecting malicious traffic) and QoS provisioning. It provides the operators with useful forecasts for long-term traffic management. Before beginning with a detailed presentation of our contribution this chapter will introduce the theoretical background of the basic concepts, which are useful to understand our contributions.

This chapter is divided into five main sections: Section 2.1 treats the concept of ML including its different types, supervised, unsupervised, semi-supervised, and reinforcement learning. Section 2.2 presents different shallow ML models. Then, Section 2.3 presents an overview of DL architecture and algorithms. section 2.4 deals with the motivation behind dimensionality reduction, and we will present a wide variety of feature selection and feature extraction methods. Next, section 2.5 presents the drawbacks and vulnerabilities of the conventional models training followed by a brief presentation of the main concept of FL in section 2.6.

2.1 What Machine learning is?

ML is a branch of Artificial Intelligence (AI) that attracts the attention of academia and industry like Google, Apple, Facebook, Netflix, and Amazon [18]. The effectiveness of ML has been validated in different application scenarios i.e. healthcare, autonomous driving, recommendation systems, network resources management, and network traffic analysis. The huge amount of data generated by IoT devices is behind the success of ML. It addresses the question of how to build a computer system that improves automatically through experience and data [19]. In particular, ML generally proceeds in two phases: (i) in the training phase, a collection of data, called *training data*, is used to build or improve a model by learning from the inherent structure and relationships within the data and (ii) this model is then applied to unseen data, called *test data*, to predict certain properties of these data. Based on these, ML can be defined as the technique that generalizes beyond the examples in the training set/environment and can be thought of as "programming by example". In other words, it is an intelligent technique used to automatically improve their performance through experience. Mitchell in [20] defined ML as *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance of tasks in T, as measured by P, improves with experience E"*. For example, in a learning system for playing a chess game, we will have: T: play chess, P: percentage of games won/lost, and E: Playing with itself/others. This learning aspect demonstrates the difference between ML

and conventional programming (Figure 2.1): ML schemes, find some "rules" from training data in order to be applied to unseen data and generate the desired output. The important product of this process is not the output, but the model (rules) that can be used to predict the output of new data (input). Unlike conventional programming where the programmer manually writes instructions (the rules) to generate the desired output from a given set of input variables according to this rules [21].

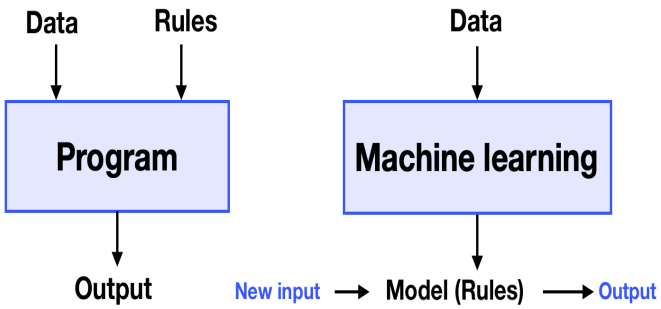


Figure 2.1: Traditional programming vs Machine Learning, inspired by [21]

ML-based models can be classified into four paradigms: Supervised learning, unsupervised learning, semi-supervised learning, and Reinforcement Learning (RL). These paradigms differ in the manner the algorithm is being trained.

2.1.1 Paradigms

2.1.1.1 Supervised learning

This type of learning process is the most commonly used. The supervised models operate when an observation needs to be assigned to a predefined class based on a number of observed features related to that observation [22]. As shown in Figure 2.2, the supervised learning models use the observations (x) and their labels (y) during the training process. They try to find model parameters that best predict the data based on a loss function $L(y, \hat{y})$. Here, y is the output variable, and $\hat{y}$ represents the output of the model obtained by feeding a data point x (input data) to the function that represents the model.

2.1.1.2 Unsupervised learning

With the rapid increases in the size and complexity of data, unsupervised learning will be the trend in the future. It tries to find the relationships between the inputs without having any prior knowledge of the outputs (Figure 2.3). In other words, the aim of unsupervised learning is to categorize the input data into distinctive clusters (i.e. groups) by examining the similarity between them. Each observation within the same cluster is having greater

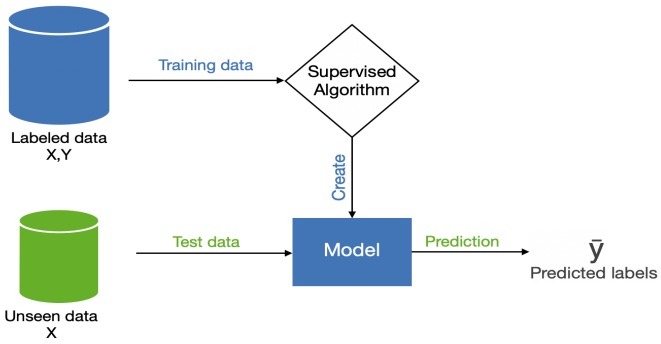


Figure 2.2: Training process of supervised learning algorithm

similarity as compared to the observation in other clusters [23]. K-means is one of the most used unsupervised learning methods.

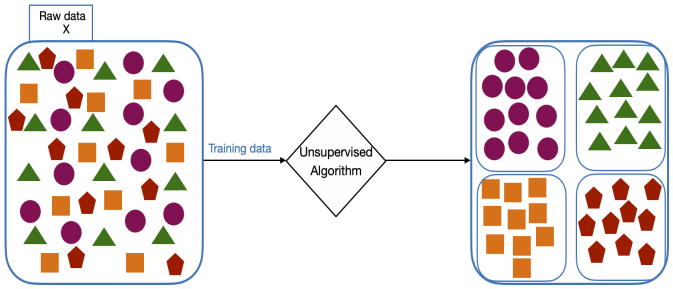


Figure 2.3: Training process of unsupervised learning algorithm

2.1.1.3 Semi-supervised learning

As the name implies, semi-supervised learning combines both supervised and unsupervised learning to get benefits from both approaches. It attempts to use unlabeled data as well as labeled data to train the model (Figure 2.4), contrary to supervised learning (data all labeled) and unsupervised learning (data all unlabeled). The labeled instances are difficult, require human effort, and are time-consuming to obtain especially for traffic classification. Semi-supervised learning tries to minimize these problems as it uses a few labeled examples with a large collection of unlabeled data [24]. It is an appropriate method when large amounts of unlabeled data are available as in the network traffic. That is why in the last few years there has been a growing interest in semi-supervised learning in the scientific community, especially for traffic classification.

2.1.1.4 Reinforcement learning

The main idea of RL was inspired by biological learning systems. It is different from supervised and unsupervised learning where instead of trying to find a pattern or learning from

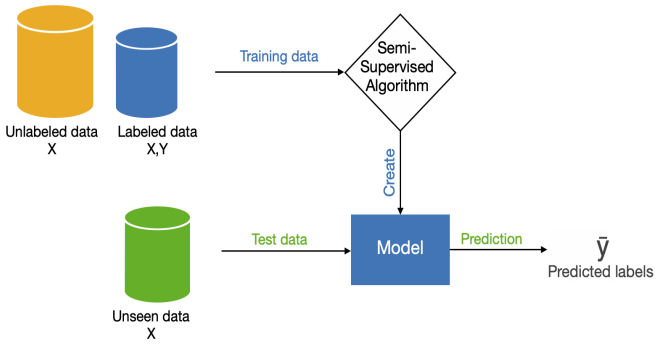


Figure 2.4: Training process of semi-supervised learning algorithm

a training set of labeled data, the only source of data for the RL is the feedback of the software agent received from its environment [25]. That is why it is considered as a fourth ML category, alongside supervised, unsupervised, and semi-supervised learning. In addition, the RL is defined by the provision of the training data by the environment. In other words, it is a technique that permits an agent to learn its behavior by interacting with its environment (Figure 2.5). There are three important elements that construct this learning approach, namely observations, reward, and action. Therefore, the software agent makes observations and executes actions within an environment, and in return it receives rewards. The agent's job is to maximize cumulative reward. The most known RL algorithm is Q-learning [26] and is widely used for network traffic routing [27]. Moreover, DL has been used to improve the performance of RL algorithms (i.e. allows the RL to be applied to larger problems). Therefore, the combination of DL and RL gives the so-called DRL. DRL began in 2013 with Google Deep Mind [28]. A good survey that presents RL and DRL approaches are available at [28].

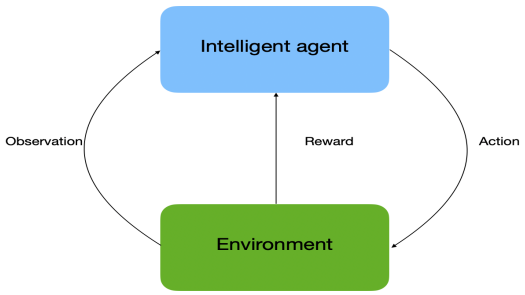


Figure 2.5: Conceptual diagram for RL system

2.1.2 Tasks

ML tasks are being made depending on the use case or the problem. Some of the well-known tasks, classification, regression, and clustering, are described below.

Table 2.1: Comparison of ML categories

Method	Strengths	Weaknesses
Supervised learning	Low computational cost, fast, scalable	Requires data labeling and data training
Unsupervised learning	Requires only the data samples, can detect unknown patterns, generates labeling data	Cannot give precise information
Semi-supervised learning	Learns from both labeled and unlabelled data	May lead to worse performance when we choose the wrong rate of unlabeled data
Reinforcement learning	Can be used to solve complex problems, efficient when the only way to collect information about the environment is to interact with it	Slow in terms of convergence

2.1.2.1 Classification

Classification is one of the most used tasks. It is based on supervised models and is used when the outputs take discrete values. Binary, multi-class, and multi-labeled are the three approaches of classification [5]. In binary classification, only two possible classes, for example, classify the traffic as "attack" or "normal". While multi-class classification implies that the input can be classified into only one class within a pool of classes such as classifying the traffic as "Chat", "Streaming", and "game". Multi-labeled classification allows the classification of an input sample into more than one class in the pool of classes like classifying the traffic as "Skype" and the traffic type as "Video".

2.1.2.2 Regression

Regression is also based on the supervised model that tries to map the data into a real-value variable (the outputs are continuous values). Regression can be used for example to forecast future load traffic based on historical data.

2.1.2.3 Clustering

It is an unsupervised ML task used to categorize the input data into distinctive groups through observable features. For example, clustering can be used to assign labels for unlabeled data that has similar behavior [29].

2.1.3 Evaluation metrics

After a model is trained, it must be tested to verify its performance. Therefore, the main purpose of the evaluation is to quantify the model performance and compare it against others. Thus, to evaluate the performances of the ML/DL models, various statistical measures are used. The standard evaluation metric is the *accuracy*. The accuracy is defined by

$$Accuracy = \frac{N_c}{N_t}. \tag{2.1}$$

where N_c denotes the amount of test data correctly assigned to the groups to which they belong, and N_t is the total amount of test data. However, accuracy is not enough metric when we have imbalanced data [30] as the accuracy is biased to the majority class, regardless of the minority class (with lower samples), which obtained poor performance [31] [22]. Further, for extremely skewed class distributions, the *recall* of the minority class is often 0, which means that there are no classification rules generated for the minority class [32]. For this reason, the evaluation of the classifications' performance must be carried out using specific metrics to calculate each class separately and yield a deeper understanding of the classifier's performance than a simple accuracy metric. The most used metrics are *recall*, *precision*, and F1-score [32]. To calculate these metrics, there are four important terms:

- TP (True Positive): Predicted to be positive and the actual value is positive.
- FP (False Positive): Predicted to be positive, but the actual value is negative.
- TN (True Negative): Predicted to be negative and the actual value is negative.
- FN (False Negative): Predicted to be negative but the actual value is positive.

Details of the metric calculation are given below.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

$F1 - score$ or *F-measure* is the harmonic mean of precision and recall which is used to integrate these two metrics into a single metric. Thus, if its value is high, the performance of classification is better.

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.4}$$

These metrics are widely used for model evaluation whether DL or shallow ML models. The term shallow model indicates any models different from the DL. The success of the shallow ML models generally depends on the features to which they are applied.

2.2 Shallow models

Here we present the most used shallow models such as Decision Tree (DT), Support Vector Machine (SVM), K-means, and ensemble-based models.

2.2.1 DT (Decision Tree)

The decision tree is a tree-like structure, where every leaf (terminal) corresponds to a class label and each internal node corresponds to an attribute. The node at the top of the tree is called the root node. Tree splitting uses Gini Index or Information Gain methods [33].

2.2.2 SVM (Support Vector Machine)

SVM is a powerful classification algorithm, which can be used for both regression and classification problems. But, it is mostly used as a classification technique, it was initially developed for binary classification, but it could be extended for multiclass problems. It can be a linear and non-linear classifier by creating a splitting hyperplane in the original input space to separate the data points. Kernel functions are used for non-linear mapping of training samples to high dimensional space such as polynomial, Gaussian, and sigmoïd.

2.2.3 KNN (K-Nearest Neighbour)

KNN is a supervised model reason with the underlying principle "*tell me who are your friends, I will tell you who are you*" [34]. It classifies new instances using the information provided by the k nearest neighbors so that the assigned class will be the most common among them (majority vote). It stores the entire training data using distance function [35] and the most used one is the Euclidean distance. Also, as it does not build a model, thus it is considered a lazy approach.

2.2.4 K-means

K-means is the oldest and most popular partitioning method. It aims to partition the data into K clusters based on a similarity measure. In other words, the examples belonging to the

same cluster have high similarity as compared to those of other clusters [36]. Each cluster has a category center μ_k and the Euclidean metric is selected as the criterion of similarity. The sum of squares of the distance between the points in each cluster to the center of the cluster μ_k is calculated to minimize the sum of squares within each cluster. The objective function is written as:

$$V = \sum_{i=1}^{K} \sum_{n=1}^{N} ||x_i - \mu_k||^2$$

where x is the sample to be clustered, N is the number of samples and K is the number of clusters. K points are randomly selected from the datasets as the initial clustering center. The Euclidean distance from each sample to the cluster center is calculated. The sample is returned to the nearest cluster center. The new clustering center is obtained by calculating the average value of the newly formed data objects of each cluster. If there is no change in two successive iterations, it shows that the sample adjustment is over and the clustering criterion function has been converged.

2.2.5 Ensemble learning

Recently, ensemble learning considered one of the promising directions due to its superiority in ML. It combines the output of several models (by weighted or unweighted voting). Ensemble methodology imitates our nature to seek several opinions before making any decision where we combine a weighed various individual opinions to find a final decision [37]. The main objective of ensemble learning is to combine heterogeneous or homogeneous models (commonly classifiers) in order to obtain a model that overcomes the single/simple model limitations (e.g. over/underfitting) [38]. Independently of the ensemble learning training process, a recent analysis demonstrates that the ensemble models can outperform the simple model and reliable decisions. The performance of the ensemble models comes from the diversity of the classifiers' strengths [39]. Consequently, there exist several techniques for ensemble model construction where bagging, boosting, and blending are the most popular ones. Similar to single models, there are no best ensemble methods. However, some methods work better than others in certain conditions (e.g. data, features).

2.2.5.1 Bagging

Bagging called bootstrap aggregating is one of the earliest ensemble techniques [40]. It is a parallel ensemble that tries to decrease the variance (i.e. overfitting). The Bagging process consists of three steps (i) sub-sampling the training set in a random way to obtain the sub-training sets, (ii) using these sub-training sets, it trains several weak models independently, and (iii) combining the outcome of these models by voting technique. Voting is generally used for classification problems. It can be weighted or not. In non-weighted voting, all the classifiers have equal weight and the predicted class is the one that has the majority vote. In

the weighted cases, different weights can be assigned to the used classifiers. Random Forest (RF) is one of the most popular bagging-based models developed by Breiman nearly 20 years ago [41]. It is an ensemble of independent decision trees in which the base trees train not just on a randomly chosen subset of the data, but also on a randomly chosen subset of the input features. Then, the outputs of the base classifiers are combined with the majority voting technique. The main advantage of the RF algorithm, compared to the gradient boosting algorithms, is that it requires fewer hyper-parameter tuning [42].

2.2.5.2 Boosting

Boosting is a sequential ensemble used to improve the performance of the decision tree [43]. It combines the models in a sequential manner and each model reduces the error of the previous ones. AdaBoost is the first boosting algorithm developed by Freund and Schapire [43]. It does not randomly select training samples like RF but focuses on the samples that do not have accurate predictions (misclassified samples). In other words, after training the model, AdaBoost increases the weight of the misclassified samples. Therefore, AdaBoost obtains different training sets by focusing on the instances that are misclassified by the previously trained classifiers.

XGBoost is one of the most efficient implementations of gradient-boosted decision trees and it is developed by Chen and Guestrin in 2014 [20]. It has been selected as one of the best ML algorithms used in Kaggle competitions due to its advantages such as easy parallelism and use as well as its high prediction accuracy. This algorithm learns from the error of prior trees to improve the accuracy in subsequent iterations. Instead of increasing the instance weights at every iteration as AdaBoost does, XGBoost tries to fit the new model according to residual errors made by the previous model.

Due to the success of the boosting-based models, other models have been proposed like CatBoost and LightGBM. CatBoost is a gradient-boosting algorithm that was developed by the Russian tech company Yandex in mid-2017 [44]. It is the best solution for heterogeneous data (i.e. categorical and numerical data). Other ML models require pre-processing steps to convert categorical data into numerical data, whereas CatBoost requires only the indices of categorical features. Thus, it performs one-hot encoding to transform the categorical data into numerical data. On the other hand, LightGBM is one of the most recent boosting models. Since conventional implementations of Gradient-Boosting Decision Tree (GBDT) may be inefficient when the number of instances is large or the dimension of the feature is high. For this reason, it is a highly efficient gradient-boosting decision tree proposed by Microsoft in 2017 [45]. It uses gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB) algorithms in order to reduce the number of examples and the number of features. In fact, EFB uses a histogram algorithm to bucket continuous feature values into discrete bins, which fastens the training procedure and results in lower memory

usage. On the other hand, GOSS helps the model to retain instances (i.e. examples) with large gradients (i.e. under-training) while performing random sampling on instances with small gradients (i.e. small training errors).

2.2.5.3 Blending

Blending ensemble is very close to stacking, which is originally introduced in the Netflix competition [46]. But, unlike stacking, blending uses only a holdout (validation set) set from the train set to make predictions. It is simpler and works better than stacking ensemble [47]. It consists of two levels, which are base-classifiers used in level-1 and meta-classifier used in level-2. The base classifiers are used to provide base predictions as new features. Then the meta-classifier is trained on these new features to give the final decision. Thus, the blending can collectively estimate the errors of all base classifiers through basic learning steps and use a meta-classifier to reduce the prediction residuals. All of this made the blending leads to greater awareness and familiarity with a dataset [48]. A general overview of the training and testing process is shown in Figure 2.6. For more details, the blending ensemble is based on several steps as follows:

1. uses a hold-out method to divide the training set into a new training set and validation set;

2. trains the base classifiers through the new training dataset, and the prediction of the base classifiers on the validation set forms the meta-training dataset (level-1);

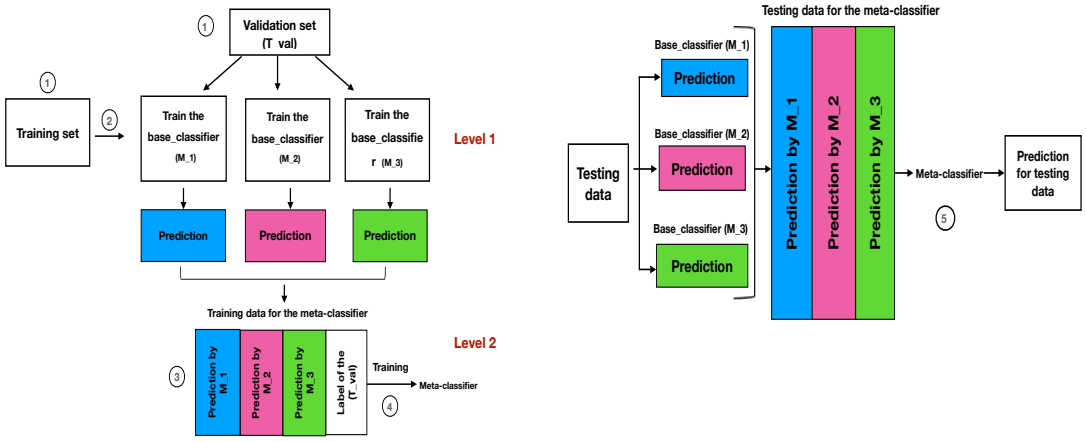
3. joins the prediction on the validation set of the base classifiers to form the meta-training dataset (Figure 2.6a);

4. trains the meta-classifier using the meta-training dataset (level-2);

5. uses the meta-classifier to make the final prediction through the test set (Figure 2.6b).

2.2.6 Summary

In summary, as shown in Table 2.1 the use of learning categories, i.e. supervised semi-supervised, unsupervised, and reinforcement learning is depending on the context and the available data. Also, based on those categories several shallow models have been proposed and most of the well-known shallow ML models are DT, RF, SVM, KNN, Boosting-based models, and K-means. It can be seen from Table 2.2, that each model has its weakness and strengths, as well as the majority of the model, yielded performances that are quite competitive with each other. It is hence left to the user to adopt an appropriate algorithm to their requirement and environment.



(a) Training process (b) Test process

Figure 2.6: Blending ensemble process

Although the performance of the shallow ML models, they are highly dependent on the amount and the quality of features. Furthermore, extracting a large number of features from the coming flow can be time-consuming and in turn, decrease the QoS of the system [49]. To tackle these issues, dimensionality reduction, including feature selection and feature extraction has been used as a pre-processing task for the shallow ML models.

2.3 Deep Learning

DL is a branch of ML that evolved from Neural Networks (NNs), which benefits from training data augmentation. It has achieved great success in many applications in comparison to shallow ML models, simply because computers acquire the computational power to build complex models that can actually process and learn from big data. The major benefits of DL over shallow ML models are its superior performance for large datasets and the integration of feature learning and model training in one architecture [50]. In particular, building models using traditional ML is bottlenecked by the amount of features engineering required since there are limits to how much human effort can be thrown at the problem. In contrast, DL algorithms hierarchically extract knowledge from the training data through multiple layers of nonlinear processing, in order to make them flexible in modeling complex relationships [22]. On the other hand, it helps to avoid human intervention and time-wasting as maximum as possible as illustrated in Figure 2.7.

DL has revolutionized various domains, such as IoT, transportation systems, and healthcare due to its ability to exceed human accuracy as well as shallow ML models. Figure 2.8 presents the search trend of four popular shallow ML models and DL from all the countries

Table 2.2: Comparison of Shallow ML models

Method	Strengths	Weaknesses
Decision Tree (DT)	Simple to understand and interpret, requires little data preparation, handles many types of data (numeric, categorical), processes easily data with high dimension	Generates complex trees with numeric data, requires large storage
Random Forest (RF)	Efficient against under-fitting	Requires large training dataset, impractical for real-time application
Support Vector Machine (SVM)	Scalable, handles complex data	Computationally expensive, there is no theorem to select the right kernel function
K-Nearest Neighbour (KNN)	Easy to implement, has good performance with a simple problem, non-expert users can use it efficiently	Requires large storage space, Determining the optimal value of K is time-consuming, K values vary depending on the dataset, testing is slow, When the training dataset is large, the computation is very time-consuming, it is not suitable for real-time classification
Boosting algorithms	High accuracy, efficient against under-fitting	Computationally expensive, hard to find the optimal hyper-parameters
K-Means	Fast, simple and less complex	Requires the number of clusters in advance, can not handle the outliers

between 01/01/2008 and 16/04/2021. It represents Google search statistics based on queries that people entered into the Google search engine. We can observe a drastic increase in interest in DL.

As DL is based on artificial neural networks (ANNs), we will start by looking at the structure of a neural network. ANNs are a computing system, inspired by the structure of the brain, which is based on a set of interconnected neurons as shown in Figure 2.9. ANNs contain very few hidden layers (i.e. with one hidden layer) while DL contains many more layers (deep). Each hidden layer comprises a set of learning units called neurons. These neurons are organized in successive layers and every layer takes as input the output produced by the previous layer, except for the first layer, which consumes the input. Besides,

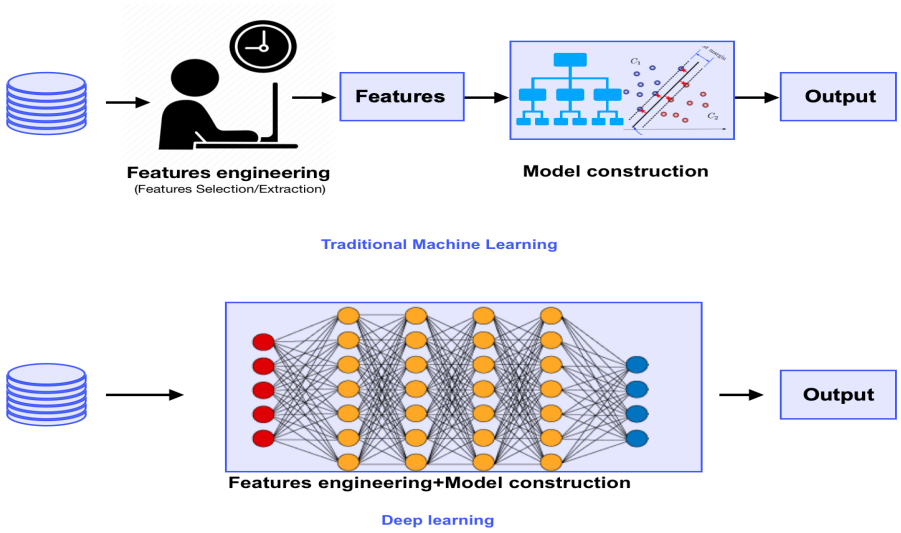


Figure 2.7: Difference between traditional Machine Learning and Deep Learning

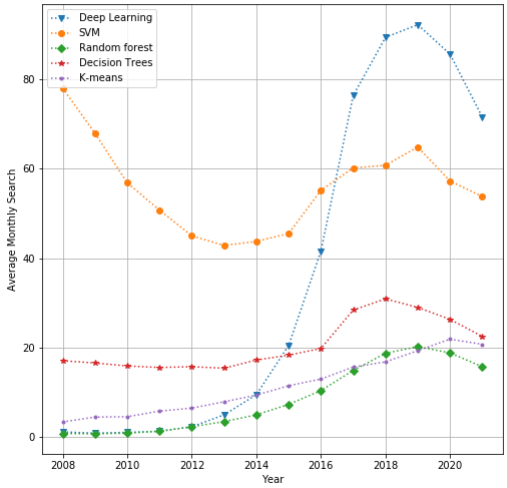


Figure 2.8: Google Trend showing more attention toward DL in recent years

the neurons of a hidden layer fully connect with those of the previous layer. There are three kinds of layers in all ANNs:

- *Input layer* stores the input data, each neuron stores a x_i component of the observing x.
- *Hidden layers* are placed between input and output layers. It has the ability to process the data obtained by the input layer and transfer it to the output layer. The hidden layer can be more than one layer.
- *Output layer* is the last layer in the network. The output of this layer is the output of the model and the prediction is generated based on the input.

In addition, the neuron receives a vector x as input and uses it to compute an output signal, which is transferred to other neurons. It is parameterized through $\{W, b\}$ where W is a weight vector, b is the bias, and f is referred to as an activation function as shown in Figure 2.9. The output of each neuron can be described as

$$f\left(\sum_{i=0}^{n} x_i \cdot w_i + b\right)$$

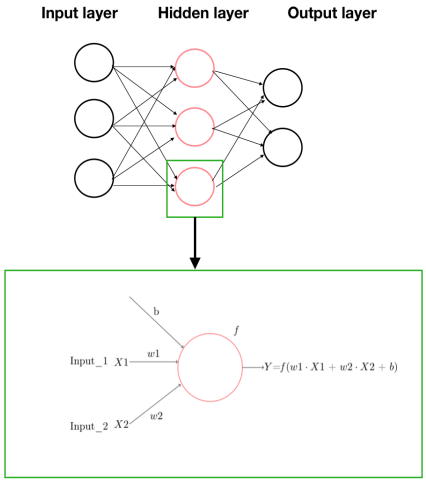


Figure 2.9: Structure of shallow Neural Network and its neuron

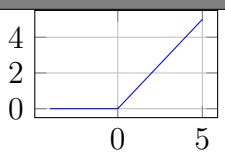
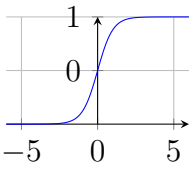
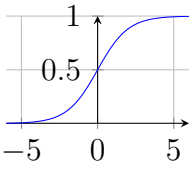
In other words, it aims to add the non-linearity into the model as well as keep the output of the neuron restricted to a certain limit and hence normalize the output. The non-linearity enables them to learn complex patterns and avoid the constraints associated with linear functions. Moreover, the choice of activation function can affect network training time [51]. The most frequently used activation functions are the ReLU (Rectified Linear Unit), sigmoïd, and hyperbolic tangent functions (TanH). The functional graph of the three non-linear activation functions is presented in Table 2.3. In fact, networks with ReLU show better convergence performance than sigmoïd and TanH [51].

The growing popularity of DL inspired several companies and open-source initiatives to develop powerful DL open-source libraries and frameworks that can be used to avoid building models from scratch [52]. The availability of such libraries and frameworks causes rapid diffusion within the research community and in our daily life.

2.3.1 Types of DL-based models

Due to the success of DL, several models have been proposed. In this subsection, we introduce the key principles underpinning these models and discuss their strengths and weaknesses.

Table 2.3: The different commonly used activation functions

Activati func-tion	Functional Graph	Values in the range	Mathematical Background
ReLU		(0,∞)	$f(x) = max(0, x)$
Tanh		(-1,1)	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Sigmoïd		(0,1)	$f(x) = \frac{1}{1 + e^{-x}}$

- **MLP (Multilayer perceptron)**

MLP is a class of feedforward ANN, which consists of three or more layers [53]. The first layer is for input data. One or more hidden layers extract features from the input. The last layer outputs the classification result. Each hidden layer is composed of multiple neurons that use a nonlinear activation function. MLP is mostly used as a baseline.

- **AE (AutoEncoder)**

  AE is one of the several ANNs-based architectures with a symmetrical structure. It is an unsupervised feature learning neural network that can extract features from unlabeled data automatically [54]. During the process, the AE tries to minimize the

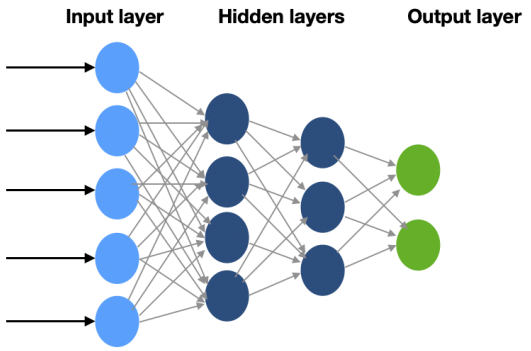


Figure 2.10: MLP (MultiLayer Perceptron)

reconstruction error, and the corresponding code is the learned feature. AE consists of two core segments placed back-to-back that have the same number of layers as shown in Figure 2.11.

1. Encoder: takes input data and maps it to hidden representation (code), this hidden layer has less dimension than the input data, and the encoder reduces initial data;
2. Decoder: uses the hidden representation (code) to reconstruct the input.

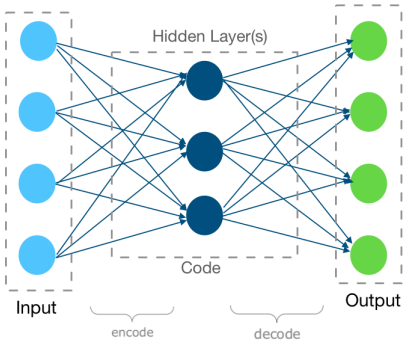


Figure 2.11: General AutoEncoder (AE) process

- **CNN (Convolution Neural Networks)**

As shown in Figure 2.12, CNN is one of the Neural Networks types, which consists of a number of convolution and pooling (subsampling) layers followed by fully connected layers [55]. Pooling and convolution layers are used to reduce the dimensions of features and find useful patterns. Next, fully connected layers are used for classification. CNN-based models achieve remarkable performance in Computer Vision and Image Processing related fields.

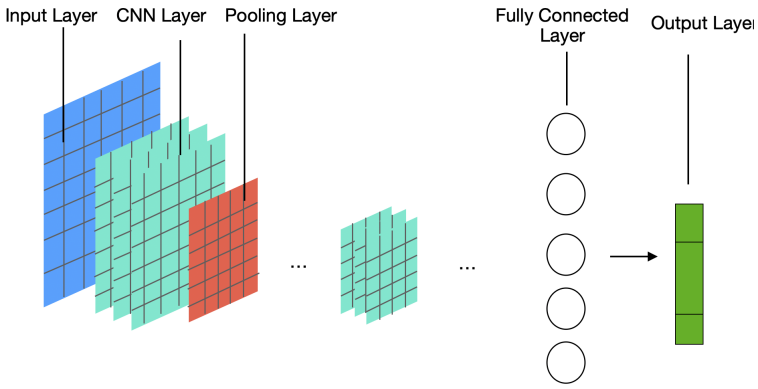


Figure 2.12: CNN (Convolution Neural Networks)

- **RNN (Recurrent Neural Network)**

RNN is a neural network that has one or more connections between neurons that form cycles (Figure 2.13). These cycles are responsible for storing and passing the feedback of one neuron to another, creating an internal memory that facilitates the learning of sequential data (X_0, X_1, ..., X_t) and their hidden state (h_0, h_1, ..., h_t). In other words, in RNN the decision made at time $t-1$ affects the decision at time t.

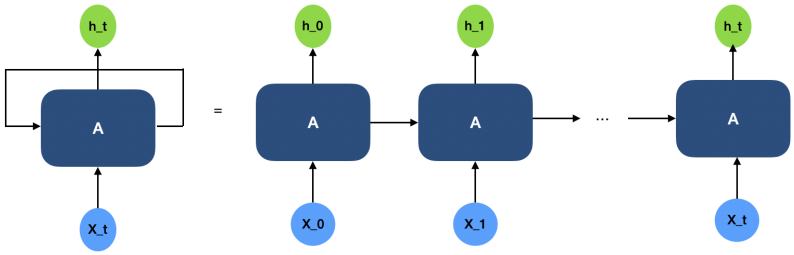


Figure 2.13: RNN (Recurrent Neural Networks)

- **LSTM (Long Short-Term Memory)**

LSTM model is an extension of RNNs, it was created as the solution to short-term memory [56]. As shown in Figure 2.14, the model has internal mechanisms called gates (forget gate, input gate, and output gate) that can learn which data in a sequence is important to keep or to throw away. The purpose of the gates is as follows:

- input gate: controls whether the input is passed on to the memory cell or ignored;
- output gate: controls whether the current activation vector of the memory cell is passed on to the output layer or not;

- forgets gate: controls whether the activation vector of the memory cell is reset to zero or maintained.

Consequently, the LSTM helps to choose which information is relevant to remember or forget during sequence processing.

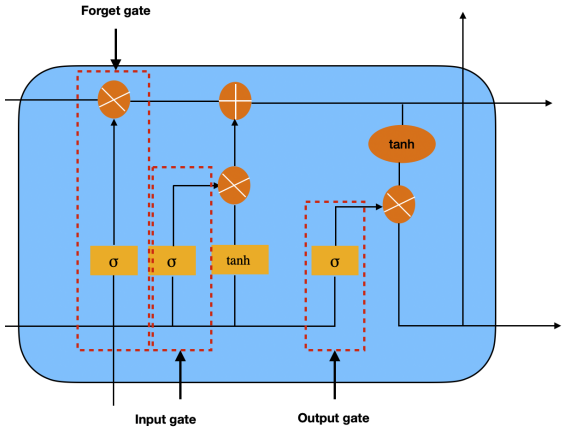


Figure 2.14: LSTM (Long Short-Term Memory)

- **GRU (Gated Recurrent Unit)**

The GRU was proposed in 2014 and is similar to LSTM but has fewer parameters. It works well with sequential data as does LSTM. But, unlike LSTM, GRU has two gates (Figure 2.15), which are the update gate and reset gate, and hence it is less complex [57]. The reset gate is similar to forget gates of LSTM, whereas the update gate is similar to the combination of an input gate and an output gate of the LSTM model.

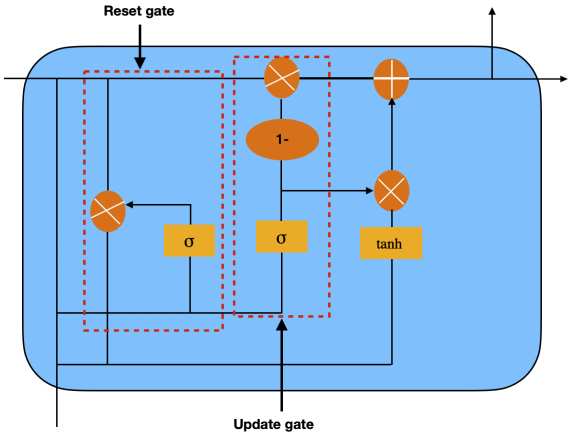


Figure 2.15: GRU (Gated Recurrent Unit)

2.3.2 Summary

DL refers to the idea of deep structured learning, hierarchical learning, and successive layers of representation. It has shown exemplary performance in several academic and industry domains, due to its unique nature for solving complex problems. Thus, different architectures have been proposed and used over the past few years. Table 2.4 summarizes the strengths/weaknesses of the well-known DL-based architectures.

Table 2.4: Summary of different deep-learning models [58] [59]

.

Models	**Strengths**	**Weaknesses**
MLP	Easy to implement.	Modest performance, slow convergence, occupies a large amount of memory.
AutoEncoder	Works with big and unlabeled datasets, suitable for feature extraction and used instead of manually engineered extraction.	The quality of features depends on the model architecture and its hyper-parameters, it is hard to find the code layer size.
CNN	Weights sharing, extracts relevant features and provides highly competitive performance.	High computational cost, requires a large training dataset and a high number of hyper-parameters tuning to achieve optimal features.
RNN	Simple to implement, faster than LSTM and GRU, ability to capture temporal behaviors.	When modeling long sequences, its ability to remember what was learned before many time steps may decline.
LSTM	Good for sequential information and works well with long sequences.	High model complexity, high computational cost.
GRU	Computationally more efficient than LSTM.	Less efficient in accuracy than LSTM.

2.4 Dimensionality reduction

As the amount of high-dimensional data has increased in recent years, data-driven methods become significantly harder. The network dataset may contain irrelevant (a feature that provides no useful information) or redundant (a feature whose predictive ability is covered

by another) features, which increase search space and decrease the model performance [24]. With the increase in the dimensionality (i.e. features) of data, the predictive ability of a model decreases as well as causes an extra computational cost for both storage and processing [60]. These challenges are referred to as *curse of dimensionality*, which is one of the most challenging in shallow ML models, first introduced by Bellman in 1961 to indicate that such models can work fine in low dimensions and become intractable with high-dimensional data [18].

Shallow ML-based models are only as good as the given features. Also, the learning ability of the model depends not only on the quality of collected features but also on the number of features considered. The model will perform poorly if the amount of the features is larger than the number of observations (over-fitting) [61]. In this situation, the model can easily separate the training data, but it fails on the unseen data (test data). All these made dimensionality reductions a critical task not only because of the higher size of the input dataset but because it needs to meet two challenges, which are: (i) the maximization of the learning capacity and (ii) the reduction of the number of features. Therefore, in a dataset with a high number of features, the data reduction process is a must in order to produce accurate models at a reasonable time [16].

Dimensionality reduction can be divided into feature selection (i.e. feature elimination) and feature extraction. The main difference is that feature selection methods select a subset from the original features while feature extraction methods create new features set from the original features. Table 4.5 presents the advantages and disadvantages of each approach.

Table 2.5: Comparison of dimensionality reduction techniques [62] [63].

Method	**Advantages**	**Disadvantages**
Filter	Low computational cost, fast, scalable	Ignores the interaction with the classifier
Wrapper	Competitive classification accuracy, interaction with the classifier	Slow, expensive for large feature space
Feature Extraction	Reduces dimension without loss of information	No information about the original features

2.4.1 Feature selection

Feature selection methods have become an indispensable component of the shallow ML models (e.g. SVM, RF, etc) [64] and one of the simplest ways to reduce data size. They

try to pick a subset of features that "optimally" characterize the target variable. Feature selection is the process of selecting the best features in a given initial set of features that yield a better classification performance [65], regression as well as finding clusters efficiencies. It helps to identify the relevant features (contribute to the identification of the output) and discard the irrelevant ones from the dataset, for the purpose of performing a more focused and faster analysis. The feature selection process consists of four basic steps as follows:

1. *subset generation:* is a search procedure that generates candidate feature subsets for evaluation based on a search strategy (i.e. start with no feature or with all features).

2. *evaluation of subset:* tries to measure the discriminating ability of a feature or a subset to distinguish the target variables.

3. *stopping criteria:* determines when the feature selection process should stop (i.e. addition or deletion of any feature does not produce a better subset).

4. *result validation:* tries to test the validity of the selected features.

Feature selection methods can be distinguished into two broad categories, which are *filters* and *wrappers*.

2.4.1.1 Wrapper Methods

Wrapper Methods require a learning algorithm to use its performance as the evaluation criterion (i.g. classifier accuracy). It calculates the estimated accuracy of a single learning algorithm through a search procedure in the space of possible features, in order to find the best ones. The search can be done with various strategies like forwarding direction (the search begins with an empty set and successively add the most relevant features) and backward direction (starting with the full set and successively deleting less relevant features) also known as Recursive Feature Elimination (RFE). Starting from all the feature sets, RFE recursively removes features in order to maximize accuracy. Then it ranks the features based on the order of their elimination. Wrapper methods are also more computationally expensive than filter methods and feature extraction methods, but they produce feature subsets with very competitive classification accuracy [35].

2.4.1.2 Filter Methods

Filter Methods find the best features set by using some independent criteria (i.e. Information measures) before applying any classification algorithm. Due to the computational efficiency, the filter methods are used to select features from high-dimensional data sets. Also, it is categorized as a binary or continuous feature selection method depending on the data type. For example, Information Gain (IG) [66] can handle both binary and nominal data, but the Pearson correlation coefficient can handle only continuous data.

- **IG (Information Gain)**

IG [66] is one of the most used univariate feature selection methods [67]. The main concept of this approach is to rank subsets of attributes by calculating the IG entropy for each attribute in decreasing order. Each attribute gains a score from 1 (most relevant) to 0 (least relevant). The entropy of Y (target variable) is:

$$H(Y) = -\sum_{y \in Y} p(y) \log_2(p(y)) \tag{2.5}$$

Then, IG measures the mutual information provided by X on Y.

$$IG = H(Y) - H(Y/X) = H(X) - H(X/Y) \tag{2.6}$$

- **CFS (Correlation-based Feature Selection)**

CFS is a multivariate filter algorithm. A correlation measure is applied to evaluate the goodness of feature subsets based on the hypothesis that *"Good feature subset contains feature highly correlated with the class, yet uncorrelated with each other"* [68]. Therefore, the correlation coefficients are used to estimate the correlation between different features and the output variables and the inter-correlations between the features. In fact, CFS is usually used to remove the redundant features and hence reduce the over-fitting of the classifiers and decrease the complexity of computation. It uses the correlation coefficient that indicates the linear correlation between two random features (i.e. variables) f and t. The correlation coefficient can be summarized as:

$$cor = \frac{\sum_{i=1}^{N}(f_i - \mu_f)(t_i - \mu_t)}{\sqrt{\sum_{i=1}^{N}(f_i - \mu_f)^2 \sum_{i=1}^{N}(t_i - \mu_t)^2}} \tag{2.7}$$

Where N is the total values of each feature, and μ_f, μ_t is the mean for features f and t, respectively. The values of correlation cor obtained according to formula (2.7); range from -1 to 1. Usually, if $|cor| > 0.5$, this means that the two features have a strong correlation; if $|cor|$ is close to 0, means that there is no linear correlation between the features.

2.4.2 Feature extraction

Feature extraction performs a transformation of the original variables to generate other features using the mapping function F that preserves most of the relevant information. This transformation can be achieved by a linear or non-linear combination of original features. For example, for n features $f_1, f_2, f_3, ..., f_n$ we extract new features set $f'_1, f'_2, f'_3, ..., f'_m$ where $m < n$ and $f'_i = F(f_1, f_2, f_3, ..., f_n)$. With the features extraction technique, the feature space can often be decreased without losing a lot of information on the original attribute space.

However, one of its limits is that the information about how the initial features contribute is often lost [69]. Moreover, it is difficult to find a relation between the original features and the new features. Therefore, the analysis of the new features is almost impossible since no physical meaning of the transformed features is obtained from feature extraction techniques. We discuss some of the most frequently used feature extraction methods such as Principal Component Analysis (PCA) and AutoEncoder (AE).

2.4.2.1 PCA (Principal Component Analysis)

PCA is the oldest technique of multivariate analysis and was introduced by Karl Pearson in 1901. It is an unsupervised (it does not take into account the target variable) that reduces the dimensionality of data from f to $p < f$, by transforming the initial features space into a smaller space. The purpose of PCA is to extract new features called principal components (PCs) (less or equal to the initial features,) which are the linear combinations of the original attributes, orthogonal to each other, and capture the maximum amount of variation in the data. In other words, it tries to compress the initial features by identifying the strongest patterns in the data. Therefore, PCA can achieve dimensionality reduction with minimum noise than the original features. According to [70], the main idea of PCA is each pair of PCs has co-variance 0 (which means no redundant features), the PCs are ordered descendingly according to their variance, the first PCs capture as much of the variance of the data as possible (PC_1 has the highest variance, and PC_p has the lowest variance). However, PCA has some limitations below:

- We do not know how many PCs should be retained (the optimal number of PCs).
- It does not consider the correlation between target outputs and input features [71].

2.4.2.2 AE (AutoEncoder)

It is frequently used for the purpose of learning discriminative features of original data. Hence, AE is potentially important for feature extraction and many researchers use it to generate reduced feature sets (code). AE consists of two core segments placed back to back that have the same number of layers as shown in Figure 2.11.

This structure is formulated as below where equation 2.8 presents the encoder and 2.9 the decoder respectively:

$$Z = f(W_1 X + b_1) \tag{2.8}$$

$$X' = f(W_2 Z + b_2) \tag{2.9}$$

where $X = (x_1, x_2, ..., x_n)$ is the input vector, and $Z = (z_1, z_2, ..., z_m)$ is the vector extracted from the input X, known as code, $X' = (x'_1, x'_2, ..., x'_n)$ is the output reconstruction of the input X, where n is the dimension of the input vector and m is the number of code units. W_1 and b_1 are the weight matrix and bias between the input layer and the second layer (i.e. code). W_2 and b_2 are the weight matrix and bias between the second and the output layer; the function $f(.)$ is the activation function.

The difference between X and X' is usually called the reconstruction error (RE), which is represented in the form of a cost function that the model tries to reduce during the training process. The cost function of the AE is computed using Equation 2.10, where the parameter set is denoted by $\theta = \{W_1, b_1, W_2, b_2\}$.

$$J(\theta) = \sum_{i=1}^{n} RE(x_i, x'_i) \tag{2.10}$$

In many cases, Deep AE outperforms conventional feature extraction methods like PCA [72, 73] since it consists of several layers with nonlinear activation functions to extract features intelligently. On the other hand, AE has different variants and its structures are dependent on the number of layers [74]. The simple AE model has just one hidden layer, which is not able to get the discriminative features. Thus, to obtain a better performance and learn more complex and abstract features than classical AE, a more complex architecture and training procedure, known as Stacked AE (SAE) [75], has been proposed. With SAE, several AE layers are stacked together and form an unsupervised pre-training stage where the encoder layer computed by an AE will be used as the input to its next AE layer. Each layer in this stage is trained like an AE by minimizing its reconstructing error. When all the layers are pre-trained, the network goes into the supervised fine-tuning stage. At the supervised fine-tuning stage, a Softmax layer can be added to the encoding layer of the unsupervised pre-training stage for the classification task and discarding the decoding layers of SAE (Figure 5.2).

- **Denoising AE**

To further improve the robustness of feature representation extracted by the SAE, denoising hyper-parameters have been used [76]. Denoising AE is trained to reconstruct a clean input from a corrupted version of it (Figure 2.17). Therefore, similar to the conventional AE network, Denoising AE is trained in order to learn a hidden representation and to reconstruct its input. However, the main difference with denoising AE is that the model should reconstruct the original input from a corrupted version in order to force even very large hidden layers to extract more relevant features. This corruption of the data is done by first corrupting the initial input X to get a partially destroyed version X'. The input can be corrupted in many ways. In our contribution, we set a certain percentage of random units of each sparse AE (neurons) to zero (i.e. a fraction of the input is deleted randomly). To train

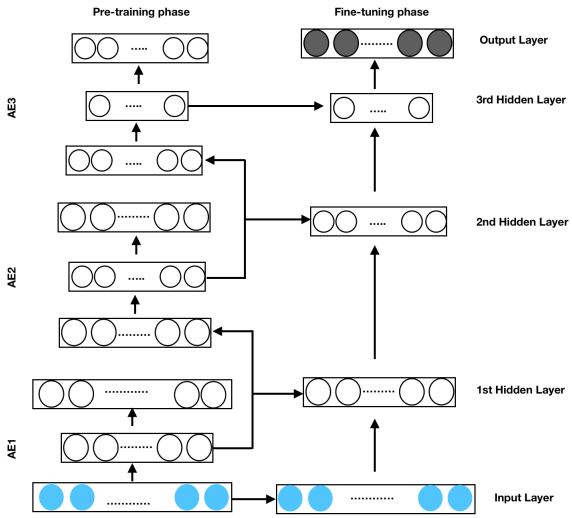


Figure 2.16: General Stacked AE process

a stacked denoising AE, each denoising AE is pre-trained independently. By doing so, the definition of good representation is changed into the following: *"a good representation is one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input"* [76].

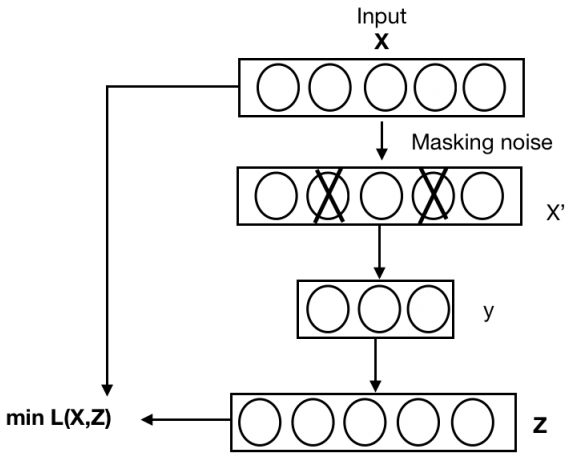


Figure 2.17: Training process of individual denoising autoencoders

2.5 ML/DL models limitations

Although DL/ML models have shown remarkable performance, they are associated with drawbacks and vulnerabilities such as over-fitting, privacy, and lack of training data.

2.5.1 Over or underfitting issue

To judge the performance of the models, the generalization capability is one of the most used evaluation metrics. To generalize well the model needs to prevent the problem of variance and bias. Variance defines the consistency of a learner's ability to predict random things (over-fitting), and bias describes the ability of a learner to learn the wrong thing (under-fitting) [18].

A model with the lowest bias, however, is not necessarily the optimal solution, because the ability to generalize from training data is also assessed by a second parameter termed variance. A major challenge for ML/DL models is to optimize the trade-off between bias and variance. To tackle this issue, ensemble learning might be used as well as adding or fine-tuning some hyper-parameters [13, 15]. For example, to improve the robustness of extracted features and prevent the over-fitting problem during the training process of DL models, dropout and/or denoising and/or sparse hyper-parameter can be introduced into the model.

- **Dropout**

Dropout is a technique applied in the training phase to reduce over-fitting effects and hence help the neural network model to learn more robust features and reduces the interdependent learning among the neurons [77]. The term "dropout" refers to dropping out units in a neural network (as shown in Figure 2.18). Technically, the "dropout" can be realized by setting the output $a = f(WX + b)$ of some hidden neurons to zero so that these neurons will not be involved in the forward propagation training process. By dropping a unit (i.e. neuron) out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections, and the choice of which units to drop is random. Consequently, random dropout makes it possible to train a huge number of different networks in a reasonable time [78].

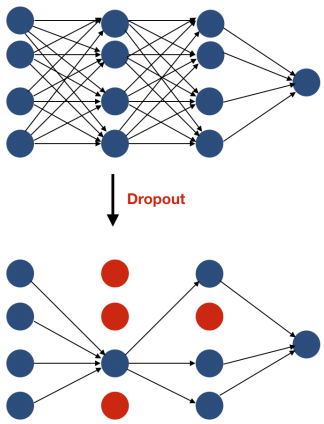


Figure 2.18: Dropout technique

- **Sparse hyper-paramters**

Imposing a sparse constraint on the hidden layers can capture high-level representations of the data. The sparsity penalty term is included in the loss function to prevent identity mapping by keeping only a selected set of neurons "active" at any instance. In practice, if the output of a neuron is close to 1, the neuron is considered to be "active", otherwise, it is "inactive". To achieve this, the sparse term is added to the objective function that penalizes $\hat{\rho}_j$ (the average activation of the hidden unit j) if it deviates significantly from ρ (the sparsity parameter). These terms are expressed as:

$$\hat{\rho}_j = \frac{1}{n}\sum_{i=1}^{n}[f_j(x(i)] \tag{2.11}$$

$$\rho_{penalty} = \sum_{j=1}^{S} KL(\rho||\hat{\rho}_j) \tag{2.12}$$

Where S is the number of neurons in the hidden layer. $KL(.)$ is the Kullback–Leibler divergence (KL divergence), which is defined as:

$$KL(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j} \tag{2.13}$$

Here the goal is that $\hat{\rho}_j$ approaches a constant ρ, which is close to zero. Adding the sparse penalty term to the cost function, it can be modified as:

$$J_{sparse}(\theta) = J(\theta) + \beta\sum_{j=1}^{S} KL(\rho||\hat{\rho}_j) \tag{2.14}$$

2.5.2 Data training collection

In particular, with the classical DL/ML training process, the users need to upload their own private data to a central entity (e.g. cloud) for training. Collecting the data to a central entity could lead to significant storage requirements and communication overhead as well as raise privacy concerns [79]. The strengths and weaknesses of the centralized model learning (CML) process are summarized in Table 2.6. These weaknesses are serious obstacles to using CML for traffic analysis as well as catering to the high scalability of 5G and beyond networks.

2.6 Federated Learning (FL)

Recently, strict laws such as the General Data Protection Regulation (GDPR) in European Union [1] completely redefined the data management policy. As a result, FL has been used as

[1] https://gdpr-info.eu/issues/data-protection-officer/

Table 2.6: Summary of the advantages and drawbacks of Centralized ML (CML).

Strengths	**Weaknesses**
• Ability to find a globally optimal ML model • Easy implementation • Can be used by non-expert • Design simplicity	• Requires large storage and computation resources • Requires huge quantities of data training to learn • Has the main server to handle collecting data and training models • Impractical for ultra-low latency environment • Lack of scalability and flexibility • Important overhead and overload for data collection • Private data must be collected in a central entity

an alternative solution to CML. The recent developments in technologies like the exponential increase in data generated by different IoT devices, end devices, computing power, and the particular cloud and edge computing enable the emergence of FL. Edge computing is one of the promising solutions to reduce network congestion and latency that occurs with the cloud. Furthermore, cloud computing offers significant computing and storage resources. With FL, the users do not need to upload their own private data to the cloud for training the central model. It allows many devices to collaboratively train a model while keeping the training data, decentralized in order to preserve data privacy.

In other words, FL is a specific category of distributed ML, where the model is trained on the data located at the decentralized devices [80]. It was first proposed by Google in 2016 [81] in order to preserve data privacy, alleviate the computational burden on the central entity (e.g. cloud), and reduce the latency as well as the communication cost. It attempts to answer the main question: *"Can we train the model without needing to transfer data over to a central location?"* [82]. As shown in Figure 2.19, the main idea of FL is to build a model based on data sets that are distributed across several devices without exchanging the end-user data with the central entity and in turn prevent data leakage. It is an iterative process, wherein each communication round the model can be improved.

In particular, the training process of FL can occur in two different ways: centralized and decentralized. The centralized FL process is characterized by the presence of the aggregation server (Figure 2.19a). In this scenario, the aggregation server or FL server sends the global model to the selected clients/devices and sends them the initialized model. Then, each client trains this model locally using its own data. Once all the updates are received, the FL server aggregates the model's parameters. This process is repeated in various rounds until the desired performance is achieved. Conversely, the decentralized scenario known as serverless

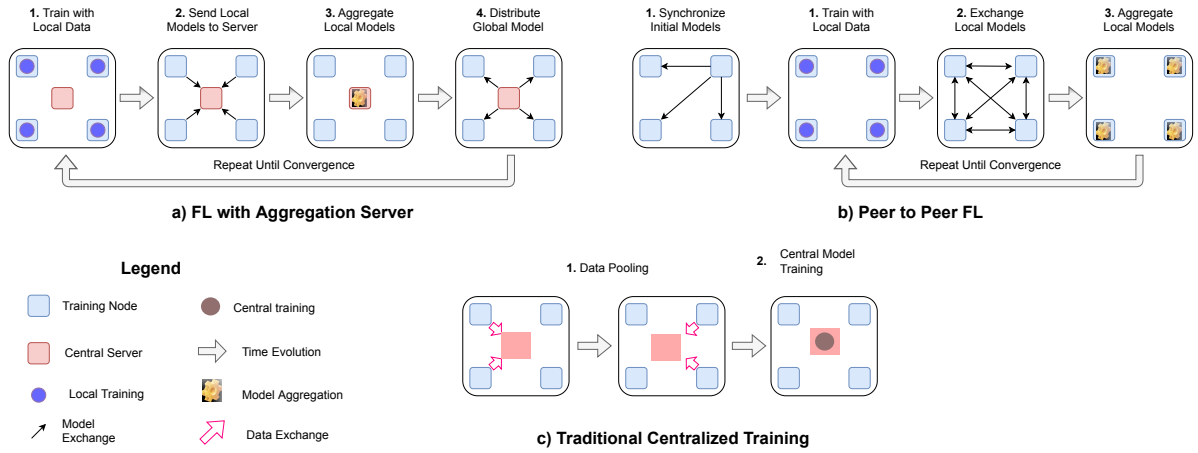


Figure 2.19: Typical FL workflows in comparison to traditional learning based on a centralized data manager. (a) Centralized FL and (b) Peer to Peer FL formulations allow private data to remain local to clients. (c) A general non-FL training workflow where the clients send their data to a central entity for model training.

or peer-to-peer FL does not require an FL server, where each client communicates its trained model with some or all the clients for self-aggregation (Figure 2.19b). As can be seen, the FL process consists of two main phases: local training and model aggregation. The local training is dependent on the designed model while the aggregation mechanism is the heart of FL and aims to share knowledge and hence improve the generalization of the model. Several aggregation mechanisms have been proposed for FL, and Federated-Averaging (FedAvg) is the most widely used algorithm, given its simplicity, efficacy, and robustness [81]. The FedAvg is calculated as follows:

- *Definition: Global Model Aggregation*

$$\theta_{t+1} = \sum_{k=1}^{K} \frac{D_k}{D} \theta_{t+1}^{k} \tag{2.15}$$

Given a client's model update, the equation 2.15 performs the global aggregation at each communication round. θ_{t+1} corresponds to the model parameters at iteration $t+1$, D is the amount of data from all the K clients, D_k is the amount of the data of the client k.

Moreover, there exist three types of FL due to the diversity within the data, namely *horizontal federated learning*, *vertical federated learning*, and *federated transfer learning* [80]. Horizontal FL is applicable to cases where two datasets share the same feature space, but are different in sampling space. On the other hand, vertical FL is introduced when the two datasets share the same sampling space but differ in feature space. Transfer FL applies when the two datasets differ in sampling and feature space.

Conclusion

In this chapter, the main concept of ML was discussed. In this way, the different categories of ML including supervised, unsupervised, semi-supervised, and reinforcement learning as well as several shallow ML models were presented. Then, the basic concept of DL and how it can outperform shallow ML models for learning features are also discussed. Next, this chapter sheds light on the motivation behind federated learning to protect privacy-sensitive data. This chapter proves that ML can be a promising candidate to solve network complex problems in order to achieve fully automated and intelligent traffic analysis, which is our objective in this thesis.

In the next chapter, we will provide a review of some related works. Specifically, we will focus on the literature proposals dealing with the application of ML/DL/FL for intelligent traffic classification, and intrusion detection systems. Besides, the aim of the next chapter is to highlight the research gap and to clearly position our contributions compared to related works.

*****

Chapter 3

ML-enabled traffic analysis: Literature review

Introduction

Efficient traffic analysis and identification form the foundation of intelligent network management. The benefits of traffic analysis range from the optimization of resource utilization to QoS provisioning and security. For example, using the historical data, ML/DL-based models can understand the traffic patterns and the resources automatically allocated according to the traffic patterns identified [83].

According to the paper of Boutaba *et al.* [7], traffic classification is one of the main components of network traffic analysis. Traffic classification is based on flow or packet-based features (attribute). It aims to understand the type of traffic carried on the Internet [4] [5]. For example, it aims to identify the application (YouTube, Netflix, Twitter, etc.) or detect network intrusion.

The purpose of this thesis is to study the application of ML and DL models for network traffic analysis. Therefore, in this chapter, we will focus on traffic analysis, including traffic classification and intrusion detection systems. Thus, this chapter is divided into two main sections: First, we will review the key works from the state-of-the-art of application classification. Then, in the second section, we will deal with the existing studies of ML-based IDS. This literature review is chosen because they are relevant to the contributions to traffic classification and IDS, explaining their contribution as well as their limitation, which drives the need for proposing novel solutions. Finally, we conclude this chapter.

3.1 Application traffic classification

Due to the importance of traffic classification, several approaches have been developed over the years to accommodate the diverse and changing needs of different application scenarios. There are three major approaches to achieving application identification and classification, which are port-based, Deep Packet Inspection (DPI), and ML-based approaches [9].

3.1.1 Traditional techniques

The port-based technique is the simplest and most scalable technique for traffic classification since an analysis of the header packets is used to identify only the port number and its matching with the well-known port numbers. The well-known ports for the protocols are assigned by the IANA [84]. It is also highly robust since a single packet is sufficient to make an application identification. However, over time some limitations of this approach became evident. To avoid detection by this method, P2P applications use dynamic port numbers [85], and also such applications can use ports associated with other protocols for masquerading purposes. Therefore, port-based classification loses its effectiveness in identifying such types

of applications and several research studies confirm that this technique becomes inaccurate e.g. less than 70% accuracy [86] [87].

Thus, to avoid total reliance on the port numbers, the payload-based technique known as Deep Packet Inspection (DPI) has been used. It is based on the packet payloads matching with a well-known signature [9]. A signature is the unique traffic pattern of an application that distinguished it from other applications. DPI checks all the data of a packet, which consumes a lot of CPU resources. As the network speed increases the performance of payload-based traffic classification will inevitably drop [88]. Also, it fails to provide fine-grained traffic classification due to encrypted packets and privacy issues [89] [4]. Moreover, DPI requires manual effort to build the application signature and for new protocols, it needs to update the signatures in the signature database [90], which makes it not scalable with the large enormous of mobile applications [91]. All these made the payload-based classification techniques no longer useful as mobile applications become more dynamic, diverse, and complex.

3.1.2 ML for application classification: Literature review

Due to the drawback of the port-based and DPI techniques, ML has been used as an alternative solution. The advent and rapid growth of such models have brought many improvements in traffic classification. As a result, many research works have already used ML for traffic classification. Here, we review key works from the state-of-the-art application traffic classification. For instance, Cherif *et al.* [92] used XGBoost for traffic classification for the first time in 2019. Using captured traffic dataset, the authors evaluate the performance of XGBoost network traffic classification. Comparison results demonstrate that the XGBoost algorithm outperforms Naive Bayes (NB), KNN, and DT models. However, the model was evaluated using a private dataset.

Perera *et al.* [93] compared different popular supervised learning algorithms (MLP, NB, RF, and DT). The evaluations were carried out together with two different feature selection techniques. The experiment results show that RF and DT models perform well with the highest accuracy and the short model building time. However, the generalization capability of the models was not discussed.

Alzoman *et al.* [94] evaluated the performance of the port-based method as well as four supervised ML algorithms, namely, SVM, RF, KNN, and DT. The evaluation results demonstrate that ML models, in particular, DT-based models perform the best. In contrast, the KNN and SVM provided the lowest average accuracy, especially with the multi-classification scenarios. Although the work compared several models, the authors did not explore the recent ones.

To improve also the classification performance, some researchers started to use DL-based models in order to take advantage of the benefits of the deep features extracted through DL.

3.1.2.1 DL-based approaches

As DL helps to eliminate the manual features engineering task and has shown success in network traffic classifications [3], Zhang *et al.* [95] proposed a DL-based model in order to classify the traffic into one of several classes (Bulk, Database, Interactive, Mail, Services, WWW, P2P, Attack, Games, Multimedia). It consists of the Stacked AutoEncoder (SAE) and Softmax classifier. SAE was used for feature extraction and Softmax was used as a supervised classifier. The experimental results show that the proposed model outperforms the SVM model. However, only labeled data have been used during the training process.

Lotfollahi *et al.* [96] proposed a DL-based approach, called `DeepPacket`, which integrates both feature extraction and classification phases into one framework. The authors compared two models (i.e. SAE and CNN) in order to classify network traffic. The experiment results show that `DeepPacket` achieved a good accuracy for the traffic categorization tasks. However, to evaluate the framework performance the authors used one dataset, as well as only labeled data, have been used during the training process.

Lyu *et al.* [97] applied DL models to the classification of four types of traffic. Using traffic data collected from the real network environment, the experiment results show that the MLP model outperforms CNN in terms of accuracy, training, and classification times.

3.1.2.2 Ensemble learning-based approaches

As each shallow ML or DL model has its own strengths and weaknesses, the researchers started to use the combination of several models in order to overcome their individual shortcomings and in turn improve the classification performance. One of the main advantages of ensemble learning is its ability to allow the production of better predictive performance compared to a single model. In such context, Gomez *et al.* [98] proposed an ensemble algorithm based on DT and then compared its performance with seven ensemble algorithms. The experiment results demonstrate that the ensemble model overcomes the single model in terms of accuracy. However, the proposed approach was evaluated on small-scale experiments as well as on private data.

In the same direction, Yang *et al.* [99] proposed a novel stacking ensemble classifier. This ensemble combines seven models in a two-tier architecture (Figure 3.1). Specifically, on the first-tier, KNN, SVM, RF, AdaBoost, Extra-Tree, and Gradient Boosting models are trained independently on the same training set. Then, on the second tier, XGboost uses the prediction of the first-tier classifiers in order to improve the final prediction results. The comparative analysis against the individual single classifier and voting ensemble demonstrates the effectiveness of the proposed ensemble. However, one dataset with five classes has been used to evaluate the performance of the proposed ensemble. Also, the generalization capability and the impact of ensemble hyper-parameters on the classification performance were not discussed.

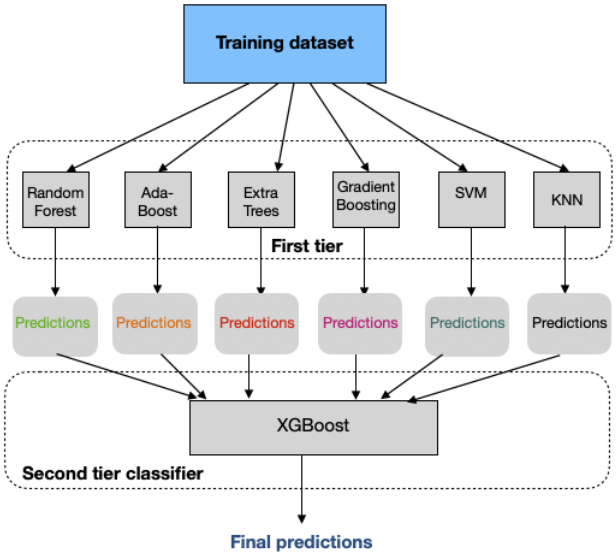


Figure 3.1: Flowchart for the proposed ensemble classifier in a two-tier architecture [99].

Furthermore, as DT-based ensemble models have advanced considerably and are widely used for traffic classification, Eom *et al.* [100] compared five DT-based models, LightGBM, XGBoost, GBM, RF, and DT. The comparison results indicate that the ensemble model generally outperforms the single DT model. In particular, the LightGBM model achieved the best results and exhibits faster training than the other ensemble models. However, this work classified only the network traffic into different classes according to the QoS requirements (e.g. Streaming, Web Browsing, Chat, etc.) and not by application (the exact application generating the traffic). Thus, this could not prove the performance of the proposed model.

In addition, the `App-Net` scheme was proposed by Wang *et al.* [101]. The `App-Net` scheme combined bi-LSTM and CNN in a parallel way for mobile traffic classification. Then, the feature extracted from such models is fed to a softmax layer to produce a probability distribution for the multi-classification task. The results show that combined different DL models can extract more relevant features and thus achieve a relatively good performance and outperform the single models. However, the training cost of the proposed scheme was not discussed and the experiment was carried out on a private dataset.

In a similar way, Lin *et al.* [102] proposed an end-to-end traffic classification DL-based approach, called `TSCRNN`. `TSCRNN` is an ensemble DL model that combines CNN and Bi-LSTM in a sequential way. Using two popular datasets, two scenarios of classification tasks (binary and multi-classification) were proposed in the experiments. The results demonstrate that `TSCRNN` outperforms other classification methods based on the DL. However, only labeled data was considered for the experiment.

To further improve the classification performance, some researchers started to combine DL-based models and shallow ML models. More specifically, they take advantage of the

benefits of the deep features extracted through DL to train shallow models and make the final decision. For example, Obasi *et al.* [103] proposed a blending model by combining different shallow ML and DL models for the classification of encrypted network traffic. In this work, the authors have used the output of the base model (DL models) as input to the XGBoost classifier for the training process. The evaluation tasks of the proposed scheme were done on the ISCX VPN-nonVPN dataset. The results demonstrate the potential of the XGBoost to boost the performance of the base models. However, the authors did not study the generalization capability of the proposed ensemble.

Based on the findings in their earlier paper [103], Obasi *et al.* [104] proposed a novel ensemble learning technique that is based on different shallow ML models, DL models, and stacking techniques, called `CARD-B`. As shown in Figure 3.2, `CARD-B` consists of the outputs from the different base classification models including Capsule Neural Networks+XGBoost, ANN+XGBoost, DT+AdaBoost, RF+XGBoost, and DT+XGBoost models that are merged and fed into a new classification model, RF model, in the second stage for training and classification tasks. The proposed ensemble was evaluated using the ISCX VPN-nonVPN dataset and the results demonstrate that the combination of several models can improve the classification performance. However, the experiment was carried out on a partial dataset (one scenario) and the generalization capability of the models as well as the training cost of the proposed model was not discussed.

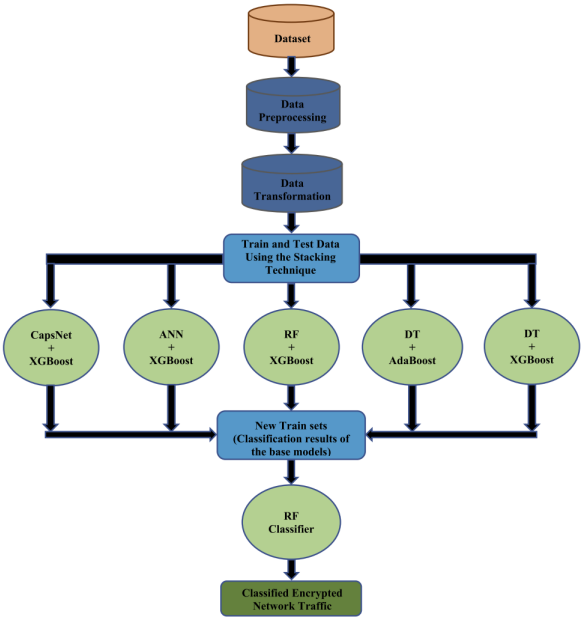


Figure 3.2: Training process of CARD-B [104]

In summary, all the works discussed above require fully labeled data. However, as new types of traffic emerge every day (and they are generally not labeled), this opens a new challenge to be handled. In fact, labeling data needs to be done in a continuous way, and

thus, it is a difficult and time-consuming task [105].

3.1.2.3 Semi-supervised learning-based approaches

To address this issue, the researchers started to reformulate traffic classification into semi-supervised where both labeled and unlabeled data are used at the same time.

For instance, a clustering labeling method has been proposed. Such a method facilitates traffic flow classification independent of known traffic classes [106]. In this context, Perera *et al.* [29] used the K-means model in order to assign the group based on similar types of traffic (Figure 3.3). Then, the new labeled data was used in an individual way by five supervised learning, which are SVM, DT, RF, and KNN, for the training and evaluation. However, using a clustering mechanism for data labeling can increase the model complexity, as well as the used features during the training, were selected manually.

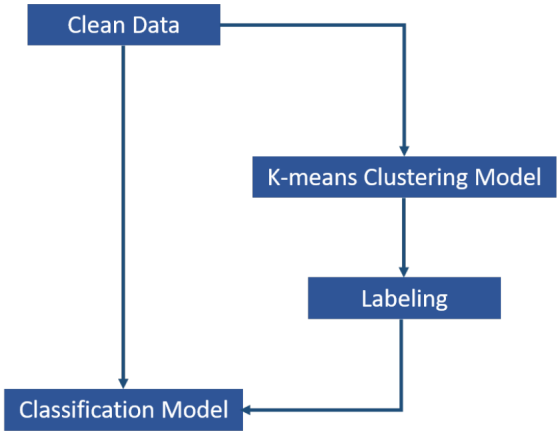


Figure 3.3: Training process of the semi-supervised model [29]

Rezaei *et al.* [107] proposed a semi-supervised approach for traffic classification. In particular, the proposed approach is based on transfer learning where the CNN model was pre-trained on a large unlabeled dataset and then the learned model is transferred to be re-trained on a small labeled dataset. The experiment results demonstrate that the pre-trained model can boost classification accuracy. However, the proposed approach was trained and evaluated on a limited number of applications (i.e. five Google applications).

Another semi-supervised approach has been proposed by Wang *et al.* [108]. Specifically, DPI has been used to label a part of traffic flows of known applications and each labeled application is categorized into four QoS classes (Voice/Video Conference, Interactive Data, Streaming, Bulk Data Transfer). Then, this data is used by semi-supervised learning algorithms (Laplacian SVM) to classify the traffic flows of unknown applications. However, the evaluation is performed using a private dataset, and using a shallow learning classification makes the model cannot scale very well.

3.1.2.4 Traffic classification in SDN

In recent years, innovation has driven scale-up traffic analysis capabilities. The intersection between Software Defined Networking (SDN) paradigm and ML brings new chances to provide intelligence inside the networks and further overcome their challenges (e.g. complexity). SDN separates the data plane from the control plane in order to dynamically program the network. The centralized SDN controller has a global network view, which facilitates traffic analysis. Several researchers argue that with the introduction of SDN, there is a high potential for collecting data from forwarding devices, which need to be handled by the ML due to their complexity [109] [110].

In this context, Raikar *et al.* [111] proposed a traffic classification using supervised learning models in the SDN environment. Specifically, a brief comparative analysis of SVM, NB, and KNN has been done, where the accuracy of traffic classification is greater than 90% in all three supervised learning models. However, the authors used only three applications (i.e. SMTP, VLC, HTTP) for the model evaluation, which is therefore not enough to verify the scalable performance of the models.

In the same line, Amaral *et al.* [112] introduced a traffic classification architecture based on an SDN environment deployed in an enterprise network using shallow ML models. In this architecture, the controller collected the flow statistics from the switches, and then the preprocessing step is used followed by several classifiers individually, which are RF, GBM, and XGBoost. The accuracy of each application is used as an evaluation metric. However, this study lacks other performance metrics such as F-score and model generalization capability.

Moreover, Uddin *et al.* [91] introduced a traffic classification framework, called `Traffic Vision` that identifies the applications and their corresponding flow type in real-time. `Traffic Vision` is deployed on the SDN controller, and one of the kernel modules of `Traffic Vision` is named TV engine, which has three major tasks: (i) collecting, storing, and extracting flow statistics and ground-truth training data from end devices, (ii) building the classifiers from the training data, (iii) and applying these classifiers to identify the application and flow-types in real-time and providing this information to the upper layer application. As a proof concept, the authors developed two prototypes of "network management" services using the `Traffic Vision` framework. The classification task of the TV engine has two modules, which are application detection using DT and flow-type detection using KNN with $K = 3$. Both of these classifiers show more than 90% of accuracy. The shortcoming of this work is the classification of the popular applications and the ignorance of other applications which can cause problems in enterprise or university networks.

Furthermore, Wang *et al.* [113] proposed an SDN home gateway (HGW) framework for encrypted traffic classification, called `DataNet`. As shown in Figure 3.4 this framework is embedded in the HGW. This classification was achieved through the use of several DL

models, which are MLP, SAE, and CNN. They used the "ISCX VPN-nonVPN" encrypted traffic dataset [114], which consists of 15 encrypted applications (i.e. Facebook, Skype, Hangout, etc.). The performance evaluation shows that MLP consumes the least computing resources among the three models. Also, it shows that both SAE and CNN models have better performance in classifying online chatting applications, whereas MLP classifies very well Netflix and Email applications. However, only labeled data have been used during the training process.

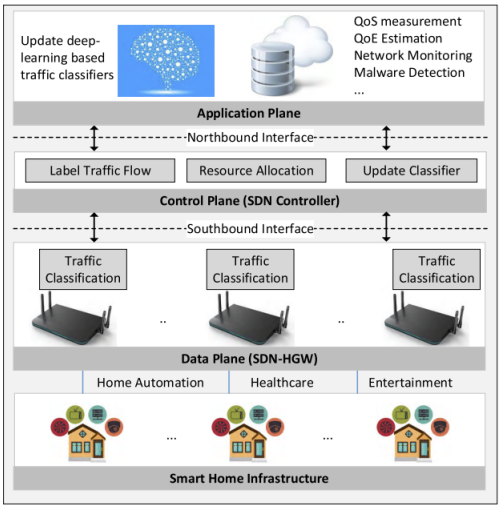


Figure 3.4: Overview of the SDN-HGW framework [113].

3.1.2.5 FL-based approaches

Last but not least, as the traffic data has increased, FL can be a promising solution for traffic classification. The FL helps the model to classify the traffic without communicating data to a central entity [12]. It keeps the traffic where it was generated (chapter 2, section 2.6). Recently, FL has started to attract the attention of researchers for traffic classification [115] [116] [117]. Although the privacy preservation offered by FL during the classification task, the proposed models lacked application type identification, and the generalization metric to verify their effectiveness, as well as they did not take advantage of the abundance of unlabeled data.

3.1.3 Shortcomings and Research Gaps

Table 3.1 summarises the related work papers with their key contributions/findings and limitations. The main research gaps that may shape the motivation of the present work are summarized as follows:

Table 3.1: Summary of investigated solutions to design ML-based traffic classification

Ref.	Models	Datasets	Contribution	Limitations
[92]	XGBoost	N/A	Performance evaluation of the XGBoost algorithm for traffic classification	Private dataset was used
[93]	MLP, NB, RF, DT	Waikato Internet Traffic	A comparative analysis of several supervised machine learning classifiers has been presented.	The generalization capability was not discussed.
[94]	SVM, RF, DT, KNN	Moore dataset	A comparative analysis of ML-based and port-based method has been proposed.	Other models were not considered.
[98]	DT-based ensemble	N/A	DT-based ensemble model has been proposed.	Small scale experiment.
[99]	KNN, SVM, RF, AdaBoost, ExtraTree, XGBoost	QUIC dataset	Combine several models in a two-tier architecture.	Small scale experiment.
[100]	LightGBM, XGBoost, GBM, RF, DT	ISCX dataset	Compare five DT-based models in terms of classification performance, training, and classification time.	Classified only the network traffic into different classes according to the QoS requirements.
[111]	SVM, NB, KNN	N/A	A comparative analysis of several models has been done.	Only three applications were used during the classification
[112]	RF, GBM, XGBoost	N/A	A comparative analysis of DT-based models in an SDN environment deployed in an enterprise network.	Lacks other performance metrics (e.g. $F-score$ and model generalization).
[91]	DT, KNN	N/A	A traffic classification framework, called `TrafficVision` that identifies the applications and their corresponding flow-type in real-time has been proposed.	Classify only the popular applications.
[95]	SAE	Moore dataset	A DL-based model in the SDN-based environment	Only labeled data have been used.
[113]	CNN, SAE, MLP	ISCX VPN-nonVPN	An encrypted data classification framework called `DataNet`, which is embedded in the SDN home gateway, has been proposed.	Only labeled data have been used.
[96]	CNN, SAE	ISCX VPN-nonVPN	An integration of both feature extraction and classification phases into one framework.	One dataset has been used to evaluate the proposed framework performance.
[97]	CNN, MLP	N/A	A comparative analysis of two DL models in term of accuracy, training and classification times.	Small scale experiment.
[101]	Bi-LSTM, CNN	N/A	A combination of DL-based models in a parallel way for mobile traffic classification.	The complexity approach was not discussed.
[102]	Bi-LSTM, CNN	ISCXTor2016, ISCX VPN-nonVPN	A combination of DL-based models in a sequential way for mobile traffic classification.	Only labeled data was considered for the experiment.
[103] [104]	XGBoost, ANN, CNN, LSTM	ISCX VPN-nonVPN	A combination of different shallow ML and DL models for classification of encrypted network traffic.	The generalization capability was not discussed.
[29]	K-means, RF, DT, KNN	IP Network Traffic Flows Labeled with 75 Apps	K-means application for traffic labeling.	The computation cost was not discussed.
[107]	CNN	QUIC, Unlabeled Waikato, and Ariel	A transfer Learning have been used in order to train a model with large quantities of unlabeled data and a few labeled observations.	The proposed scheme classifies only five Google applications.
[108]	SVM	N/A	A method using a semi-supervised learning algorithm, which is Laplacian SVM to classify the traffic flows of unknown applications.	the evaluation is performed using a private dataset.

- The homogeneous ensemble models (e.g., bagging and boosting) are widely used for traffic classification. However, despite the performance of the heterogeneous ensemble and the combination of shallow models and DL, is still rarely used in traffic classification.
- Despite tree-based models being widely used for traffic classification tasks, their generalization capability was not evaluated.
- Most of the existing solutions try to improve the classification performance of the used model. Despite the performance of the proposed ensemble models, their generalization capability was not evaluated and improved.
- Most of the previous solutions missed extensive comparison against some recent models. They used only a few models like SVM, RF, and MLP as baselines.
- A lot of datasets are partially labeled [105] due to many reasons and unlabeled portions of the data, which can also provide informative characteristics, are ignored during the model training with the existing solutions.
- Most of the studies evaluated with private data or not mentioned the source of the benchmark datasets. This drawback can limit the re-use and the comparison with those works.

3.2 ML-based Intrusion Detection Systems: Literature review

Due to the increase in network complexity, the network systems are subjected to several security vulnerabilities including intrusions. Data security and privacy issues are closely related to users' information, and in turn, protecting data security and privacy is a crucial task for network operators. According to Base and Mell [118], intrusions are defined as *"Attempts to compromise the confidentiality, integrity, or availability of a computer or network, or to bypass the security mechanisms of a computer or network"*. Thus, regarding the increasing diversity of cybersecurity attacks, providing an efficient and accurate technique to mitigate different types of attacks is one of the needs of the hour.

Intrusion Detection System (IDS) is a well-known software system for monitoring and detecting malicious traffic [119]. It can be defined as *"A combination of software and hardware which monitors network or systems to identify malicious activities and gives immediate alerts"* [120].

1. **IDS based deployment:** Depending on where the detection takes place, IDS can be divided into two broad categories, which are host-based IDS (HIDS) and network-based IDS (NIDS) [121] [122].

   - ***Host-based IDS (HIDS): HIDS*** is a software component, e.g. anti-virus, installed on a host device to monitor and analyze the activities of an individual host. However, HIDS suffers from severe disadvantages such as it only monitors the host as well as it requires the collection and management of the sensitive datasets [123].

   - ***Network-based IDS (NIDS): NIDS*** continuously monitors and analyzes the traffic in the network to detect possible network attacks. It has attracted numerous researchers' attention from both industrial and academic institutes in response to the increasing cyber-attacks [120].

2. **IDS based methodology:** The detection of malicious traffic can be carried out through the IDS. Depending on the analysis technique, there are two major approaches for IDS, which are signature-based IDS and anomaly-based IDS.

   - ***Signature-based IDS:*** A signature-based IDS approach uses a predefined signature database of different attacks in order to detect the matching traffic. Although signature-based detection is accurate for detecting known attacks, it became less efficient and scalable because it cannot identify the unknown or novel attacks (as it does not have the signature of novel attacks in its signature database) [124]. Besides, the computational cost is usually high and takes time, and in turn, it becomes a critical issue for real-time attack detection.
   - ***Anomaly-based IDS:*** In contrast to signature-based IDS, anomaly-based IDS approaches attempt to learn the normal behaviors and classify every deviation from the normal behaviors as an anomaly or intrusion [125]. In other words, anomaly-based IDS made the detection of never-before-seen attacks possible.

In general, intrusion detection can be considered as a particular case of traffic classification task by classifying the incoming traffic into normal or attack. As a result, some advances such as ML including shallow models, DL, and FL have greatly improved intrusion detection or attack classification tasks without extensive human-based intervention as maximum as possible. These models can automatically diagnose and detect attacks using flow and packet-based features. The models use these features to find useful patterns to detect attacks that did not take place in the past. The integration of ML/DL/FL and IDS leads to the appearance of several works which are summarized in the following. Thus, our main focus in this thesis is on network intrusion detection systems (NIDS) and particularly anomaly-based IDS using ML-based models.

3.2.1 Conventional ML/DL related work

Using the shallow ML and DL models, many solutions have been proposed for network traffic classification. For example, Dutta *et al.* [126] proposed a hybrid anomaly detection method that combines the AE with DNN. It can be seen from Figure 3.5 that this method takes advantage of both AE and DNN in order to extract the relevant features through the AE model and reduce the error rate of classification, respectively. The experimental results demonstrate that the proposed method outperforms some baseline algorithms such as DNN and RF models. However, the model uses only labeled data and the authors did not evaluate the model on attack type identification (i.e. multi-classification scenario).

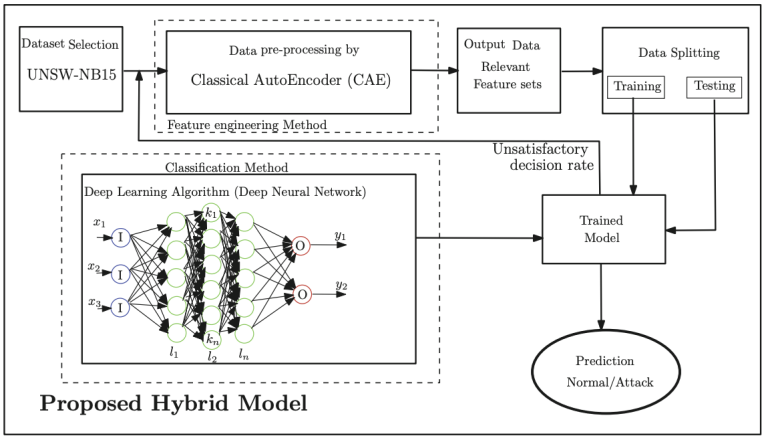


Figure 3.5: Training process of the hybrid anomaly detection method [126]

Similar to the applications, new types of attacks emerge every day and thus it brings new challenges for labeling data. Therefore, the researchers started to use a semi-supervised learning model for IDS. In such context, Soheily *et al.* [127] proposed a hybrid intrusion detection method, called `kM-RF`. `kM-RF` is a semi-supervised model that used K-means clustering and RF algorithms. In particular, K-means clustering pre-processing results have been used as input data for RF classifier model construction and then intrusions detection. The experiment results show that the proposed model outperforms the other state-of-the-art methods. However, it has used the `ISCX dataset` 2012 which contains outdated attacks.

In the same direction, Verma *et al.* [128] proposed a semi-supervised IDS by combining K-means and boosting-based models (i.e. XGBoost, AdaBoost). Further, to verify the classification performance, the proposed IDS was compared with different existing works. In terms of performance, the proposed scheme achieved an accuracy of 84.25% using the `NSL-KDD` dataset. The proposed scheme was also evaluated on an outdated dataset.

Wagh *et al.* [129] proposed a semi-supervised model for IDS using boosting-based models. The labeled data were used for model training, whereas the unlabeled data were used for the testing. Then, the unlabeled data with high confidence with their predicted labels will

be added to the labeled set. All experiments were carried out for attack classification using KDD99 dataset, which is an old dataset.

Hara *et al.* [105] proposed an IDS system using a semi-supervised learning model. They used the AE model in order to extract the relevant features. The experimental results demonstrate that the proposed model can achieve high accuracy with a minimal amount of labeled data as compared to DNN. To evaluate the performance of their solution, the authors used NSL-KDD dataset, which is quite an old dataset and can miss modern network behaviors.

3.2.2 FL related work

Recently, FL for DL was also investigated and implemented for IDS. Nguyen *et al.* [130] introduced an FL system for detecting compromised IoT devices, called DÏoT. This is the first system that deployed FL for IDS. It consists of two components, which are security gateways and IoT security services. The security gateways use the local data to train the local models (GRU) and the IoT security service aggregates the local models into a global model.

Moreover, to improve the system performance and to increase the weights of important update parameters, Chen *et al.* [131] introduced an attention mechanism. More specifically, the authors proposed an IDS for wireless edge networks based on the FL-based Attention GRU scheme, called FedAGRU. The FedAGRU system uses the attention mechanism on the FL server in order to assign different weights to the model parameters of different clients. The attention mechanism is a mechanism used with neural network models that can mimic the visual attention mechanism in humans. It becomes popular for natural language processing tasks [132] [133]. It has the characteristics of information filtering used to calculate the intermediate output weight of the neural network model and to discard some of the irrelevant information [134]. The experiment results show that FedAGRU provided an improvement of detection accuracy by approximately 8%. However, only the labeled data have been used to train the proposed model.

Zhao *et al.* [135] proposed an intelligent intrusion detection model, called FL-LSTM. In particular, the proposed model uses LSTM in FL architecture to effectively detect the intrusion. The simulation results on the SEA dataset (i.e. produced by the AT&T Shannon Lab) demonstrate that the FL-LSTM model can accurately detect the intrusion. Although FL-LSTM improved the classification accuracy it lacked attack-type identification.

Mothukuri *et al.* [136] proposed an ensemble FL-based attack detection and classification in IoT networks. The authors combine RF and GRUs models to construct their ensemble. In other words, they used RF in order to combine the predictions from the GRUs model to further improve the classification performance of the FL approach. The experimental results demonstrate a minimized error rate in predicting attacks and a reduced number of

false alarms in comparison to the centralized ML approaches. However, the proposed scheme lacked attack-type identification.

Similarly, Attota *et al.* [137] proposed an ensemble FL-based intrusion detection, called `MV-FLID`. Specifically, the authors have developed three FNN models for three views (i.e. Biflow View, Packet View, and Uniflow View), and the outcomes of these models are sent to an ensembler model (RF), which combines the predictions of these models and classifies the instances. The results show that the FL approach can outperform the Non-FL one.

Rahman *et al.* [138] proposed an FL-based IDS scheme for IoT networks. The results showed that FL outperforms self-learning (i.e. without device collaboration) and achieved competitive accuracy in comparison to centralized learning. However, the proposed scheme was carried out using an outdated dataset (`NSL-KDD` dataset).

Friha *et al.* [139] proposed federated DL-based IDS for the cybersecurity of agricultural-IoT networks (Figure 3.6), called `FLIDS`. `FLIDS` uses three DL-based models, namely, DNN, CNN, and RNN and the experimental results show that it achieved a competitive performance to the centralized model. However, the proposed system assumes that both data and labels are initially available on the devices.

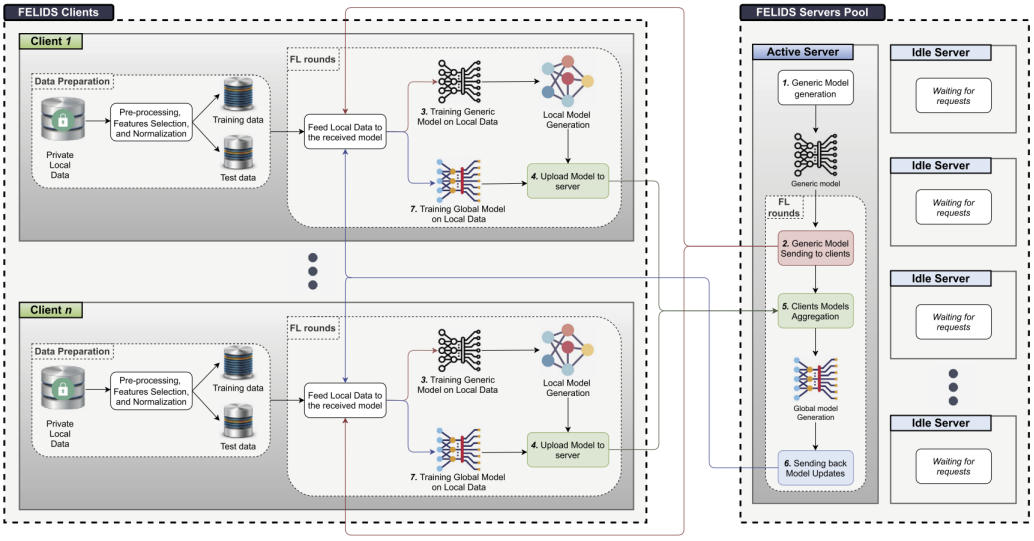


Figure 3.6: FLIDS architecture [139]

To detect the types of attacks against industrial cyber-physical systems (CPS), Li *et al.* [140] proposed a federated DL scheme, called `DeepFed`. A combination of CNN and GRUs has been used for intrusion detection. The experimental results with a real industrial dataset, released by the Mississippi State University in 2014 [141], demonstrate the high accuracy of `DeepFed` as compared to some other FL approaches. However, the proposed scheme did not benefit from the unlabeled data abundance.

Since FL is trained on sensitive user data, it may suffer from reverse engineering data. Thus, to preserve better the privacy of end-users, Al-Marri *et al.* [142] proposed a federated

mimic learning by combining FL and mimic learning. Using the NSL-KDD dataset, the results show that the federated mimic learning-based method can achieve 98.11% detection accuracy which is close to the centralized DL performance while improving the privacy preservation of the user data significantly. However, the `NSL-KDD` dataset is obsolete and can lack modern IoT-based attacks.

3.2.3 DDoS attack detection or classification

A Distributed DoS (DDoS) attack is a complex form of a Denial-of-Service attack (DoS). It aims to exhaust the target networks with malicious traffic. Consequently, the detection and mitigation of DDoS attacks in real-time is a critical task. As a result, in recent years, several ML/DL models are used for DDoS attack detection. For instance, Niyaz *et al.* [143] proposed a DL-based system for DDoS attack detection in an SDN environment. Specifically, SAE has been used for feature reduction in an unsupervised manner. Then, the traffic classification was performed with the softmax layer in the scenario of 2-class (DDoS/benign traffic) and the 8-class scenario including normal and seven kinds of DDoS attacks. The experimental results show that the SAE model achieved higher performance compared to the MLP model.

Also, Aamir *et al.* [144] proposed a semi-supervised model for DDoS attack detection. In the first step, PCA was used for data reduction and feature extraction. In the second step, a clustering algorithm was used to label the data based on their cluster. Finally, after label assignment, several supervised models are applied for training and attack detection. The experimental results show that the RF model is more accurate than KNN and SVM.

Although, the DL-based models are efficient for DDoS attack detection or classification, combining deep and shallow models can take advantage of both techniques and hence improve the performance of the security system. Krishnan *et al.* [145] proposed a hybrid model by combining DL and shallow ML models, called `VARMAN`. AE was used to generate a new reduced 50 features from the `CICIDS-2017` dataset as well as to optimize the computation and memory usage. Then, RF acts as the main DDoS classifier. The experiments demonstrate that `VARMAN` outperforms other DL-based models and this is attributed to the fact of the combination of different models.

FL also has been used for DDoS detection. In this context, Li *et al.* [146] proposed a federated DDoS attack detection, called `FIDS`. Using the `CICDDoS-2019` dataset the results show that FIDS improves the performance of classification and stability compared with baselines. However, `FIDS` uses only labeled data during the training process.

3.2.4 Shortcomings and Research Gaps

Table 3.2 highlights the key points of different ML-based IDS schemes proposed in the literature. Despite the performance of the proposed solutions, they suffer from several drawbacks

Table 3.2: Summary of investigated methods to design ML-based IDS

Ref.	Models	Datasets	Semi-supervised Learning	Attack Type Detection	Federated Environment
[126]	AE, DNN	UNSW-NB15	x	x	x
[127]	K-means, RF	ISCX 2012	✓	x	x
[128]	K-means, XGBoost, AdaBoost	NSL-KDD	✓	x	x
[129]	N/A	KDD99	✓	✓	x
[105]	AE	NSL-KDD	✓	x	x
[130]	GRU	N/A	x	✓	✓
[135]	LSTM	SEA dataset (produced by the AT&T Shannon Lab)	x	x	✓
[136]	GRU, RF	Modbus dataset	x	x	✓
[137]	ANN, RF	MQTT	x	x	✓
[139]	DNN, CNN, RNN	CSE-CIC-IDS2018, MQTTset, InSDN	x	✓	✓
[146]	GRU	CICDDoS-2019	x	✓	✓
[138]	DNN	NSL-KDD	x	✓	✓
[143]	SAE, ANN	N/A	✓	✓	x
[131]	GRU, Attention mechanism	KDD99, CICIDS-2017, WSN-DS	x	✓	✓
[145]	AE, RF	CICIDS-2017	x	✓	x
[142]	MLP	NSL-KDD	x	✓	✓
[140]	CNN, GRU	Gas dataset	x	✓	✓

and challenges that should be considered when designing and evaluating the FL-based approaches. The main research gaps that may shape the motivation of the present work are summarized as follows:

- Most of the existing solutions proposed for IDS are evaluated on outdated datasets (e.g., KDD99, NSL-KDD) as well as in one scenario either binary classification (benign/attack) or multi-classification (attack type identification) and not both scenarios.
- Most of the existing solutions assume that both data and label data are available. For labels, this assumption seems unrealistic, as this would mean that a human would have already tagged all the network traffic. This is difficult in practice (i) due to the resource constraints on the devices/clients as well as some edge nodes and (ii) due to the difficulty of manually labeling data on such devices which are far from human reach.

Conclusion

In this chapter, we have reviewed some related ML-based solutions for network traffic analysis, including traffic classification and IDS. According to the literature review, it appears how ML-based solutions are improving traffic classification by providing network operators with better diagnostic tools, helping find useful patterns, and hence can optimize the resources in a complex environment. However, the outlined shortcomings of the proposed solutions have not answered all the technical questions. In particular, the current literature shows that it is promising to classify network applications using ML-based approaches, but there still exist issues related to the performance and generalization capability of the proposed models as well as the exploration of unlabeled data. For example, strategies such as bagging and boosting are widely used to build ensemble models. Despite the tree-based models being widely used for traffic classification, the generalization capability was not discussed. Also, our literature review shows that despite the heterogeneous ensemble learning performance, it is still rarely used in traffic classification as well as its generalization capability was not evaluated.

Furthermore, as presented previously, the majority of the existing traffic classification solutions use supervised learning algorithms and have focused only on the labeled data where the unknown flows were not considered. Even a few semi-supervised models have been proposed; they did not study the impact of the unlabeled ratio on the performance of the final model, nor the generalization performance of the final model. Consequently, an ensemble learning model and semi-supervised model for traffic classification (Chapter 4 and 5) are proposed to address the aforementioned challenges.

On the other hand, although the effectiveness of FL for IDS maintains the privacy of the client's data as well as reduces transmission latency and communication overhead by sending model parameters instead of raw data, unfortunately, most of the existing solutions focus on supervised modes. In fact, the clients are far from human reach, which made accessing them to label their data a difficult and impractical task. Thus, to be more realistic, a novel semi-supervised approach, based on FL, called `FLUIDS`, is designed in Chapter 6.

As such, in the following chapters, we will present our contributions in order to tackle the limitations that appeared during the literature review by proposing ML/DL-based models.

*****

Part I
Conclusion

This part consists of two chapters, which are (i) background and (ii) literature review. The objective of the background chapter is to provide a theoretical background of the basic concepts, which are useful to understand the state-of-the-art approaches as well as our contributions. On the other hand, the literature review chapter detailed the related work papers with their key contributions and limitations. Based on these two chapters, we have noticed that while ML models, and especially DL, are becoming the de-facto knowledge discovery approach in traffic analysis, it appeared new challenges that require new and efficient solutions. In particular, this part outlined the state-of-the-art approaches and models' backgrounds associated with them. Although the shallow ML/DL/FL-based approaches have demonstrated benefits for traffic analysis, the current setting of such approaches has not answered all the technical questions and problems. Consequently, there is a need to consider further improvements to address the research gap raised in the literature review. Therefore, the next part will present deeper investigations on ML/DL-based models for traffic classification in order to provide better network service with less frustration for the end-user.

Part II
Contributions

We have seen in Part I the huge use of ML for traffic analysis and its importance to uncover the complex pattern within traffic behavior. As we can see that intelligent traffic analysis nowadays raises many challenges for network operators. In Part II, deeper investigations on the challenges of intelligent traffic analysis that need to be addressed will be conducted in the following chapters. These challenges come mainly from models' overfitting and their generalization capabilities on the training data. Also, the network datasets are generally partially labeled due to many reasons making it more complicated to analyze the traffic, hence efficient mechanism based on both unlabeled and labeled traffic is indispensable. In addition, enhancing the performance of the models is not enough for intrusion detection, especially with the recent privacy concerns of user data wherein collecting data can cause some damage to the central entity if the traffic contains some attacks. Also, collecting the data in a central entity during model training is expensive in terms of storage resources. Thus, proposing a new solution with further privacy and security improvement is needed. As a result, in this part, three solutions for application classification and intrusion detection will be discussed. For each solution, at least two datasets and different scenarios will be used for the performance evaluation.

Chapter 4

Ensemble-based Deep Learning model for network traffic classification

Introduction

Traffic classification is a subgroup of traffic analysis strategies that aims to classify traffic into predefined categories. It is a basic requirement for providing a good QoS in network management and can help Internet Services Providers (ISPs) manage efficiently their infrastructures. In this context, several techniques such as port-based classification and Deep Packet Inspection (DPI) have been proposed. However, as mentioned in the previous chapter, such techniques are becoming less efficient to handle and classify because the applications can use dynamic port numbers or ports associated with other protocols to hide from network security tools. Also, DPI checks all packet data, which consumes a lot of CPU resources and can cause a scalability problem. To avoid these issues and to efficiently classify the traffic, more intelligence needs to be deployed. With the progress of high-performance computing, ML has been used as an alternative approach in recent years. Using ML for traffic classification can achieve an acceptable trade-off between computation complexity and accuracy [90]. Consequently, several models have been used for traffic classification. Among these models, DT-based models are one of the most used models for traffic classification [112]. This may be attributed to the fact that DT models are simple and can be easily understood by human experts [147]. In particular, as shown in the literature review (Table 3.1) RF and boosting models, which are based on DT, are widely used for classification tasks.

In fact, RF and boosting models are ensemble models that have been proposed in order to break through the limitations of a single model. Ensemble models are widely used in several applications due to their remarkable generalized performance [47] [148]. Since the ensemble models are based on the combination of several models, the difficulty of choosing the appropriate model can be avoided to some extent. The ensemble models can be homogeneous in bagging and boosting and heterogeneous in blending and majority voting ensemble. Although the performance of RF and boosting models, are based only on the DT model and hence can suffer from overfitting issues. In contrast to the homogeneous ensemble models, the heterogeneous ensemble takes advantage of different models. The majority voting algorithm uses a voting method to combine the output of the models. On the other hand, a blending ensemble consists of two levels: base classifier and meta-classifier. Indeed, through the use of a meta-classifier, that combines the base classifier, blending outperforms the majority voting-based ensemble as well as the homogeneous ensemble [47]. The meta-classifier tries to correct the errors that occur during the learning process of the base classifiers. Using a non-linear meta-classifier (e.g. DL models) can explore the non-linear relationship among the base-classifiers and in turn, gives promising results as well as outperforms the linear meta-classifier [48]. Although the effectiveness of the blending ensemble in several application [149] [150] [148] [47], to the best of our knowledge, its potential in network traffic classification is not fully studied. Consequently, our approach leverages its successful experiences and creates a new model to perform a better classification.

Key Contributions

Based on the aforementioned motivation, in this chapter, we focus on developing an ensemble model using DT-based models and DL for traffic classification. Our model deploys a blending ensemble learning method to combine tree-based classifiers in order to improve the classification performance as well as maximize the generalization accuracy. In brief, the main contributions of our research are the following:

- performance investigation of seven DT-based models with respect to their algorithms (e.g. gradient boosting or bagging);
- improved the generalization performance by combining decisions from several classifiers using the ensemble method;
- boosted the performance of traffic classification with a blending model of DT-based models and the use of the DL model as a meta-classifier;
- performance evaluation of the proposed ensemble using both non-encrypted and encrypted network traffic;
- comparison of the results against, neural networks, ensemble, linear blending, and simple models, as well as some state-of-the-art approaches;

The rest of this chapter is organized as follows. Section 4.1 presents the architecture and the main components of the proposed ensemble model. In Section 4.2 experimental results and performance evaluation of the proposed model are presented. Discussion and analysis of the results are provided in Section 4.3. Finally, the conclusion is given in Section 4.3.

4.1 Proposed Ensemble Learning Model

This section introduces a blending ensemble learning model for network traffic classification. In particular, the architecture of the proposed ensemble model and its design principles are presented. Figure 4.1 summarizes the methodology of this model. For the used data, a *data pre-processing* step is performed. After extensive experiments, we decide to deploy a *blending ensemble learning* to improve the performance and the generalization capability for network traffic classification.

4.1.1 Data pre-processing

As the dataset has different features containing different data types, data pre-processing is an important step for model construction. In other words, data pre-processing aims to

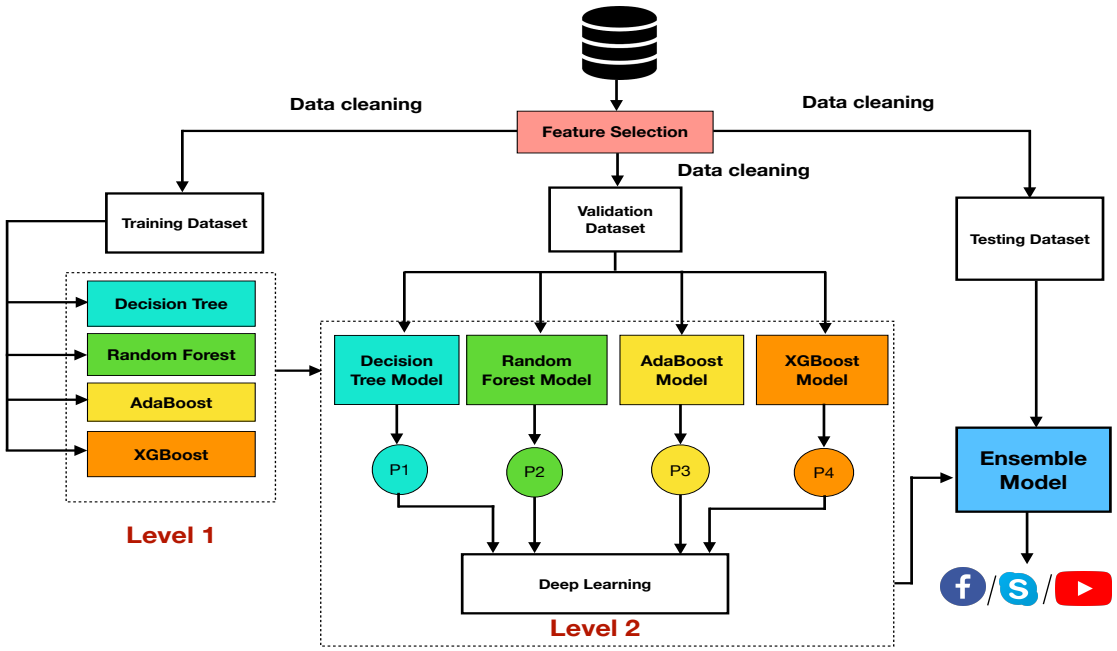


Figure 4.1: Flowchart for the proposed blending ensemble model

transform the data and make it suitable for other processing use. It is a preliminary step that can be done with several techniques among which are data cleaning and feature selection.

- **Data cleaning:** Traffic features can have different types of data, including numerical and categorical values. However, some ML models can only work with numeric values, thus it is necessary to convert the categorical features into numerical ones. Consequently, in this work, we have converted the initial values of some features to numerical values. Also, as the dataset consists of different features with values on different scales, it needs to be scaled. This can be done by Min-Max normalization to perform feature scaling. It is a technique that scales every feature of our dataset between 0 and 1 (Equation 4.1). The Min-Max normalization technique is helpful for the ML model, as it removes any bias from the incoming traffic.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.1}$$

  where X is the feature to be scaled down, X_{max} is the maximum value, and X_{min} is the minimum value of this feature.

- **Feature selection:** Feature selection tries to identify the optimal feature subset and discards the irrelevant and redundant ones. It has various benefits such as i) defying the curse of dimensionality to improve the performance of learning algorithms either

in terms of learning speed and generalization capacity, and ii) reducing the storage requirement. In the proposed model, correlation filtering has been used in order to delete redundant features and reduce the training data complexity. Then to find the optimal features subset a comparative analysis of Information Gain Attribute Evaluation (IG) and Recursive Features Elimination (RFE) has been conducted. We have chosen these methods because they are widely used and perform well in the literature review. Since RFE is a wrapper method, we have used the classification and regression tree (CART) model. Note that these methods are presented in Section 2.6, Chapter 2.

4.1.2 Models hyper-parameters tuning

Data pre-processing and the combination of several classifiers of the ensemble model is not enough, finding the right configuration of the models for both the base and meta classifiers by tuning its hyper-parameters is an important task. These hyper-parameters are the parameters that are not directly set during the training phase because they should not be learned from the training set [151]. The main steps of hyper-parameters tuning are summarized in Algorithm 1. In particular, to find the optimal hyper-parameters, the training of the models using different combinations of the hyper-parameters followed by the performance evaluation of the models on the validation set has been done. This process is repeated until the models achieved a satisfactory performance.

Algorithm 1: Hyper-parameters and parameters tuning

while *Accuracy on the validation set not satisfactory* **do**
    Choose a set of hyper-parameters;
    Given the chosen hyper-parameters train the model and optimize its parameters using the training set;
    Evaluate the model performance on the validation set;
end

4.1.3 Blending ensemble model

The proposed ensemble framework is presented in Figure 4.1. It can be seen that the proposed model mainly consists of two levels, which are base-classifiers using the DT models (level 1) and meta-classifiers using DL (level 2). Level 1 aims to construct the base classifier models through the training set and to produce the meta-data using the validation set. Then, the meta-classifier uses the meta-data (i.e. DL in our case) for the training task and to make the final classification on the test set. More specifically, as presented in Section 2.2.5.3, Chapter 2, the hold-out method is used with the blending model in order to divide the

training set into a new training set and validation set. Next, the base classifiers are trained through the new training set, and their predictions on the validation set are used as meta-data for the meta-classifier. Finally, the meta-classifier uses the meta-data for the training process to make the final decision using the test set.

As presented in the Introduction, DT-based models are one of the most used models for traffic classification as well as since the base classifiers (level 1) need to be accurate, diverse, and complementary as possible in order to provide highly discriminative meta-data for classification [152], a comparative analysis has been done on seven DT-based models with respect to the algorithm on which it is based (e.g. gradient boosting or bagging). In contrast to level 1, level 2 is based on the DL model (i.e. ANN) as a meta-classifier. DL combines the prediction output of the base classifiers, explores the non-linear relationship among them, and hence gets a non-linear blending model. This step is known as the combination phase where the full connected layers are trained on the meta-data generated by the base classifiers. It is important to note that since DL acts as a combination method and its inputs are the prediction of the base classifiers, known as meta-data, we use a simple Neural Network architecture (i.e. three hidden layers).

For more details, Algorithm 2 describes the steps involved in designing a blending-based ensemble model, and Table 4.1 presents the notations used in Algorithm 2.

Algorithm 2: Learning step of the proposed blending algorithm

1: **Input:** Training Dataset $DT = \{x_i, y_i\}$ $(i = 1, 2, ..., n,\ x_i \in X,\ y_i \in Y)$;
2: **Output:** Prediction of Ensemble Algorithm $H(X)$;
3: Use Hold-out validation to divide the training set D_train set & D_val;
   /* --- Train the base classifiers used in the level 1 --- */
4: **for** j=1 to 4 **do in parallel**
5: Train M_j from D_train;
6: **end for**
   /* --- Construct new dataset of prediction --- */
7: D'=$\{P, Y\}$, where P=$M_1(D\_val)$,..., $M_4(D\_val)$;
   /* --- Train the meta-classifier MC --- */
8: Train MC based on meta-data D' (level 2);
9: **return** $H(X) = MC(M_1(X), \ldots, M_4(X))$

4.2 Experimental study and results analysis

In this section, we evaluate the performance of the proposed blending model by performing extensive experiments. Then, the results are analyzed and discussed.

Table 4.1: List of notations used in the ensemble learning algorithm.

List of notations	**Meaning**
X	Set of features
Y	Class label set
M_j	Base classifier
H	Ensemble classifier
n	Number of training samples
D_train	training set of the base classifier (level-1)
D_val	Validation set
P	Base classifier prediction using D_val
D'	meta-data used to train the meta-classifier (level-2)

4.2.1 Dataset description

We evaluate the different classifiers on a real-world traffic dataset[1]. It is a public dataset and has been presented in a research project and collected in a network section from Universidad Del Cauca, Popayán, Colombia [153]. It has been constructed by performing packet captures at different hours, during the morning and afternoon over six days in 2017. We chose this dataset because it can be useful to find many traffic behaviors as it is a real dataset and rich enough in diversity and quantity. It consists of 87 features (attributes), 3,577,296 instances, and 78 classes (Facebook, Google, YouTube, Yahoo, Dropbox, etc.). The distribution of the well-known application is presented in Table 4.2. In this experiment, we have separated the dataset into 80% for training, 10% for validation, and 10% for testing.

It is important to note that this dataset consists of flow and packet-based features for network traffic classification tasks (e.g. flow duration, packet length, port number, etc.). A flow is a set of network packets with the same source/destination IP addresses, source/destination port numbers, and protocol [3].

4.2.2 Experiment setup

`Python 3.7` is used as a programming language and `Scikit-learn 0.23` is used to develop the shallow ML models and `Keras 2.4` framework is used for the DL meta-classifier. *XGBoost*, *LightGBM*, and *Catboost* libraries were used for those models. All experiments were run using a four-core Intel® Core™ i7-6700 CPU@3.40GHz processor, and 32.00 GB of RAM. For reliable evaluation, the reported results are *averaged from ten runs* of each model.

[1]https://www.kaggle.com/jsrojas/ip-network-traffic-flows-labeled-with-87-apps

Table 4.2: Dataset description [153].

Label	**# Total observations**
Google	959,110
HTTP	683,734
Amazon	86,875
Microsoft	54,710
Skype	30,657
Facebook	29,033
Dropbox	25,102
Yahoo	21,268
Twitter	18,259
Apple	7,615
Whatsapp	4,593
Instagram	2,415
Wikipedia	2,025
Netflix	1,560
Spotify	1,269
TeamViewer	527
Telegram	33

4.2.3 Modeling hyper-parameters

The selection of hyper-parameters (also called parameter tuning) is a crucial step in the construction of ML models and the final classification results. These hyper-parameters were set to reach the right trade-off between latency and accuracy. All the hyper-parameters descriptions and settings of the base classifiers/meta-classifier are presented in Table 4.3.

4.2.4 Performance evaluation of the proposed blending model

To evaluate the performance of the proposed model, we will deal with all the following objectives:

- find the appropriate feature set for the proposed model;
- evaluate the classification performance of the DT-based models;
- propose an ensemble model using DL and several DT-based models;
- evaluate the proposed model against the base classifiers;
- evaluate the impact of the hold-out validation set on the ensemble performance;

Table 4.3: Hyper-parameters values of the different classifiers

Model	**Hyper-parameters**	**Values**
Decision Tree	*max_depth*	40
	min_samples_split	40
Random Forest	*max_depth*	50
	n_estimators	50
AdaBoost	*max_depth*	35
	n_estimators	50
	learning_rate	0.4
XGBoost	*max_depth*	35
	learning_rate	0.2
	n_estimators	100
CatBoost	*max_depth*	8
	n_estimators	1000
LightGBM	*max_depth*	35
	boosting_type	goss
	learning_rate	0.2
	num_leaves	1000
	top_rate	0.6
	other_rate	0.4
	n_estimators	250
NN model	*hidden_layer*	3
	activation_function	ReLu
	Learning_rate	0.001
	Optimizer	Nadam

- evaluate the impact of different combinations of base classifiers on the final ensemble;
- evaluate the proposed model against well-known classifiers including boosting, bagging, simple, neural network-based models, linear blending, and state-of-the-art proposed models;
- evaluate the proposed model in terms of training and classification time;
- evaluate the performance of the ensemble using encrypted traffic.

4.2.4.1 Feature Selection

As RF and DT models are the simplest classifiers, they are used to study the impact of the feature selection methods. The obtained results are presented in Table 4.4 and in Table 4.5. First of all, correlation filtering is used in order to delete the redundant features and in turn reduce the processing time for the IG and RFE methods. The reason behind the choice of

this technique is that it is one of the most used methods during the pre-processing step and it is not computationally costly.

Using correlation filtering we have deleted just the redundant features. In other words, we deleted every one of two features that have a correlation $|cor| = 1$, which means that they are totally redundant. This is because, features are usually designed with their unique contributions, and removing any of them may affect the training *accuracy* to some degrees [45]. Then, an evaluation of RF and DT on different reduced versions of the data selected by IG and RFE methods in an individual way has been done. The selected feature subsets contain 10, 15, and 25 features. As shown in Table 4.4, DT and RF models perform better with the 15 features selected through RFE with 82.24% and 85.28% *accuracy*. Also, it can be seen that the performance started to decrease with more features (e.g. 25 features) and this demonstrates that the performance does not always increase with more features.

Table 4.4: Classification accuracy (%) with **RFE** and **IG** on different features set

Classifiers	**10**		**15**		**25**	
	IG	RFE	IG	RFE	IG	RFE
DT	81.38	80.18	81.93	**82.24**	81.58	82.19
RF	85.04	84.69	84.88	**85.28**	84.27	84.91

On the other hand, Table 4.5 indicates that these models can perform better with the reduced features set than with the entire dataset in terms of *accuracy*, *precision*, and *recall*. This may be attributed to the fact that not all features in the dataset can help to separate classes during the classification task. Moreover, this is due to the combination of the wrapper (i.e. RFE) and filter methods (i.e. correlation filtering) that help to find the relevant features.

Table 4.5: The accuracy, precision, and recall (%) of the **entire data** and **selected data** analyzed by DT and RF

Data size	**Accuracy**	**Precision**	**Recall**
Entire dataset + DT	82.16	81.94	82.16
Reduced dataset (RFE) + DT	**82.24**	**82.06**	**82.24**
Entire dataset + RF	82.52	83.13	82.50
Reduced dataset (RFE) + RF	**85.28**	**85.57**	**85.28**

4.2.4.2 Base classifiers selection

As a start, we first randomly partitioned the data into a ratio of 80% for training, 10% for validation, and 10% for testing. It can be seen from Table 4.6 that RF has the best *accuracy*

as a Bagging method with 85.28% *accuracy* whereas the *accuracy* of Extra-Trees is 84.87%. Also, it is more efficient in terms of training and classification time compared to Extra-Trees. On the other hand, the AdaBoost and XGBoost outperform the LightGMB and CatBoost in terms of *accuracy*. They also outperform LightGBM in terms of classification time (test time) and CatBoost in terms of training time.

Table 4.6: Comparison of different methods

Methods	**Models**	**Accuracy (%)**	**Training time (s)**	**Test time (s)**
Single classifier	Decision Tree	82.24	110.21	0.24
Bagging	Random Forest	85.28	193.67	9.63
	Extra Tree	84.87	205.874	150.544
Boosting	AdaBoost	88.51	7921.74	53.44
	XGBoost	88.70	52423.84	197.07
	CatBoost	77.79	231064.01	15.54
	LightGBM	84.57	6292.55	987.50

Based on these results, DT, RF, XGBoost, and AdaBoost are used as base classifiers (i.e. level-1 of the blending). Although these models are based on DT, they work in a different way (e.g. their training process does not use the same training set) and hence guarantee the diversity of level 1 of the blending model. In particular, DT is a simple and classical model, RF is a bagging model that improves the classification performance and builds different versions of the training set by using sampling with replacement technique. Moreover, XGBoost and AdaBoost as boosting models can help to avoid the problem of underfitting (bias). We have chosen XGboost and Adaboost as they perform better than the other boosting models (i.e. LightGBM, CatBoost) and their boosting process is different as explained in Section 2.2.5.2, Chapter 2. Therefore, these make DT, RF, AdaBoost, and XGBoost appropriate models to build our ensemble.

4.2.4.3 Proposed Ensemble classifier

To build the blending models, DT, RF, AdaBoost, and XGBoost are used as base classifiers. Then, DL is used as a meta-classifier to make the final decision. In order to evaluate the performance and the generalization capability of our ensemble against the base classifiers (i.e. without using a meta-classifier), the classification performance using training and test set is measured. As it can be observed in Table 4.7, using the training set AdaBoost gives the best results, followed by RF and XGBoost, our model, and DT.

The *accuracy* achieved by DT, RF, AdaBoost, and XGBoost on the training set is 87.56%,

94.68%, 97.13%, and 92.59%, respectively. Whereas, the classification performance of these models using the test set is different. Indeed, their *accuracy* are 82.24%, 85.28%, 88.51%, and 88.70%, respectively. It is important to note here that the models need to perform very well on the test set (i.e. unseen data during the training). In contrast to DT, RF, AdaBoost, and XGBoost, the difference between classification results using the training and test set of the proposed ensemble model is almost negligible (Table 4.7), where the training and test *accuracy* are **91.57%** and **91.51%**. This demonstrates that the blending ensemble does not have an overfitting problem. This may be attributed to the generalization ability of the blending ensemble. Furthermore, the classification *accuracy* is **2.81%**, **3%**, **6.23%**, **9.27%** better than XGBoost, AdaBoost, RF, and DT, respectively. Similarly, in terms of *F1-score*, the proposed model also outperforms its base classifiers. This is due to that taking advantage of several classifiers, the proposed blending model is shown to be accurate and efficient for traffic classification tasks.

Table 4.7: Statistical measures of the base classifiers and the proposed blending model using training and test sets.

Dataset	**Model**	**Accuracy**	**F1-score**
Training set	DT	87.56	87.36
	RF	94.68	94.59
	AdaBoost	**97.13**	**97.11**
	XGBoost	92.59	92.43
	Proposed model	91.57	91.70
Test set	DT	82.24	81.99
	RF	85.28	84.71
	AdaBoost	88.51	88.16
	XGBoost	88.70	88.47
	Proposed model	**91.51**	**91.64**

- **Impact of the hold-out validation set**

Here the impact of the hold-out validation set on the performance of the blending method as well as the base classifiers are presented. *The hold-out method is a technique used to divide the training set into a new training set and a validation set.* To study the impact of the hold-out validation set, various ratios of the validation set are used. The ratio of the validation set varied between 10% and 40% with a step size of 10% (we stopped because the performance started to decrease). It can be seen from Figure 4.2 that the blending method maintained a better prediction performance than the base classifiers. In particular, when the ratio of the hold-out validation set is 10%, the blending method can obtain the best results. Also, we

can notice that the performance of the base classifiers decreases with more validation sets (i.e. fewer training sets), and this makes the performance of the blending method decreases. It is important to note here that we have used the same test with the different ratios of the hold-out validation set as well as for the rest of the experiment.

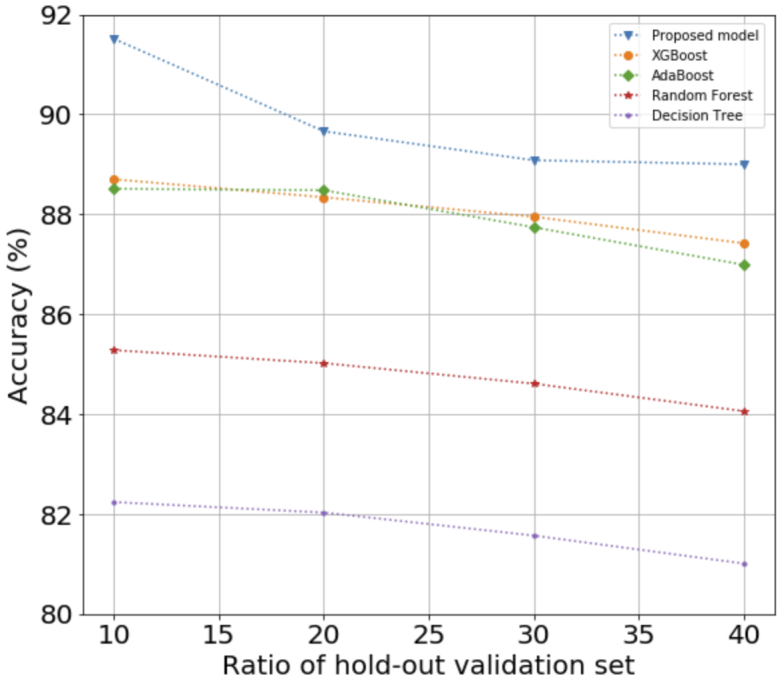


Figure 4.2: Effect of hold-out validation set ratio on the proposed blending model.

- **Impact of the base classifiers**

Additionally, to find the base classifier that can optimize the performance of the proposed ensemble model, some experiments have been conducted. To do so, the performance of the proposed ensemble model with different combinations of base classifiers is presented. Figure 4.3 shows the *accuracy* of different combinations of three base classifiers and the one with all the base classifiers (proposed model). It is important to note here that with all the cases we used DL as a meta-classifier.

As shown in this Figure, the proposed model that integrates all the base classifiers has a better result than the three base classifiers. This means that no model negatively impacts the performance of the proposed ensemble. Moreover, as still shown in this figure, by dropping out one of the boosting models (AdaBoost and XGBoost) the *accuracy* of our ensemble decreases. Specifically, XGBoost plays the most critical role in achieving good performance. The accuracy of the blending decreases notably without XGBoost, which means that this model is an important component of the blending model.

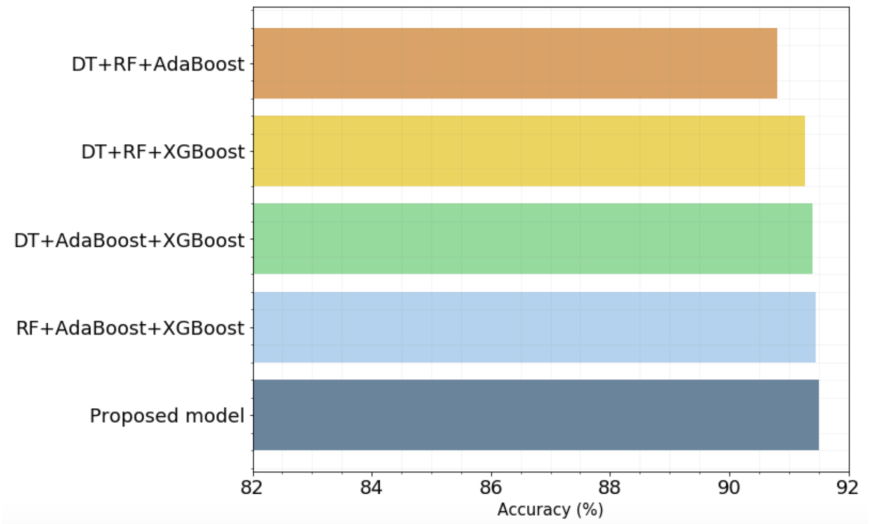


Figure 4.3: Results of the different combinations of the base classifier experiments and their impact on the proposed blending model.

- **Comparative analysis over various ML algorithms**

In order to further validate the efficiency of the proposed ensemble, which demonstrated the best performance against the base classifiers, we compared this framework with other models. The optimal hyper-parameters of these models have been used. These models include (i) **simple models** such as SVM and KNN, (ii) **ensemble models** such as Extra-Trees, LightBGM, and CatBoost, (iii) **neural network model**, which is Multi-layer Perceptron (MLP) classifier. We select these classifiers as our baselines because Extra-Trees and KNN are easy to train, SVM is widely used and proved to be useful in several applications [154], and MLP is a neural network model as well as CatBoost and LightBGM because they are recent models for the classification task.

Figure 4.4 shows the accuracy of the proposed ensemble against other models. In this case, a clear hierarchy of models emerges from best to worst. As can be observed from the results that the proposed model performs well when compared with the other models. Its accuracy is **6.64%**, **6.94%**, **13.72%**, **16.03%**, **17.85%**, **44.75%** better than Extra-Trees, LightBGM, CatBoost, MLP, KNN, and SVM, respectively. This may be attributed to the fact that the combination of DT-based models and DL helps the proposed ensemble to yield far superior results compared to several models.

- **Comparative analysis over linear blending**

In this section, a comparison of the proposed model with a linear blending model is presented. For the linear blending, a logistic regression model is used as a meta-classifier instead of a DL model. Put another way, the linear blending uses the same base classifiers

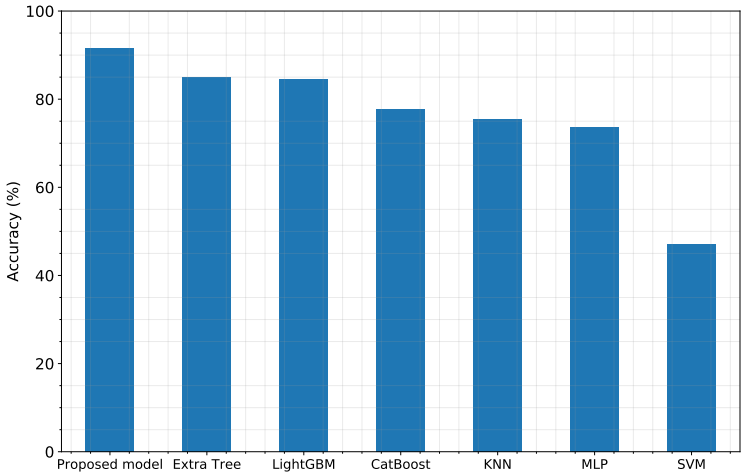


Figure 4.4: Comprehensive comparison of well-known classifiers against the proposed blending model.

as our ensemble and changed only the meta-classifier. The objective of this comparison is to study the impact of DL on the proposed ensemble. Figure 4.5 shows that our ensemble (non-linear blending) outperforms the linear blending by **2.25%**, **2.59%** in terms of *accuracy* and *F1-score*, respectively.

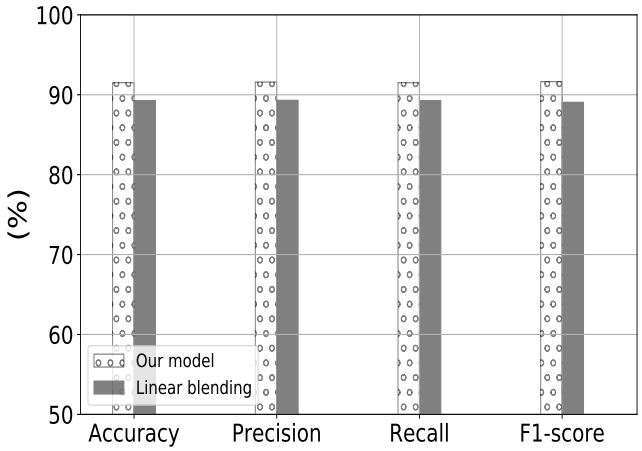


Figure 4.5: Our model vs. linear blending.

Also, Table 4.8 presents the achieved performance of both linear and our ensemble (i.e. non-linear blending) for the application identification task. It can be seen that the *precision* of the Linear blending is high which is not the case with the *recall*. This means that the false positive is few and the false negative is high. On the other hand, the *F1-score* of our

ensemble outperforms that of linear blending for all the applications. This is attributed to the fact that the harmonic mean of *precision* and the *recall* is better than the linear blending. Moreover, we can notice that the linear blending could not detect the minor class like the Telegram application, whereas our ensemble achieved a **100%** *F1-score*. This is may be attributed to the use of DL as a meta-classifier in our ensemble model.

Table 4.8: Linear blending vs. our ensemble for application classification

Class Name	**Linear Blending**			**Our ensemble**		
	Precision (%)	Recall (%)	F1-score (%)	Precision (%)	Recall (%)	F1-score (%)
Google	85.3	**93.43**	89.18	**95.11**	86.93	**90.83**
HTTP	93.65	93.26	93.45	**94.45**	**94.83**	**94.64**
Amazon	**98.52**	93.11	95.74	94.94	**98.84**	**96.85**
Microsoft	**90.24**	81.93	85.88	89.82	**93.35**	**91.55**
Skype	91.49	80.36	85.57	**92.55**	**92.15**	**92.35**
Facebook	**97.89**	91.26	94.46	94.27	**98.52**	**96.35**
Dropbox	**98.61**	92.62	95.52	96.84	**98.1**	**97.47**
Yahoo	**90.87**	77.46	83.63	86.79	**93.59**	**90.06**
Twitter	**94.24**	73.86	82.81	84.81	**94.81**	**89.53**
Apple	93.80	82.45	87.76	**94.11**	**95.65**	**94.87**
Whatsapp	**94.57**	74.2	83.15	94.5	**94.7**	**94.6**
Instagram	**94.76**	79.74	86.6	93.24	**97.78**	**95.46**
Wikipedia	**99.32**	71.71	83.29	90.41	**98.01**	**94.06**
Netflix	**98.99**	69.01	81.33	82.91	**97.76**	**89.72**
Spotify	**96.3**	80	87.39	88.63	**96.69**	**92.49**
TeamViewer	**100**	87.3	93.22	96.82	**100**	**98.38**
Telegram	0	0	0	**100**	**100**	**100**

- **Computational costs of the proposed model**

Moreover, we have compared the computational efficiency of the proposed model against other models. As shown in Table 4.9, we have compared the classification time (CT) per sample and the training time (TT) taken by the respective models.

As explained in Section 4.1.3, our model consists of two main phases (level 1, and level 2 process). Thus, this can explain its high training time. Even though it has a high training time, there is a slight difference with XGBoost this is because one of the advantages of our model is that the base classifiers: DT, RF, AdaBoost, and XGBoost can be trained in parallel. Consequently, since boosting models take a long time to train (Table 4.7), thus they can impact the training time of the proposed model more than RF and DT. However, we assume that the training time can be proceeded offline and thus does not impact the real-time utilization of the classification process.

Table 4.9: Training and classification time comparison

Metric	**XGBoost**	**AdaBoost**	**Linear blending**	**Our model**
TT (s)	52423	7921	52523	53100
CT_per_sample (µs)	550.89	149.38	552.67	582.7

4.2.5 Experiments on the second dataset (VPN-nonVPN dataset)

To validate the effectiveness of the proposed ensemble, we also simulated experiments based on another public dataset, which includes encrypted data, called *VPN-nonVPN dataset* [114]. This is one of the most popular encrypted traffic classification datasets. It contains only time-related features (flow duration, flow inter-arrival time, etc.) and 14 classes. These classes are Browsing, Chat, Email, FTP, P2P, Streaming, VoIP, vpnBrowsing, vpnChat, vpnEmail, vpnFTP, vpnP2P, vpnStreaming, and vpnVoIP (Table 4.10).

Table 4.10: VPN-nonVPN dataset description.

Label	**# Total observations**
VoIP	2,826
Browsing	2,500
File Transfer (FT)	1,018
P2P	1,000
Chat	890
Streaming	482
Mail	249
VPN-VoIP	2,271
VPN-Browsing	2,500
VPN-FT	1,932
VPN-P2P	928
VPN-Chat	1,196
VPN-Streaming	475
VPN-Mail	491

In order to evaluate the performance of the proposed ensemble, we used two scenarios: Scenario A and Scenario D. As shown in Table 4.11, Scenario A, **(Sc_A)**, is a binary classification to indicate whether the traffic flow is VPN or not. Both scenario B, **(Sc_B)**, and scenario C, **(Sc_C)**, are 7-classification tasks. Scenario B is to distinguish between seven non-VPN traffic services like audio, browsing, etc. Scenario C is similar to Scenario B, while its target labels are seven traffic services of the VPN version Scenario D, **(Sc_D)**, mixes all fourteen applications to perform the 14-classification task. To find the most relevant features in the two scenarios, we have used RFE also as a feature selection method in order to find the optimal subset. Then, using those features, we have compared the performance of our

model against several ML-based classifiers as well as against the linear-blending model in the **(Sc_A)** and **(Sc_D)**.

Table 4.11: Scenario description.

Scenario	**Description**
Sc_A	VPN and Non-VPN traffic identification (2-class)
Sc_B	Regular non-VPN traffic classification 7-class
Sc_C	VPN traffic classification 7-class
Sc_D	All traffic classification 14-class

It can be seen from Table 4.12 that our ensemble achieves the best results with the VPN-nonVPN dataset in the two scenarios. This demonstrates that our model can achieve high performance using only time-related features. Specifically, for example, in Scenario A, the *accuracy* of our model is **9.69%**, **14.67%**, **36.22%**, **28.21%**, **7.12%**, **6.85%**, **5.89%**, **5.28%**, **6.88%**, **5.02%** better than DT, KNN, SVM, MLP, Extra-Tree, RF, CatBoost, LightGBM, AdaBoost, XGBoost, respectively. Also, in scenario D, the accuracy of our model is **18.02%**, **24.14%**, **52.66%**, **42.22%**, **12.73%**, **12.66%**, **11.65%**, **9.95%**, **13.21%**, **10.15%**, better than DT, KNN, SVM, MLP, Extra-Tree, RF, CatBoost, LightGBM, AdaBoost, XGBoost, respectively. These results illustrate the high performance of our model with both binary and multi-classification scenarios.

Table 4.12: The classification accuracy (%) of baseline and ensemble methods on VPN-nonVPN Dataset.

Model	**Sc_A**	**Sc_D**
DT	87.85	78.83
KNN	82.87	72.71
SVM	61.32	44.19
MLP	69.33	54.63
Extra-Tree	90.42	84.12
RF	90.69	84.19
CatBoost	91.65	85.20
LightGBM	92.26	86.90
AdaBoost	90.66	83.64
XGBoost	92.52	86.70
Linear blending	95.48	93.88
Proposed model	**97.54**	**96.85**

Similar to the first dataset, we compared the proposed ensemble with the linear blending model using the VPN-nonVPN dataset. Figure 4.6 shows that our model outperforms the linear blending on all metrics, similar to the first dataset.

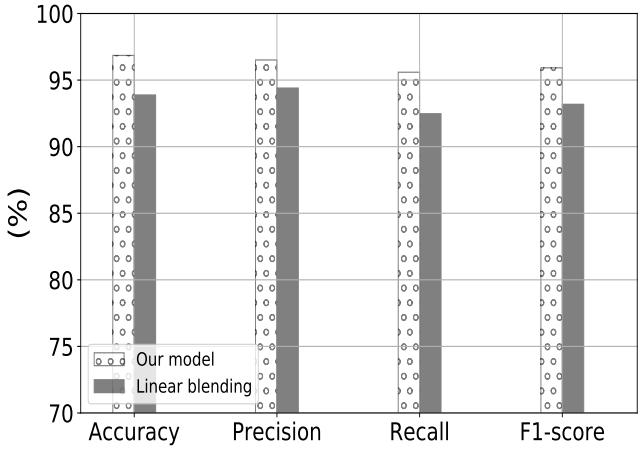


Figure 4.6: Our model vs. linear blending

Also, to better evaluate the performance of our model, the fourteen classification tasks of the VPN-nonVPN dataset are reported in Table 4.13. From this Table, it can be seen that our model, achieves the *F1-score* up to 91% with almost all the classes. Also, our model has a better *F1-score* than linear blending in 11 out of 14 cases. Consequently, from those results, we can conclude that our model is a promising model and can better differentiate the applications for encrypted traffic.

4.2.6 Performance against state-of-the-art models

Finally, using VPN-nonVPN dataset we compared our ensemble with some state-of-art approaches including [114], [155], and [102] to validate the performance of our ensemble model. [114] is the original dataset paper, where the authors used a Decision Tree and KNN classifiers in order to characterize network traffic. However, since the Decision Tree performed a little better, we use its result for the comparison task. In contrast, the authors in [155] and [102] proposed DL-based ensemble approaches. More specifically, Yao et al. [155] used the Attention Based LSTM model in order to improve the classification performance. Then, Lin et al. [102] combined CNN and LSTM models to extract the features and realize the traffic classification.

From the simulation results in Table 4.14, we can observe that the blending scheme (linear/non-linear) achieves competitive *accuracy* in comparison to the state-of-art methods. In particular, we can see that our model outperforms the proposed approach in [114] for both scenarios (Sc_A and Sc_D), however, it is not the case with [155] (Attention-LSTM) and [102] (CNN-LSTM) for the Sc_A (binary classification). For Sc_D (multi-classification) we can see that the blending scheme and especially our ensemble (non-linear blending)

Table 4.13: Linear blending vs. our model using VPN-nonVPN dataset

Class Name	Linear Blending			Our model		
	Precision (%)	Recall (%)	F1-score (%)	Precision (%)	Recall (%)	F1-score (%)
Browsing	92.73	**98.08**	95.33	**98.91**	97.15	**98.02**
Chat	95.24	90.91	93.02	**97.50**	**95.12**	**96.29**
FT	**95.45**	87.50	91.30	85.71	**98.82**	**91.80**
Mail	**100**	**100**	**100**	92.30	**100**	96.00
P2P	93.75	**100**	96.77	**100**	95.91	**97.91**
Streaming	**100**	75.00	85.71	87.17	**97.14**	**91.89**
VOIP	**100**	98.48	99.24	99.62	**100**	**99.81**
VPN-Browsing	91.67	89.19	90.41	**97.39**	**97.39**	**97.39**
VPN-Chat	83.33	86.96	85.11	**90.32**	**97.39**	**93.72**
VPN-FT	90.70	90.70	90.70	**96.27**	**99.04**	**97.64**
VPN-Mail	**100**	83.33	90.91	**100**	**93.33**	**96.55**
VPN-P2P	80.95	**94.44**	**87.18**	**95.32**	80.31	87.17
VPN-Streaming	**100**	**100**	**100**	97.82	**100**	98.90
VPN-VOIP	97.73	**100**	98.85	**100**	99.54	**99.77**

outperforms the [114], [155], [102] by **15.08% (12.11%)**, **5.65% (2.68%)**, and **5.15% (2.18%)**, respectively. This demonstrates that with a more complicated classification task (multi-classification) our model performs well. This is because the use of a meta-classifier corrects the errors that occur during the learning process of the base classifiers.

Table 4.14: The classification accuracy (%) of baseline and ensemble methods on VPN-nonVPN Dataset.

Schemes	Sc_A	Sc_D
DT [114]	89.7	81.77
Attention-LSTM [155]	**99.7**	91.2
CNN-LSTM [102]	**99.7**	91.7
Linear blending (ours)	95.48	**93.88**
Non-linear blending (ours)	97.51	**96.85**

4.3 Discussion

In this chapter, a novel classifier model is proposed to explore the potential of ensemble learning and improve the performance of the DT-based models. Using two datasets and different scenarios, the results demonstrate that by taking advantage of several classifiers, our

ensemble model is shown to be accurate and efficient for traffic classification tasks. Specifically, the use of a meta-classifier corrects the errors that occur during the learning process of the base classifiers. Moreover, it prevents overfitting and reduces bias simultaneously to some extent (Table 4.7). The presented results also confirm that the neural network layers help to discover the nonlinear relationships with very little hand engineering and increase the learning ability of the whole ensemble model.

To evaluate the effectiveness of the proposed model, we have conducted experiments on different ensemble hyper-parameters (e.g. validation set, base classifiers) and comparative analysis with the base classifiers and other well-known classifiers. Several evaluation metrics are used to evaluate the performance of the proposed ensemble in terms of *accuracy*, *F1-score*, and computation cost. Based on the extensive study of our blending model, we have shown that the boosting models, XGBoost and AdaBoost, obtained higher prediction *accuracy* than those of the bagging methods (e.g. RF and Extra-Trees) as well as the single classifier (e.g. DT, KNN). Also, we have noticed that using the data generated by the base classifiers, the meta-classifier outperforms the MLP model that learns from the initial data. This is because the data generated by the base classifiers help the meta-classifier to converge quickly and very well. In addition, the performance of the ensemble is depending on the base classifiers and hence the choice of the base classifiers is a crucial task. Finally, our study demonstrates that no model negatively impacts the performance of the proposed ensemble.

Pros and cons of proposed ensemble

Our proposed model has several advantages:

+ achieves higher classification and generalization performance than its base classifiers;
+ supports parallel training since the base classifiers are independent of each other;
+ supports the application-aware training where the base classifiers can be used or trained according to the specific needs;
+ works very well with different situations, including binary and multi-classification tasks as well as encrypted and non-encrypted traffic.

The shortcoming of the proposed model is that:

− takes more training time than the base classifiers;
− is still difficult to decide which classifiers should be used in level-1 of the ensemble.

Conclusion

In this chapter, we presented a novel ensemble model based on the DL and DT-based models for traffic classification. The proposed model is based on three main steps. The first, data preprocessing, includes feature selection using two feature selection methods and data cleaning. In order to find the optimal features subset, we have used correlation filtering to pick up and delete the redundant features and in turn, reduce the processing time. Then, a comparative analysis of two well-used feature selection methods, IG and RFE, has been presented. The results demonstrate that selecting almost 18% of the dataset through a features selection method (RFE) improves the classification performance of the DT and RF by 0.15% and 2.97%, respectively. Also, it reduces the training data size by finding useful subset features. In the second step, to select the base classifiers, we conducted an empirical analysis of seven DT-based models in terms of accuracy, generalization capability, training, and classification time. Next, in the third and final step, an ensemble model that incorporates four DT-based models and DL is applied to improve overall classification accuracy. By using DL as a meta-classifier, the relationships among the base classifiers are learned automatically, thus enabling the ensemble method to achieve better classification performance.

Using two datasets, the simulation results show that the proposed ensemble model outperforms other shallow ML models (DT, SVM, KNN) and ensemble learning models (e.g. RF, MLP), the linear blending model (logistic regression as a meta-classifier) as well as some existing state-of-the-art approaches. Moreover, we have studied the impact of the base classifiers and the hold-out validation set ratio on the performance of the whole model.

For future work, we will investigate further the performance of the proposed ensemble model in another context (e.g. intrusion detection). Although, the performance of our ensemble, finding totally labeled data is almost impossible. Thus, to be more realistic, in the next chapter, we will use a semi-supervised learning model in order to take advantage of a few labeled traffic and the abundance of unlabeled data.

*****

Chapter 5

Handling partially labeled network data: a semi-supervised approach using stacked sparse autoencoder

Introduction

Machine Learning (ML) is opening the ways to develop network traffic classifiers, which achieve an acceptable trade-off between computation complexity and accuracy [90]. Most of the classifiers are based on supervised learning where only labeled data are used as well as their learning process requires a large volume of labeled data. However, under the explosion of new traffic and applications, it is very difficult if not impossible to collect sufficient labeled samples for all existing applications. At the same time, labeling all the traffic requires a huge effort of human annotators sometimes with a specific domain of expertise. On the other hand, since the unlabeled data provide informative characteristics, they could improve the performance of the supervised learning algorithms [24]. Therefore, semi-supervised learning that uses a large amount of unlabeled data together with a limited amount of labeled data is a promising solution and has attracted more and more attention in network traffic classification [108].

Apart from the previous issue, building models using shallow models is also bottlenecked by the amount of features engineering effort required since there are limits to how much human effort can be thrown at the problem as well [50]. In this regard, DL has gained popularity in the machine learning community because of its unique nature for solving complex problems, and it outperforms the other shallow ML models in several fields such as healthcare, computer vision, and network resource management, and has shown success in network traffic classification [5]. DL provides a variety of algorithms that allow exploiting unlabeled data to learn useful patterns in an unsupervised manner; for example, an AutoEncoder (AE) is one of the most popular and most widely used models for feature extraction [156]. Specifically, AE has different variations and its structures are dependent on the number of layers [74]. The simple model has just one hidden layer, which is not able to get the discriminative features. Thus, to obtain a better performance and learn more complex and abstract features than classical AE, more complex architecture and training procedures have been proposed. This model is known as Stacked AE (SAE) [75] (Chapter 2, Section 2.4.2.2). In fact, SAE has many advantages such as conducting learning based on unlabeled data and benefiting from its abundance. Also, learning the features from unlabeled data automatically in advance, which is called pre-training, is much better than learning them from hand-crafted features [157] [158].

Motivations and key challenges

Applying semi-supervised learning and DL for traffic classification is a promising solution, but it is accompanied by key challenges, which are listed below:

- extracted features for the traffic classification should distinguish applications from each other as much as possible;

- finding the representative features using only a limited amount of labeled data (with the help of numerous unlabeled data) should be done automatically;
- finding DL hyper-parameters that should learn robust features and classify new network traffic very well.

Key contributions

Facing the above challenges, in this chapter, we present a semi-supervised model with a deeper investigation into the variable ratios of unlabeled data as well as their impact on *accuracy*. As presented in Chapter 3, section 3.1.3, even a few semi-supervised models and SAE-based models have been used in traffic classification; they did not study the impact of the unlabeled ratio on the performance of the final model, nor the effect of hyper-parameters (e.g. dropout) on the generalization performance of the SAE model nor a comparative analysis with several shallow ML and DL models, which have been provided in this study. In particular, during this contribution, extensive experiments have been conducted to obtain a robust model, which is then compared against representative models using supervised approaches as well as DL-based approaches. In brief, the contribution of this chapter can be summarized as follows:

- a robust SAE model based on both unlabeled and labeled traffic has been proposed;
- an introduction of the sparse, dropout, and denoising coding hyper-parameters are injected into the model to avoid the over-fitting problem and extract robust features;
- a performance evaluation and comparison against semi-supervised learning (e.g. AE model), as well as supervised learning (well-known supervised models), have been conducted;
- a performance evaluation of the proposed model using both non-encrypted and encrypted network traffic.

The rest of this chapter is organized as follows. Section 5.1 presents the architecture and the main components of the proposed semi-supervised model. In Section 5.2 experimental results and performance evaluation of the proposed model are presented. Discussion and analysis of the results are provided in Section 5.3. Finally, the conclusion is given in Section 5.3.

5.1 SAE-based semi-supervised model

In this section, we describe our proposed methodology: an SAE-based semi-supervised method for traffic classification. Figure 5.1 presents the structure of the proposed model. It

can be seen that the proposed model consists of an *unsupervised feature extraction task* and a *supervised learning task*. We present the methodology in the following section.

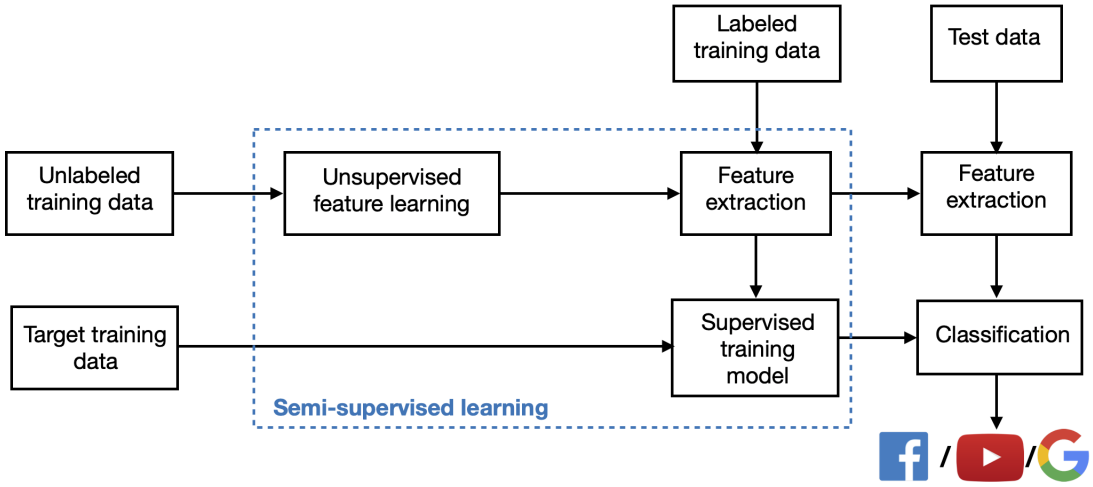


Figure 5.1: Structure of the semi-supervised network traffic classification model

5.1.1 SAE model

By taking advantage of *unlabeled* and *labeled* data, a semi-supervised classification model has been proposed as illustrated in Figure 5.1. The proposed semi-supervised classification model consists of (i) the unsupervised feature extraction stage using unlabeled data, and (ii) the supervised learning using labeled data. To obtain a better performance and learn more complex and abstract features than the classical AE model, we deploy a more complex architecture and training procedure, known as Stacked AutoEncoder (SAE). As presented in Section 2.4.2.2 and as shown in Figures 5.2, with SAE, several AE layers are stacked together and form an unsupervised pre-training stage where the encoder layer computed by an AE will be used as the input to its next AE layer. Each layer in this stage is trained like an AE by minimizing its reconstructing error. When all the layers are pre-trained, the network goes into the supervised fine-tuning stage.

In particular, using the unlabeled data, the unsupervised learning algorithm is pre-trained in a bottom-up way. Then, the decoder layers of the SAE model have been ignored and then we directly linked the last hidden layer (i.e. code) to a neural network classifier (i.e. Softmax layer); hence, we get a new deep-learning model. Next, using the few labeled data, a fine-tuning process was done in a top-down fashion by training the pre-trained layers as a single model. Finally, the backpropagation algorithm is employed to get the gradient to update the parameters of the whole model. This learning process makes the proposed model takes advantage of both labeled and unlabeled data. In fact, the pre-training helps the deep neural network models to yield much better results with local initialization than random

initialization. Also, the global fine-tuning process optimizes the parameters of the entire model, which greatly improves the classification task. Moreover, in order to extract more robust features and prevent the over-fitting problem during the training process, we injected other hyper-parameters such as denoising coding, sparse, and dropout (Chapter 2).

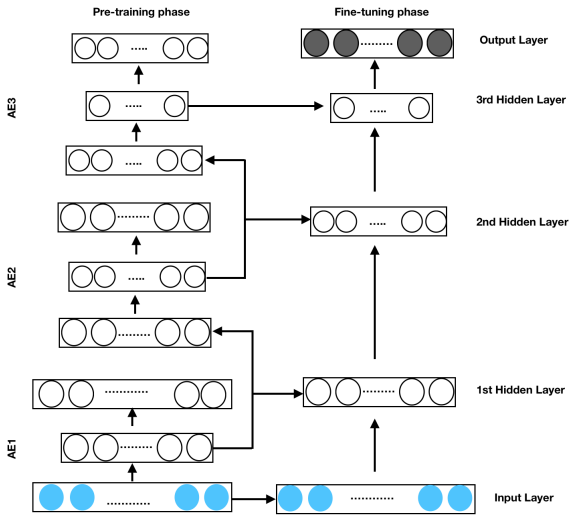


Figure 5.2: General Stacked Autoencoder process

5.1.2 Data pre-processing

Network datasets can have different types of data, including numerical and categorical values. Therefore, it is important to pre-process this traffic in order to build the proposed model. Moreover, when the dataset consists of different features with values on different scales, it needs to be scaled. As with the previous chapter, data cleaning and feature normalization have been done. Then, we separated the data into unlabeled and labeled sets, where the labeled set is split into training (80%), validation (10%), and testing (10%). It is important to note that during this experiment, we fix the amount of labeled data and vary only the amount of unlabeled data.

5.2 Experimental study and results analysis

In this section, we evaluate the performance of the proposed SAE model by performing extensive experiments. Then, the results are analyzed and discussed. `Python 3.7` is used as a programming language and `Scikit-learn 0.23` is used to develop the shallow ML models and `Keras 2.4` framework is used for the DL-based models. For reliable evaluation, the reported results are *averaged from only five runs* of each model (because SAE is expansive in terms of training time).

5.2.1 Objectives

To evaluate the performance of the proposed model, we will deal with all the following objectives:

1. Study the impact of some hyperparameters on the proposed model:
    - evaluate the performance of the SAE under different ratios of unlabeled data;
    - evaluate the impact of the sparse parameter on the SAE performance;
    - evaluate the impact of dropout and corruption noise hyper-parameters on the SAE performance.
2. Compare the proposed model against other well-known models:
    - evaluate the impact of features extracted by the SAE on the performance of other learning models;
    - evaluate SAE against well-known models including simple AE (deep-learning) and supervised models that use 100% labeled data;
    - evaluate SAE against supervised SAE using the same architecture and hyper-parameters;
    - evaluate the classification performance on the well-known applications through a confusion matrix;
    - evaluate the proposed model in terms of training and classification time;
    - evaluate and position our approach in the literature using one of the most popular traffic datasets, called (ISCX VPN-nonVPN 2016).

5.2.2 Dataset description

To evaluate the performance of the proposed semi-supervised models, we have used the first dataset of the previous contribution[1]. However, for facilitating computation, we have used only the traffic collected from one day, which is 09/05/2017. Therefore, our sub-dataset consists of 504,731 instances and 54 applications. The simulation of a partially-labeled dataset has been done through the random selection of a portion of the known applications and removing the application labels of their instances. Table 5.1 presents the statistical information (unlabeled/labeled observations along with train/validation/test split) used in this work.

[1] https://www.kaggle.com/jsrojas/ip-network-traffic-flows-labeled-with-87-apps

Table 5.1: The numerical information of dataset.

Unlabeled (#samples)	Labeled		
	Train	Validation	Test
384,366	96,293	12,036	12,036
364,155	96,293	12,036	12,036
283,217	96,293	12,036	12,036
202,322	96,293	12,036	12,036
161,868	96,293	12,036	12,036
40,484	96,293	12,036	12,036
20,259	96,293	12,036	12,036

5.2.3 SAE-based semi-supervised architecture and hyperparameters

Here, we study the impact of several hyper-parameters on the performance of the whole model. In the beginning, since in a deep neural network, there is no clear mathematical proof to interpret its architecture. Therefore, to find the optimal model, we tested different architectures as well as hyper-parameters that maximize the *accuracy* of the classification. The configuration, with 4 hidden layers **[100, 200, 400, 50]** is selected for further experiments since it provides the best results. Also, it is important to note that, after extensive simulations, we have selected a learning rate equal to 0.0001. Besides, the selected activation of hidden layers is ReLU (rectified linear unit) because it provides better convergence performance than sigmoid and tanh [51].

5.2.3.1 Trade-off between performance and unlabeled ratio

One of the important concerns in this contribution is to study the impact of the unlabeled data ratio on the performance of the proposed model. Therefore, in this section, we explore in-depth the trade-off between the performance of the proposed model and the amount of unlabeled data. To do so, we trained our system using different ratios of unlabeled samples called R_u, which is expressed below.

$$R_u = \frac{nb\ unlabeled\ data}{nb\ labeled\ data} \tag{5.1}$$

The *accuracy* of the model while varying R_u is presented in Figure 5.3. It can be seen that increasing the amount of unlabeled data (increasing R_u) boosts the classification performance of the model. This can be explained by the fact that increasing the amount of unlabeled data

provides more informative characteristics and SAE can benefit from this data in the pre-training process and in turn, improves the classification performance for unseen observations. As a result, the R_u=3.2 has been used for the rest of the experiments.

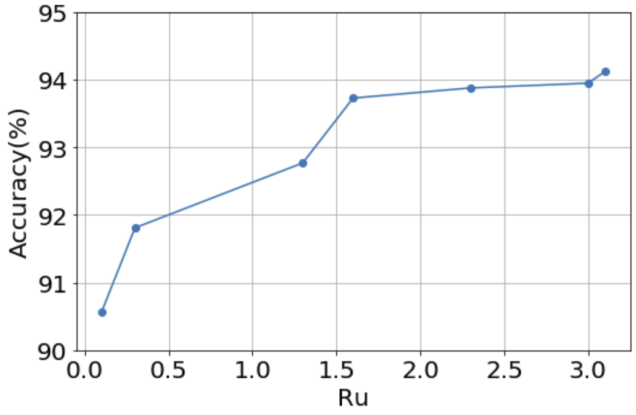


Figure 5.3: Performance of model with different unlabeled ratios.

5.2.3.2 Impact of the sparse hyper-parameter

Since the number of neurons in hidden layers is large (i.e. [100, 200, 400, 50]), using a sparse constraint can allow discovering better the complex structure behind the data. However, like all the hyper-parameters, it is crucial to select an optimal sparsity parameter (Chapter 2, Section 2.5) for better traffic classification. As presented in Figure 5.4, we have tested the effect of the sparse parameter on the performance of our model. Here, the rate varied between 0.01 and 0.07 (we stopped when the performance started to decrease). It can be seen that when the value of the sparse parameter is 0.06, the SAE model gives the best *accuracy* (94.40%) and training time. Larger than this value, the training time of the model begins to increase. Using the sparse hyper-parameters, the SAE is converted into **Stacked Sparse AutoEncoder (SSAE).**

5.2.3.3 Impact of dropout and denoising hyper-parameters

Although `SSAE` performs well, we can further improve its generalization performance through other hyper-parameters. Using the dropout (Chapter 2, Section 2.5) and denoising (Chapter 2, Section 2.4.2.2) hyper-parameters can improve the performance of the classification. As presented in Figure 5.5 and Figure 5.6, we have tested the impact of dropout and corruption noise on the accuracy of our model. Here, the rate varied between 0 and 0.05 (we stopped when the performance started to decrease). The results show that the best classification performance was obtained at a dropout rate and corruption both equal to 0.02. Based on the results shown in these figures, it can be seen that the `SSAE` has the ability to

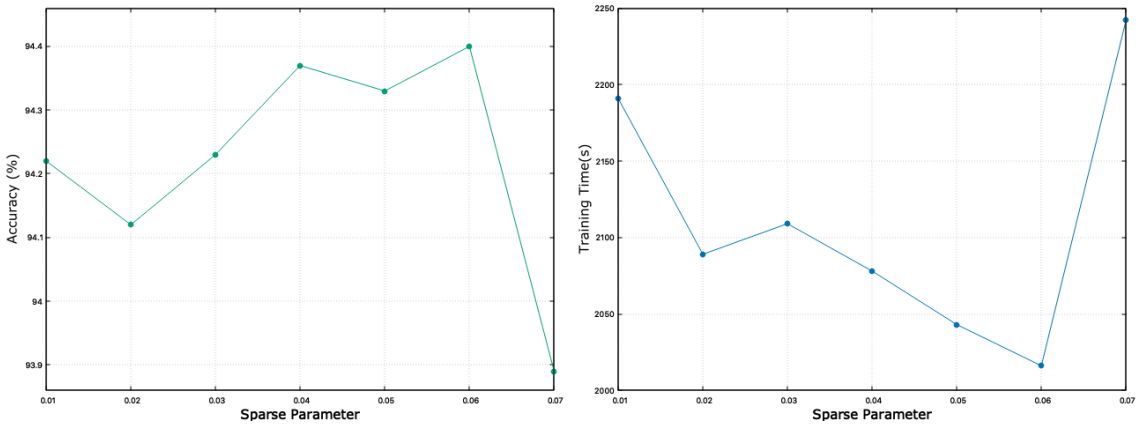


Figure 5.4: Accuracy and training time of different sparse parameter

restore a good reconstruction from a corrupted input version even with a high corruption level (Figure 5.6). Moreover, we can interpret that too much a dropout rate can decrease the classification performance.

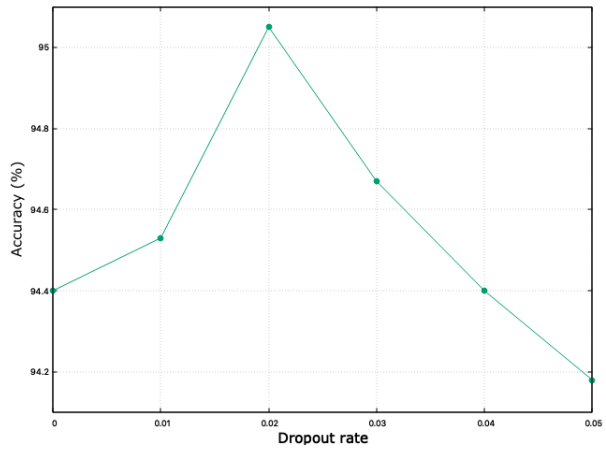


Figure 5.5: Effect of dropout

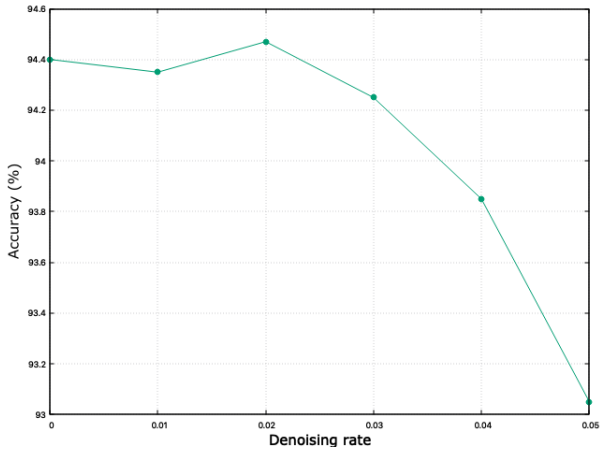


Figure 5.6: Effect of denoising coding

Furthermore, to verify the impact of these hyper-parameters on the generalization performance of the proposed model, we compare the combined solution against a simple `SSAE` (without enhancement). The results are shown in Figure 5.7. It can be seen that the `SSAE` with dropout and denoising code combined has shown a better performance (i.e. *test accuracy*) with 95.03% *accuracy*. In addition, the generalization capability of the model has been improved (i.e. the difference between the training and testing *accuracy* has been reduced). This can be explained by the fact that the denoising rate can help to extract robust features and the dropout prevents the co-adaptation between the hidden neurons and hence avoids over-fitting.

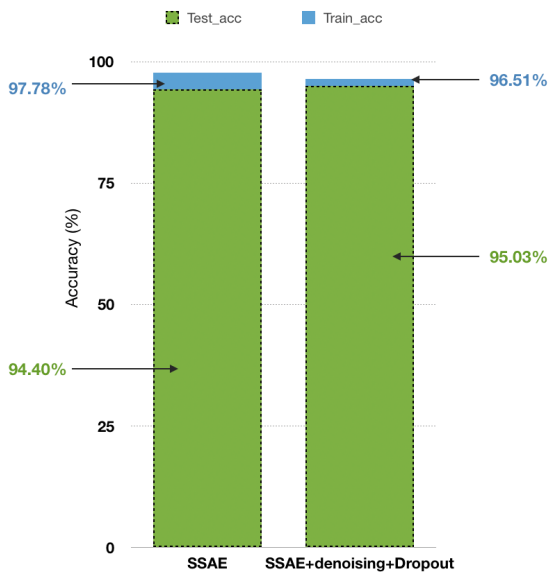


Figure 5.7: Accuracy of our model without and with enforcement (dropout and denoising)

5.2.4 Comparison Analysis

To evaluate the performance of the proposed model, we perform a comparative analysis against other shallow ML and DL models including the following two categories: semi-supervised and supervised models.

5.2.4.1 Comparison with semi-supervised learning models

To verify the classification efficiency of the proposed model labeled as `SSAE*` (i.e. `SSAE` with denoising and dropout), we compared it to four reference ML classification algorithms, namely DT, RF, SVM, XGBoost as well as simple `SSAE` (without dropout+denoising). To create the semi-supervised models *(SSAE+DT, SSAE+RF, SSAE+SVM, SSAE+XGBoost)*, these algorithms are built on top of the unsupervised part of our proposed model (feature extraction part) instead of Softmax layer. They try to benefit from the automatic feature extraction of the pre-trained part of the model. In fact, the labeled data is passed through the pre-trained of the `SSAE` that is trained by the unlabeled data and obtains X', the transformed data. The last layer of our model (the encoding vector) has only 50 neurons, which is a smaller dimension than X (i.e. 87 features). Finally, these features are used with the aforementioned classifiers.

Moreover, we also used the AE model for comparison as it has a deep-learning architecture similar to `SSAE`. After learning of the AE (i.e. unsupervised learning), the decoder is removed and a Softmax layer is attached and the whole model is fine-tuned for the classification task (i.e. supervised learning). As shown in Figure 5.8, the AE here reconstructs the input and

then classifies it through the encoder part. It should be noted that the used AE has the same network structures as `SSAE` except that there is no sparse constraint as well as no noise and dropout hyper-parameters.

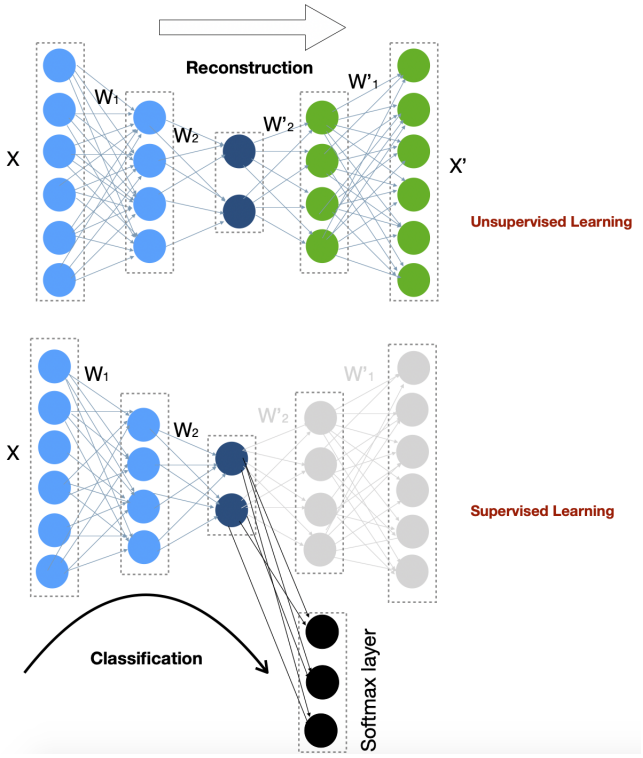


Figure 5.8: Classification process with AE model

Table 5.2 summarizes the experimental results of the five semi-supervised models (previously described), `SSAE`, and `SSAE*`. It can be seen that our proposed model outperforms each of them. In fact, DT, RF, XGBoost, and SVM cannot fine-tune the features extracted by the SSAE and this may explain their lower performance. Also, using the test dataset, it can be seen that DL-based models (i.e. `SSAE` and AE) perform better than the shallow models (i.e. RF, DT, XGBoost, and SVM). For example, the *accuracy* of our model is **2.45%**, **3.33%**, **4.96%**, **38.84%**, **6.14%** better than AE, SSAE+XGBoost, SSAE+RF, SSAE+SVM, SSAE+DT, respectively. Furthermore, the results clearly demonstrate that the `SSAE` models with/without denoising and dropout hyper-parameters are more accurate than AE. Moreover, it performs better in terms of *precision*, *recall*, and the trade-off between them (i.e. F1-score). These results are attributed to the pre-training process of the stacked AEs layers (Figure 5.2) and the sparse constraint used with the `SSAE` models.

Table 5.2: Comparison of SSAE against different semi-supervised models on the test dataset.

Model	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
SSAE+DT	88.89	90.02	89.02	87.42
SSAE+SVM	56.19	63.65	55.09	57.11
SSAE+RF	90.07	90.91	89.68	88.11
SSAE+XGBoost	91.70	93.01	91.11	92.21
AE	92.58	93.11	92.84	92.65
SSAE	94.40	94.78	93.10	93.31
SSAE*	**95.03**	**96.40**	**94.01**	**94.40**

5.2.4.2 Comparison with the commonly-used supervised classification models (100% labeled data)

Also, to verify the efficiency of our model for traffic classification, we compared it with five references supervised classification models including (i) simple classifiers, which are DT and SVM, (ii) ensemble learning such as XGBoost and RF, (iii) neural network classifier, which is MLP classifier. We select these classifiers as our baselines because RF and DT are easy to train [16], SVM is widely used and proved to be useful in several applications [154], and MLP is a neural network model as well as XGBoost because it is an effective model for the classification task. In contrast to the above section, these classifiers use all the labeled data during the learning process and use all the original features X.

Table 5.3 presents the results achieved by our proposed model compared with the supervised classifiers. It is very clear that our model outperforms the ensemble models (i.e. XGBoost and RF), simple DL model (i.e. MLP) as well as simple classifiers (SVM and DT). Specifically, although the competitive results of XGBoost with 100% labeled data, our model gets the best results with the least amount of labeled data. It means that the proposed model with limited labeled data can get competitive *accuracy* compared with the well-known supervised models. In particular, the *accuracy* of our model is **2.47%**, **1.88%**, **0.15%**, **62.65%**, **5.75%** better than DT, RF, XGBoost, SVM, MLP, respectively. This may be attributed to the fact that our proposed model-based generates deeply learned features that yield far superior results compared to the initial statistical features. Moreover, it uses a pre-trained process that can boost the *accuracy* instead of the supervised models that are trained from scratch using all labeled data. From these results, we can conclude that our model is a robust model, extracts relevant features as well as can differentiate the applications very well.

Table 5.3: Comparison of SSAE against some supervised models.

Model	**Accuracy** (%)	**recision** (%)	**Recall** (%)	**F1-score** (%)
DT	92.56	93.36	93.00	93.16
RF	93.15	93.79	93.62	93.67
XGBoost	94.88	95.43	**94.12**	94.39
SVM	32.38	55.20	30.38	38.87
MLP	89.28	91.04	89.08	89.51
SSAE*	**95.03**	**96.40**	94.01	**94.40**

5.2.4.3 Comparison with supervised SSAE* models (using only the labeled ratio)

In order to further study the impact of the unlabeled data on the efficiency of our proposed model, we compared it with its supervised version. In other words, we evaluated the performance of our model (SSAE*) using only the labeled portion of data for its learning process without taking advantage of the unlabeled data. Table 5.4 illustrates the comparison of our model with and without the unlabeled data. It can be seen that our model in a supervised manner performs worse than the one using unlabeled data. In particular, the results show that unlabeled data boost the *accuracy* of the traffic classification by **11.29%**. This is because unlabeled data can provide informative characteristics and hence can boost the performance of traffic classification. This advantage might become even more significant in the case of larger training data.

Table 5.4: Comparison with supervised SSAE*.

Model	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
Supervised SSAE*	83.74	84.54	84.98	83.86
SSAE*	**95.03**	**96.40**	**94.01**	**94.40**

5.2.4.4 Confusion matrix (CM) comparison

Now we analyze a subset of results limited to some **well-known applications** using a confusion matrix (CM). CM compared the efficiency of the proposed model against two

methods (one based on non-DL and another based on the DL). According to the above results, we selected XGBoost as a supervised (non-deep learning) classifier because it gives the best results and we select AE (deep learning) as a semi-supervised model.

Figure 5.9 shows the classification results of these 3 models using CM where columns correspond to the actual class, rows refer to the predicted class. The CM provides information about the classes that are correctly or incorrectly classified and the type of misclassification.

As shown in Figure 5.9 (c), with XGBoost, many Youtube and Gmail flows, are incorrectly classified as Google, and vice versa. Moreover, we can see some interesting confusion between Google and Youtube, Gmail and Google, and Facebook and Gmail with AE (Figure 5.9 (b)). Finally, our proposed model incorrectly classifies Google as Youtube, but not the opposite (Figure 5.9 (a)). As a result, we can conclude that the proposed model provides slightly better classification results comparable to AE. It also performs slightly better than XGBoost but without the need to label all the data as with XGBoost.

5.2.4.5 Cost in terms of training and testing times

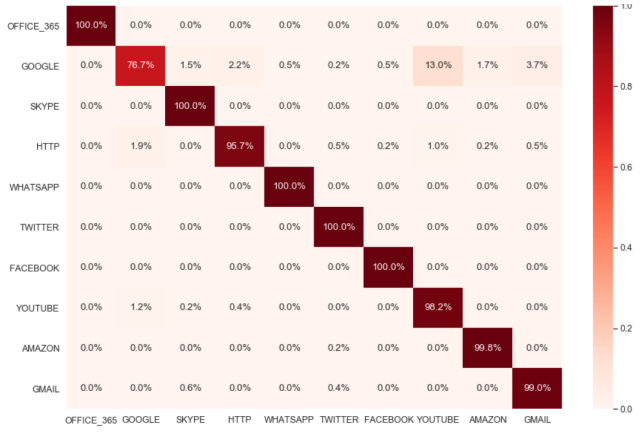
As a continuation of the previous subsection, we also compared the computational efficiency of the proposed model against XGBoost and AE. This comparison has been done in terms of training and classification time. Experiments are performed on a PC with, 8.00 GB of RAM and two cores of Intel® Core™ i5-7200U CPU@2.50GHz processor.

As shown in Figure 5.10 and Figure 5.11 that in terms of training time, the DL-based models (our model and the simple AE) need longer training time compared to the boosting model (i.e. XGBoost). However, once these models were trained, they were actually more efficient compared to the XGBoost in terms of classification time. In fact, as explained in Section 5.1.1, our model consists of two main phases (pre-training, and fine-tuning process) and this may explain its high training time. In contrast to the training, it is very fast for the classification task. We assume that the training time can be proceeded offline and thus does not impact the real-time utilization of the classification process.

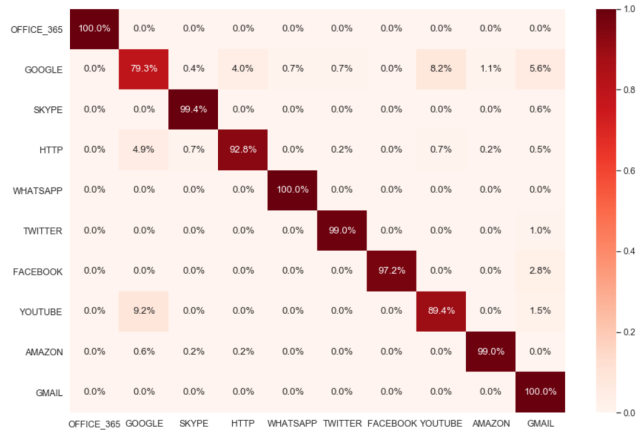
5.2.5 Experiments on the VPN-nonVPN dataset

To validate the effectiveness of the proposed model, we also conduct experiments based on another dataset, which includes encrypted data (VPN and non-VPN data).

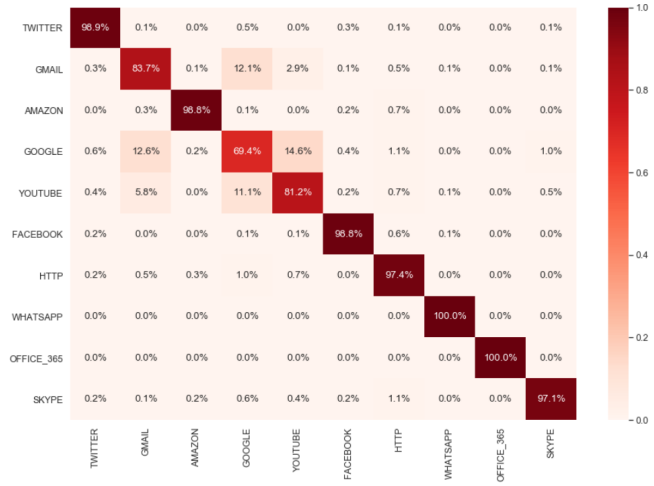
As with our previous contribution, from this dataset, we have selected two representative scenarios: (i) scenario A (Sc_A), which is a binary classification to indicate whether the traffic flow is VPN or not, and (ii) scenario D (Sc_D) that mixes all the applications to perform the multi-classification task (e.g. Chat, Streaming, VNP-chat, etc.). Note that all flows in the dataset are labeled. However, to evaluate our model, we only use a small portion of class labels during the training process. Specifically, we split the data into 80% for training, 10% for validation, and 10% for the test. Then, we distribute the training set into half labeled



(a) Confusion matrix for the proposed model



(b) Confusion matrix for Autoencoder



(c) Confusion matrix for XGBoost

Figure 5.9: A confusion matrix of the proposed model against AE and XGBoost under the most popular applications.

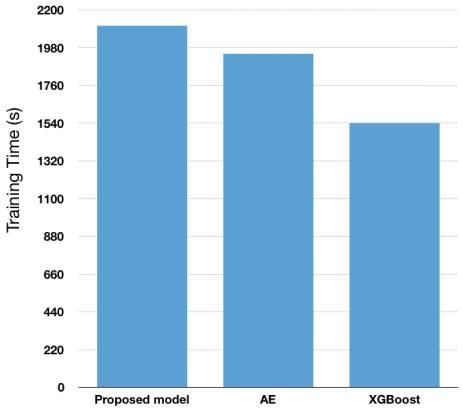


Figure 5.10: Training time comparison

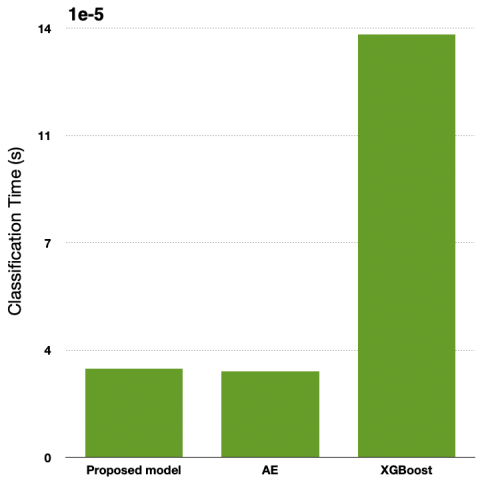


Figure 5.11: Classification time comparison per sample

and half-unlabeled. In addition, with our encoder layer, we have reduced the features from 23 to 15 features.

It can be seen from Table 5.5 and Table 5.6 that with half amount of labeled observations the *accuracy* of our model can achieve over 88%, 84% for Sc_A and Sc_D, respectively. Also, it can be seen that with few labeled observations as well as with the least amount of features, our model performs better than the simple classifiers like SVM and MLP. However, with this dataset, ensemble-based models (XGBoost) using totally labeled data, give better accuracy.

Table 5.5: Comparison with supervised models on Sc_A (using 100% labeled data).

Model	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
RF	90.30	90.35	90.18	90.24
XGBoost	93.02	93.18	92.85	92.97
SVM	60.98	61.71	59.77	58.69
MLP	73.72	73.63	73.68	73.65
SSAE*	88.04	88.05	88.02	88.03

Table 5.6: Comparison with supervised models on Sc_D (using 100% labeled data).

Model	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
RF	82.84	82.31	79.36	80.49
XGBoost	87.10	85.27	83.51	84.17
SVM	43.44	50.28	27.24	26.89
MLP	58.90	51.35	52.48	50.34
SSAE*	84.13	80.70	79.55	79.93

5.2.6 Performance against state-of-the-art models

In order to further validate the efficiency of the proposed model, a comparison with some state-of-the-art approaches has been done. The experimental results are presented in Table 5.7. It can be seen that the DL-based supervised model such as [102] and [155] outperforms all the approaches and specifically our model because they are more complex and use totally labeled data. However, this is not the case with the DT-based approach [114], where our model gives better results. This is may be attributed to the deep architecture used in our case. Also, our model outperforms the DL-based semi-supervised learning proposed by [159], and this is because of the pre-training process of the stacked AEs layer as well as the introduction of some hyper-parameters such as denoising and dropout rate.

Table 5.7: The classification accuracy (%) of baseline and ensemble methods on VPN-nonVPN Dataset.

Type	**Ref.**	**Model**	**Sc_A**	**Sc_D**
Supervised	[114]	DT	89.7	81.77
	[102]	CNN-LSTM	99.7	91.7
	[155]	Attention Based LSTM	99.7	91.2
Semi-supervised	[159]	Multi-task model based CNN	N/A	80.67
	SSAE*	SSAE+NN	88.04	84.31

5.3 Discussion

In this study, we proposed a complete and robust traffic classification system that makes use of both unlabeled and labeled data. We have analyzed the impact of the unlabeled data ratio on the performance of the proposed model. The evaluation demonstrates that this model needs a limited number of labels to get an *accuracy* over 95%. Next, hyper-parameter tuning has been done in order to improve performance. These hyper-parameters are specified to make the trained model lie on the balance point, which is neither under-fitting nor over-fitting.

Specifically, the proposed model has proved to classify the unknown applications very well and demonstrates the usefulness of the sparsity, denoising, and dropout for model generalization. Moreover, the performance of the proposed model may be attributed to the use of all the data contained in the dataset as well as the pre-training stage where each single AE is trained to exploit the relationship between high-level features and helps the deep neural network models to yield much better results with local initialization than random initialization. Then, the global fine-tuning process optimizes the parameters of the entire model, which greatly improves the classification task.

Pros and cons of the proposed semi-supervised classifier based on deep-learning

+ The proposed model has several advantages. First, it is simple and easy to implement. Second, it automatically provides feature extraction without human intervention and avoids time-wasting as maximum as possible. Third, tuning the model hyper-parameters helps to improve the performance of the final model. Finally, its performance continually improves when it is trained with more unlabeled data. Therefore, these ensure that the model is suitable for a real network environment containing a huge amount of unlabeled data.

− One of the disadvantages of this system is choosing the appropriate architecture and hyper-parameters. Furthermore, it requires an important time for the training task as well as needs some pre-processing like feature transformation and normalization. However, these are the normal procedures for any ML/DL model.

Conclusion

In this chapter, a semi-supervised network traffic classification system based on Stacked Sparse Autoencoder (SSAE) using two real network datasets has been proposed. It extracts features from unlabeled data and trains the classification model with a limited amount of labeled data. To do this, the `SSAE` model captured high-level feature representations in

an unsupervised manner through the pre-trained strategy and without human intervention. Then, a supervised neural network classifier is linked to the SSAE for the fine-tuning process and the classification task. Furthermore, different unlabeled data ratios have been investigated in order to obtain optimal performance based on the accuracy of the whole model. Next, sparse, dropout, and denoising code hyper-parameters have been injected to improve the generalization performance of the SSAE.

The simulation results show that the enhanced model SSAE* performs better than SSAE (i.e. without denoising and dropout hyper-parameters), simple AE (i.e. without stacked AEs pre-training and sparsity parameter), shallow ML models (DT, SVM), ensemble learning models (RF and XGBoost), as well as the supervised version of SSAE* (using only the labeled ratio). Moreover, we have evaluated the computational efficiency of the proposed model and the experimental results show that it outperforms XGBoost in terms of the classification time. In addition, the performance of the proposed model has also been evaluated against baseline approaches using a well-known dataset with different use cases and scenarios (binary classification and multi-classification).

Although the performance of the model, cannot be an efficient solution for intrusion detection. This is because collecting the data in a central entity for model training is a crucial step. Also, our model can cause some damage to the central entity if the traffic contains some attacks. Thus, to solve these issues in the next chapter, we will propose a Federated semi-supervised model for attack detection in order to keep the data where it was generated.

*****

Chapter 6

FLUIDS: Federated Learning with semi-supervised approach for Intrusion Detection System

Introduction

A large number of smart devices and sensors act in the background to collect the environment and user data. However, these devices are vulnerable to several cyber-attacks [160] where the attackers can intercept and analyze some sensitive data. These create significant challenges for the network operator, especially, for the security of the end-users. Consequently, to improve network security, some ultra-efficient, fast, and intelligent traffic analysis approaches are crucial. In general, intrusion detection can be considered as a classification problem by classifying the incoming traffic into normal or attack. Therefore, attack detection can take place via network traffic classification using models, especially DL-based models [161]. Besides the application classification, traffic classification has significance in security monitoring through attack classification or detection. DL models can automatically diagnose and detect attacks using flow and packet-based features.

In fact, building a conventional DL model consists of three steps: (i) data capture and labeling, (ii) data pre-processing, and (iii) model training. The first step requires that data are manually labeled after being captured, which is a highly expensive and time-consuming process. The second and third steps require the data owners to send their private data to a central entity for data pre-processing and model training. However, intrusion detection requires fast analysis, whereas sending user data to some central server is time-consuming [12]. Sending the data to the cloud/central entity over limited-bandwidth causes network congestion and in turn, unacceptable latency for applications in which real-time decisions have to be made. Furthermore, the conventional model training process raises privacy issues as confidential data might need to be shared in the process. Due to privacy concerns, sending the traffic for model training can cause some damage to the central entity if the traffic contains some attacks. In addition, the user may not be willing to share their data with a central entity [162]. Consequently, these render the conventional model training process out of intrusion detection application, and thus a decentralized computationally scalable methodology is very much in need.

To cope with these issues and leverage the value of existing network datasets while protecting privacy-sensitive users' data, Federated Learning (FL) appeared as a promising solution. It does not need to move the data to a central entity (Section 2.6, Chapter 2). With FL, a global model is trained collaboratively by each agent of the system (e.g. gateway) over the decentralized network. This is done via local updates, without exchanging private data [81]. Despite FL can be promising, one of the strong assumptions behind current FL-based solutions for IDS is that all the data is labeled (Section 3.2, Chapter 3). For labels, this assumption seems unrealistic, as this would mean that a human would have already tagged all the network traffic. This is difficult in practice (i) due to the resource constraints on the devices as well as some edge nodes and (ii) due to the difficulty of manually labeling data on such devices, which are far from human reach. To address these limitations, the

integration of FL and semi-supervised learning for the attack detection task is a promising direction.

Key contributions

Thus, we propose an FL-based semi-supervised model, called `FLUIDS`, which combines FL and semi-supervised learning for Intrusion Detection Systems (IDS). `FLUIDS` ensures privacy and takes advantage of the unlabeled and labeled data for intrusion detection. More specifically, we train a model using only a small amount of labeled data combined with more abundant unlabeled data. The contribution of this chapter can be summarized as follows:

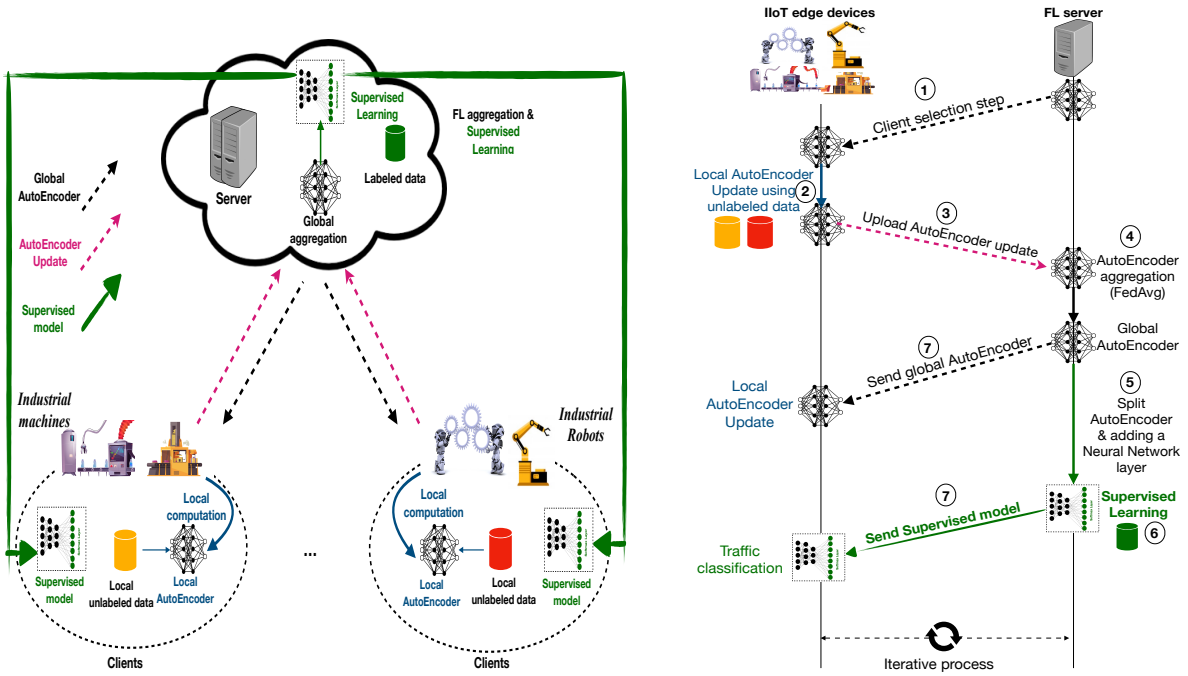
- a semi-supervised FL, combining the use of unsupervised learning at the client and supervised learning at the server. The unsupervised and supervised models are then concatenated to obtain a unified representation learning and classification solution for intrusion detection and attack classification;
- enabling the edge nodes to learn an efficient intrusion detection model without the need to label their local data;
- decreasing the burden of transmitting and labeling all the traffic at the server: by using FL, we add the edge nodes into the pipeline of the learning process and employ unsupervised learning at the edge;
- an extensive simulation using different datasets for binary classification (intrusion detection) and multi-classification (attack classification) has been done, as well as a comparison of `FLUIDS` against some state-of-the-art approaches, is presented.

The rest of this chapter is organized as follows. Section 6.1 presents the architecture and the main components of the proposed model. In Section 6.2 experimental results and performance evaluation of the proposed model are presented. Discussion and analysis of the results are provided in Section 6.3. Finally, the conclusion is given in Section 6.3.

6.1 FLUIDS methodology

The objective of this chapter is to train a semi-supervised model using only a small amount of labeled data while preserving data privacy for IDS. Thus, the proposed model (`FLUIDS`) takes advantage of both labeled and unlabeled data in a decentralized way and minimizes data exchange.

Specifically, as shown in Figure 6.1, `FLUIDS` consists of two parts: the client's side and the server's side. The clients perform the model pre-training using their *unlabeled* data and the server fine-tunes the global parameters using its limited *labeled* data (Figure 6.1 (a)).



(a) Federated semi-supervised architecture (b) Communication process

Figure 6.1: The network architecture and communication process of FLUIDS.

In **step 1** in Figure 6.1 (b), the server selects a random subset of clients, $n \leq K$, that will participate in the learning process and send them the initial AutoEncoder (AE) model. Then, on the client's side (edge devices), the AE model is trained for a selected number of epochs using the clients' unlabeled data with the objective of reducing the reconstruction error **(step 2)**. Here, the total unlabeled data are randomly distributed across the clients. Generally, the clients are often far from human reach, thus accessing them to label their data is a difficult and impractical task. Thus, to be more realistic, in this work we consider that the client data is fully unlabeled. Also, since the client generally has limited resources compared to the cloud node, we used AE instead of the SAE model (less complex).

Once the local training is finished, the clients send their local models to the cloud server for global aggregation **(step 3)**. On the server side, the AE is aggregated **(step 4)**, then the decoder is removed and a Fully Connected Network (FCN) layer is attached to the encoder layers in order to fine-tune the model parameters for the supervised learning using a limited labeled data located on the server **(step 5 and 6)**. Specifically, the server uses its limited labeled data for supervised learning, and thus unlike the classical FL, in our case, the server is not only used for model aggregation but also for supervised learning. Finally, the server sends back the global AE to the clients for a further update as well as the trained supervised model for inference (classification of the attacks) on the network data of the devices **(step 7)**. It is important to note that (i) no raw data is exchanged between the clients and the central

server, and (ii) the supervised model is trained through domain-specific public datasets or laboratory data located on the server without privacy concerns. Algorithm 3 describes the steps involved in designing a blending-based ensemble design, and Table 6.1 presents the notations used in Algorithm 3.

Algorithm 3: Learning procedure.

Input: Public labeled Dataset $D^l = \{x_i, y_i\}$ $(i = 1, 2, \ldots, n;\ x_i \in X;\ y_i \in Y)$; Private unlabeled dataset $D_k^u = \{x_i\}$ $(k = 1, 2, \ldots, K;\ x_i \in X_k)$, R, E_c, E_s, K, r_k
`/* --- Server side ---` */
Send initial global model $\theta = \theta_0$ to clients
for $i = 1$ to R **do**
  $n = r_k * K$
  for $j = 1$ to n **do in parallel**
`/* --- Client side ---` */
    for $e_c = 1$ to E_c **do**
      Update local AE parameters θ_j using D_j^u
    end for
    Send updated θ_j to the Server
  end for
`/* --- Server side ---` */
  Aggregate $\{\theta\}_{j=1\ldots n}$ with FedAvg into θ
  Extract `encoder`
  Concatenate `encoder` and `FCNLayer` into `H`
  for $e_s = 1$ to E_s **do**
    Train `H` using D^l
  end for
  Send models θ and `H` to the clients
end for

6.2 Experiment and performance evaluations

In this section, we evaluate the performance of `FLUIDS` through extensive experiments and discuss the results.

6.2.1 Experimental Setup

In this study, we split the dataset into two subsets: train (80%) and test (20%). We used `Python 3.7` as a programming language and `Scikit-learn 0.23` for the shallow models

Table 6.1: List of notations used in our model.

Notation	Meaning
K	Total number of clients
r_k	a fraction of the clients randomly selected from K
n	The number of clients selected at each round
E_c	Local clients epochs
E_s	Local server epochs
R	Total number of rounds
H	Supervised model
θ_0	Initial AE parameters

and `PyTorch 1.12` for DL models. All experiments are run on a four-core Intel® Core™ i7-6700 CPU@3.40GHz processor, and 32GB of RAM. Table 5.7 summarizes the model parameters and their selected settings in our simulations. As done in the previous chapter, the simulation of a partially-labeled dataset from this fully labeled data has been done by randomly selecting rows from the training set and removing their labels. It is important to note that during this experiment, we fixed the amount of labeled data and vary only the amount of unlabeled data. In addition, in our experiments, we computed the average of the evaluation metric from *five runs.*

6.2.2 Dataset description

To validate the effectiveness of our proposition, we used different datasets three datasets, which are *UNSW-NB15 dataset*[1] *[163]*, *gas pipeline SCADA system dataset*, and *water storage tank control system dataset*. The UNSW-NB15 dataset is recent and referenced in many existing papers. The simulation period of data was 16 hours on Jan 22, 2015, and 15 hours on Feb 17, 2015. The training set contains 175,341 and the testing set contains 82,332 total observations. Each observation is labeled either 0 if it is normal or 1 if it is an attack. Thus, this dataset has been used for intrusion detection tasks (binary classification).

Moreover, to validate the effectiveness of `FLUIDS` for attack classification or multi-classification tasks, we used a gas pipeline SCADA system dataset, which is a benchmark dataset for security research [141]. It was released by Mississippi State University in 2014. This dataset consists of 26 features and 1 label, the label contains eight possible values, 'benign' and seven different types of attacks. The possible values for the label are presented in Table 6.2.

The description of these attacks are as follows:

[1]https://research.unsw.edu.au/projects/unsw-nb15-data-set

Table 6.2: Gas pipeline SCADA system dataset description.

Label	**Description**	**# Total observations**
Benign	Normal traffic	61,156
NMRI	Naive malicious response injection	2,763
CMRI	Complex malicious response injection	15,466
MSCI	Malicious state command injection	782
MPCI	Malicious parameter command injection	7,637
MFCI	Malicious function command injection	573
DoS	Denial of service	1,837
Rec	Reconnaissance	6,805

- **DoS**: DoS attacks are the most common attacks over a network. It tries to weaken the network and the server by overwhelming traffic. It forces the victim to process these attack-generated requests or causes the machine to crash, thus making the provided service unavailable.

- **NMRI**: NMRI is a kind of response injection attack. These attacks leverage the ability to inject or alter response packets in a network; however, they lack the ability to obtain information about the underlying process being monitored and controlled.

- **CMRI**: CMRI is a kind of response injection attack. They are more sophisticated than NMRI attacks because they require an in-depth understanding of the targeted system. As such, CMRI attacks are designed to appear like normal process functionality. These attacks can be used to mask alterations to the process state perpetrated by malicious command injection attacks.

- **MSCI**: MSCI is a kind of command injection attack. These attacks change the state of the process control system to drive the system from a safe state to a critical state by sending malicious commands to remote field devices. In the case of the gas pipeline system, this attack includes command injections that turn the compressor on or off, and those that open or close the relief valve.

- **MPCI**: MPCI is a kind of command injection attack. They try to change the high and low set points for the water storage tank while disabling the liquid level alarms or changing the proportional integral derivative (attempts to maintain the air pressure in the pipeline) parameters used in the gas pipeline system.

- **Rec**: Rec is a kind of attack in which the gather information system and identify victim characteristics before launching an actual attack.

In addition, as this dataset consists of different features with values on different scales, the data were normalized in order to optimize the training process' performance.

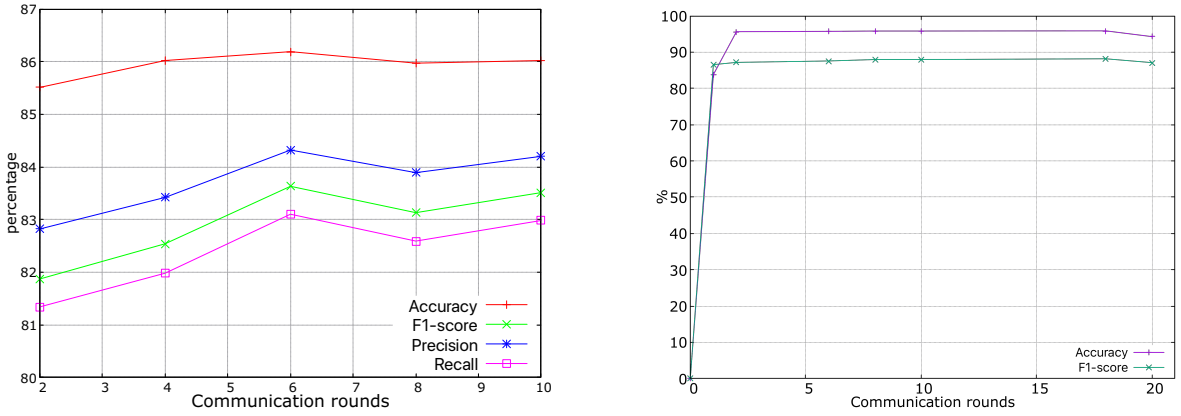
6.2.3 Performance under different factors

In this section, we study the performance of FLUIDS under different factors such as model architecture, communication rounds, the ratio of unlabeled data, and communication overhead. Also, we compare our model with its non-FL version (using the same model, and amount of labeled/unlabeled data). Since in real-time situations, the probability of client failures is significant [12] we use the Joint-Announcement Protocol (JAP) to avoid the problem of client failure and communication overhead. Given the ratio of clients (r_k), JAP selects randomly the clients that participate in the i^{th} training round [164].

6.2.3.1 Impact of communication rounds

In this subsection, we study the relationship between the performance of FLUIDS and the communication rounds. Since the edge nodes generally have limited resources compared to the cloud node, with the UNSW-NB15 dataset we set the number of epochs to 5 for the AE model training on the edges (i.e. client) and to 20 epochs for the supervised training located in the cloud. Similarly, with the Gas pipeline dataset, we set the number of epochs to 20 for AE model training and 100 epochs for supervised training. For each communication round (between FL server and clients), we present the performance of the proposed model (Figure 6.2) while keeping the remaining parameters fixed.

It can be seen from this result that with the UNSW-NB15 dataset the a*Accuracy* goes from **82.82%** in the second round to **84.32%** in round 6. Also, it can be seen using the gas pipeline dataset, FLUIDS can converge quickly and its accuracy starts to be stable after 2 rounds. These may be attributed to the fact that the AE trained on the clients provide some pre-trained layers, which capture relevant features. However, increasing the number of rounds does not always lead to better performance. This is because the model can overfit with large communication rounds.



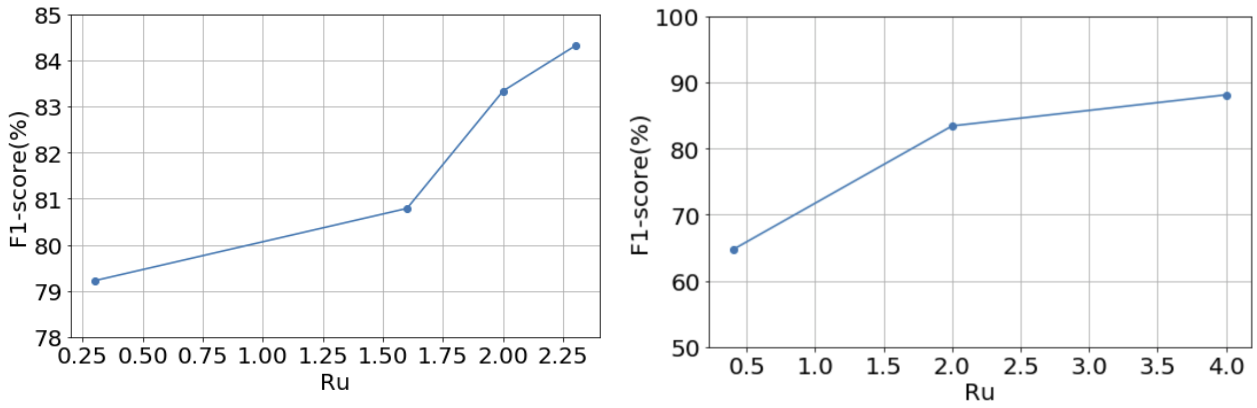
(a) UNSW-NB15 dataset (b) Gas pipeline dataset

Figure 6.2: Effect of communication rounds on FLUIDS performance

6.2.3.2 Impact of the unlabeled data available on the clients:

To investigate the impact of unlabeled data, we train our system using different ratios of unlabeled samples R_u while keeping the amount of labeled data fixed.

As shown in Figure 6.3, increasing the size of unlabeled data improves the performance of the whole model. These results are attributed to the fact that accessing more (diverse) unlabeled data provides informative characteristics to find a more discriminatory latent space (i.e. features) and, in turn, our model benefits from these data and boosts its performance to classify unseen observations.



(a) UNSW-NB15 dataset (b) Gas pipeline dataset

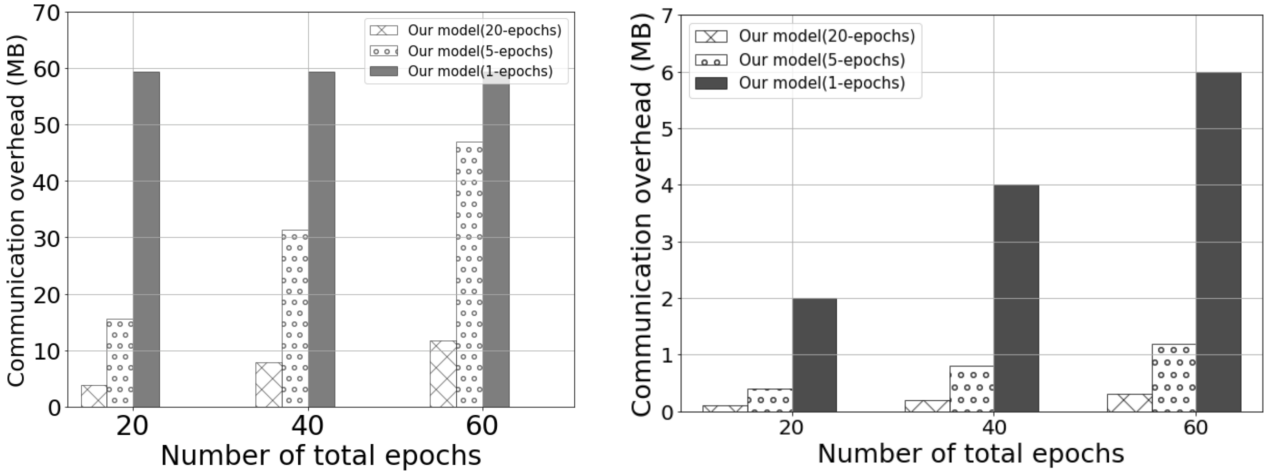
Figure 6.3: F1-score with various unlabeled ratio (R_u).

6.2.3.3 Communication overhead

By only exchanging the local model updates between the FL server and the clients, FL can help to reduce the communication overhead. Therefore, to minimize the communication overhead, two key aspects need to be considered: (i) reducing the local's model update frequency, and (ii) reducing the size of data communicated between the FL server and the clients [165]. We have taken into consideration these two aspects by considering two scenarios. The first scenario is with centralized learning and the second one is with FL. For the FL, we also considered two other scenarios by changing AE frequency updates, in the first one (our model with an update every 5 epochs), the edge nodes update their local model (e.g. AE model) after 5 epochs, while in the second one (20-epochs), the local models are updated after 20 epochs.

We can observe from Figure 6.4 that the local models' update frequency can impact the communication overhead. Moreover, in comparison to the centralized scenario, `FLUIDS` significantly reduces the size of the message. This advantage will become even more significant in the case of larger training data. This is mainly due to the fact that FL avoids transferring

raw data samples to the central entity and sends only model parameters. Also, through the use of the AE model, the FL clients compress their local data and hence reduce the size of the parameters communicated with the FL server.



(a) UNSW-NB15 dataset

(b) Gas pipeline dataset

Figure 6.4: Comparison in terms of communication overhead for two datasets

6.2.3.4 Performance against other models

In order to further validate the effectiveness of our model, we compare its performance against the non-FL semi-supervised model, different supervised models as well as some state-of-the-art models.

- **Comparison with non-FL mode**

Evaluation results of our proposed model in comparison to the non-FL model are presented in Table 6.3. With the non-FL model, the clients need to send their data to a central server in order to train the AE and the supervised models. The below formulas give the training process of our model and the non-FL model where R is the total communication rounds, E_c is client AE epochs and E_s is supervised learning epochs on the cloud. T_{FL} (resp. T_{nFL}) is the training process of our FL model (resp. the equivalent non-FL version of our model).

$$T_{FL} = R \times (E_c + E_s) \tag{6.1}$$

$$T_{nFL} = E_c + E_s \tag{6.2}$$

In comparison with the non-FL model, `FLUIDS` has the best results in terms of *accuracy* and *F1-score*. This is because the communication rounds help to improve the AE model

and in turn extract more relevant features that have been used during the supervised model training.

Table 6.3: Performance of the proposed model against the non-FL model.

Dataset	**Metric (%)**	**FLUIDS**	**non-FL model**
UNSW-NB15 dataset	Accuracy	**84.32**	81.40
	F1-score	**83.63**	80.39
Gas pipeline dataset	Accuracy	**95.84**	95.40
	F1-score	**88.14**	86.70

- **Comparison with supervised models**

In order to evaluate the effectiveness of `FLUIDS`, we compare its performance with several supervised models. It is important to note here, that we tested the performance of these models on the same test set used with `FLUIDS`.

Table 6.4 and Figure 6.5 illustrate the comparison of `FLUIDS` with those models using the UNSW-NB15 dataset. It is worth noting that our semi-supervised FL model outperforms the shallow supervised models in terms of all the evaluation metrics. For example, the *F1-score* is increased by **3.68%**, **5.46%**, **6.21%**, **7.55%** for MLP, RF, SVM, and DT, respectively.

Table 6.4: Comparison with supervised models using UNSW-NB15 dataset.

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
DT	76.48	76.36	75.94	76.08
SVM	79.46	85.94	77.21	77.42
RF	80.13	86.70	77.90	78.17
MLP	81.11	84.16	79.49	79.95
FLUIDS	**84.32**	**86.19**	**83.10**	**83.63**

To verify the efficiency of our model for traffic classification, we also compared it with different DT-based models including DT, RF, EXTree, AdaBoost, and LightGBM. To note here, since the gas pipeline dataset was used for the multi-classification task which is a more complex task than binary classification, we tried to use different ensemble models as baseline models. Also, we used the DT model because it is one of the most used models for attack classification.

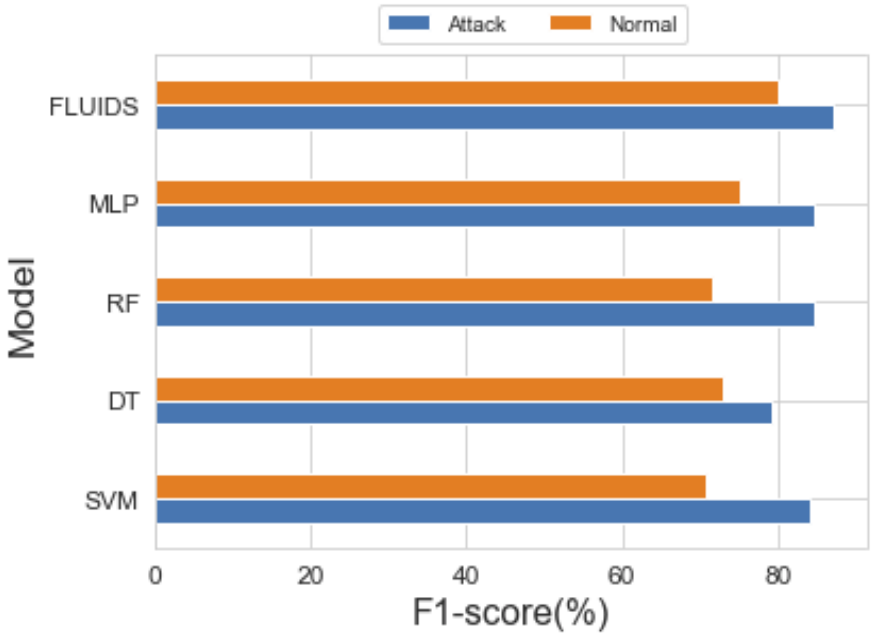


Figure 6.5: The performance of identifying normal and attack flows of FLUIDS against supervised models using the UNSW-NB15 dataset.

It can be seen from Table 6.5 that using the gas pipeline dataset, `FLUIDS` has a good performance in detecting benign network traffic and the different attacks. This is thanks to the use of deep architecture, which covers benign and attack patterns. Specifically, the benign traffic identification by `FLUIDS` is **6%**, **20%**, **21%**, **21%**, **27%** better than AdaBoost, LightGBM, EXTree, RF, and DT, respectively. This demonstrates that `FLUIDS` is more practical in the Industrial IoT network as it will trigger fewer false alarms. Moreover, although the competitive results of LightGBM and RF for $MSCI$, $MPCI$, and $MFCI$ attack classification, `FLUIDS` get the best results. On the other hand, it achieves a greater *F1-score* for the classification of the attack, except for the *DoS* and especially with *NMRI* attack. This is because the observation amount of this attack is relatively low (25%) in the labeled set.

These results may be attributed to the fact that the use of unlabeled data in the training process boosts the performance of `FLUIDS`. In addition, the proposed model outperforms these classifiers because, with the help of the clients' private data, the AE models generate deeply learned features that yield superior results compared to the initial statistical features.

- **Performance against state-of-the-art models**

To further validate the effectiveness of `FLUIDS`, we also compare its performance against some state-of-the-art schemes using the Gas pipeline dataset. The experimental results of these schemes are presented in Table 6.6 and they include a simple model with fully labeled data [166], a DL-based ensemble model using fully labeled data [167], a semi-supervised

Table 6.5: F1-score comparison of our model vs. supervised models for the identification of normal vs. attack traffic using gas pipeline dataset.

Model	**Benign**	**NMRI**	**CMRI**	**MSCI**	**MPCI**	**MFCI**	**DoS**	**Rec**
DT	0.70	0.32	0.94	0.94	0.97	0.98	**0.98**	**1**
RF	0.76	0.38	**0.96**	0.95	0.97	0.98	**0.98**	**1**
EXTree	0.76	0.49	0.92	0.95	0.95	0.97	0.87	**1**
AdaBoost	0.91	**0.89**	0.92	0.93	0.95	0.97	0.96	**1**
LightGBM	0.77	0.50	**0.96**	0.96	0.97	**0.99**	0.95	**1**
FLUIDS	**0.97**	0.20	**0.96**	**0.97**	**0.98**	**0.99**	0.97	**1**

model without FL [168], and a supervised ensemble FL scheme [140]. Note that we have selected these works because of their variety. Anton et al. [166] used SVM for intrusion detection. Huda et al. [167] proposed an ensemble Deep Belief Network (DBN) model for attack classification. In particular, different structures of DBNs are combined to construct an ensemble of DBNs and the final classification is decided based on a majority voting scheme. Also, Chang et al. [168] proposed an ensemble semi-supervised model using the K-means and convolutional autoencoder (CAE) methods. Using this ensemble, the test data is predicted as normal only if the predicted outputs of k-means and CAE methods are normal. Recently, Li et al. [140] proposed a novel FL model, called `DeepFed`. `DeepFed` is an ensemble model that trains CNN and GRU in a federated way to detect the attacks.

As shown in Table 6.6, the model based on FL including `FLUIDS` and `DeepFed` scheme [140] incurs the best results. This is due to the communication round can improve the performance of traffic classification to some extent. More specifically, although `FLUIDS` uses a few amounts (only 25%) of labeled data during the training task, it still achieves a competitive accuracy as compared with `DeepFed`. Moreover, with `FLUIDS`, the clients only train the AE model, which in turn is a less complex model as compared to the CNN-GRU models used with the `DeepFed` scheme. Note that a fair comparison for `FLUIDS` will be to compare it with only semi-supervised schemes. Nonetheless, we include some fully supervised schemes in our comparison for reference.

Table 6.6: Overall performance analysis of the proposed model with existing schemes.

Type	Ref.	FL	Accuracy (%)	Precision (%)	Recall (%)
Supervised	[166]		92.50	78.20	93.60
	[167]		95.60	85.36	85.53
	[140]	✓	99.20	98.85	97.45
Semi-supervised	[168]		95.53	95.43	83.52
	FLUIDS	✓	95.84	97.89	87.15

- **Experiments on third dataset**

Using another dataset, we have also tested FLUIDS against its non-FL version. This dataset is also an open dataset and has been released by Mississippi State University's lab in 2014 [141]. The traffic in this dataset corresponds to the water storage tank control system. It consists of 23 features and 236,179 observations. The label contains eight possible values, benign and seven different types of attacks, the same as the gas pipeline dataset. The possible values for the label are presented in Table 6.7. The aim of this section is to further verify the performance of FLUIDS for attack classification (i.e. multi-classification tasks).

Table 6.7: Water tank system dataset description.

Label	Description	# Total observations
Benign	Normal traffic	172,415
NMRI	Naive malicious response injection	9,187
CMRI	Complex malicious response injection	12,460
MSCI	Malicious state command injection	1,833
MPCI	Malicious parameter command injection	3,725
MFCI	Malicious function command injection	1,320
DoS	Denial of service	1,237
Rec	Reconnaissance	34,002

From the simulation results presented in Table 6.8 and Figure 6.6, we can see that FLUIDS performs slightly better than the non-FL model in terms of *accuracy*, *F1-score* as well as communication overhead.

Specifically, the improvement in terms of communication overhead becomes more significant with this dataset (reduced by almost 75%), as it contains more traffic than the gas pipeline system and hence sending row data to the central entity becomes more expansive. This shows that our model is suitable for a real scenario because industrial machines and robots can generate tremendous amounts of traffic.

Table 6.8: Performance of the proposed model against the non-FL model.

Metric	non-FL model	**FLUIDS**
Accuracy	90.41%	**90.73%**
F1-score	85.99%	**86.41%**

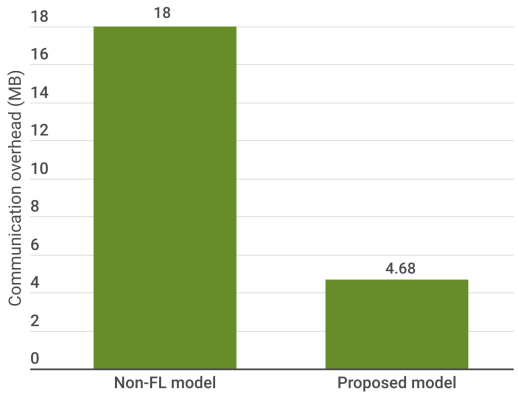


Figure 6.6: Comparison in terms of communication overhead.

6.3 Discussion

In this chapter, a semi-supervised FL model is proposed for attack and intrusion detection, called `FLUIDS`. This model uses both unlabeled and labeled data during the training process without privacy concerns. We have analyzed the impact of the different parameters on the performance of the proposed model. First, the evaluation demonstrates that this model performs well even with a limited amount of labels. It automatically provides feature extraction without human intervention and avoids time-wasting for labeling data as maximum as possible. Second, communication overhead, and storage requirements have been reduced thanks to the use of FL. Also, we have demonstrated that the local epochs play a critical role in communication overhead improvement. The experiment results demonstrate that the frequency model update has an impact directly on the communication overhead. Third, using a joint announcement protocol addresses the problem of communication overhead and the failure of some clients as well as alleviates the out-of-sync issue. In addition, by taking the advantage of FL our model solves the dilemma of data sharing. Last but not least, to show the features of `FLUIDS`, we have compared it against its non-FL setting, some state-of-the-art models including, simple, ensemble, semi-supervised, and FL models. Also, we applied our model to the different scenarios (i.e. binary and multi-classification tasks) and evaluated its performance in terms of *accuracy*, *F1-score*, and *communication overhead*.

From the above results and as can be seen from Table 6.9 that `FLUIDS` reduces the communication overhead and the storage requirement without breaching privacy. This is

because, with FL, the clients send only the model parameters. Also, in contrast to the non-FL supervised models, FLUIDS takes advantage of both unlabeled and labeled data, which makes them more efficient. Therefore, from this study, we show that FLUIDS can achieve higher classification with less communication overhead as well as without privacy concerns.

Table 6.9: FLUIDS vs equivalent model in non-FL setting vs Supervised models.

	non-FL Semi-supervised model	non-FL Supervised models	FLUIDS
Data aggregation	✓	✓	**x**
Parameters aggregation	**x**	**x**	✓
Communication overhead	High	High	Low
Data privacy	**x**	**x**	✓
Latency	High	High	Low
Storage requirement	High	High	Low
Data labeling	Low	High	Low
Labeled data	✓	✓	✓
Unlabeled data	✓	**x**	✓

Limitations of FLUIDS

Although FLUIDS uses the joint-announcement protocol, the random client selection can increase the training time and communication cost due to the clients who become stragglers. To handle these issues, more intelligent client selection algorithms are needed. One idea could be to use reinforcement learning, which can be a promising solution, to learn client selection based on the learning performance. Also, as FLUIDS still needs labels, in future research, we will investigate the fully unsupervised FL model by looking at the AE reconstruction error.

Conclusion

In this chapter, a federated semi-supervised learning model, called FLUIDS, has been proposed. This model uses a limited amount of labeled data and a huge amount of unlabeled data without privacy concerns. Also, unlike the classical FL model, with FLUIDS, the server is not only used for the model aggregation task, but also for supervised learning. The proposed model has been evaluated in terms of its ability to identify network intrusion and different attacks. The chapter presents the performance analysis of FLUIDS while varying different factors. Using different scenarios and datasets, the experimental results demonstrate that the support of unlabeled data for the training process can enhance the performance of the

learned model as well as decrease communication overhead. Also, the numerical simulations showed that `FLUIDS` with a limited amount of labeled data can achieve competitive results, compared to some state-of-the-art schemes.

*****

Conclusion and Future Directions

In this last chapter, the major findings of this thesis are summarised, and perspectives are provided for possible future research.

General conclusions

In this thesis, we focus on three important challenges in network analysis: (i) improving the classification performance and the model generalization on the training data, (ii), training a model using a limited amount of labeled data, and (iii) detecting attacks without the need to label the data on the edge nodes and preserve the data privacy at the same time. To deal with these complex challenges, and be motivated by the success of ML algorithms in solving complex tasks in several domains, extensive studies have been conducted, and solutions have been proposed. In particular, first, we provided a brief review of some related ML-based solutions for network traffic analysis, including traffic classification and IDS, as well as lastly we addressed their limitations. Then, to handle those limitations, we used machine, deep and federated learning technologies extensively to improve the network traffic classification task. In brief, the contributions of this thesis can be summarized as follows:

- In Chapter 4, we proposed a blending-based model to improve the classification and generalization capability of the model. In the blending design, a data pre-processing step includes feature selection using two feature selection methods and data cleaning has been conducted. In order to find the optimal features subset, we have used correlation filtering to pick up and delete the redundant features and in turn, reduce the processing time. Next, four tree-based models, i.e., DT, RF, AdaBoost, and XGBoost, are used as base classifiers, and then DL has been used as a meta-classifier in order to combine the output and correct the errors that occur during the learning process of the base classifiers. The experiment results show that the proposed model prevents overfitting and reduces bias simultaneously to some extent. In addition, we achieve good results on both non-encrypted, encrypted, binary, and multi-classification scenarios.

- In Chapter 5, a new semi-supervised model called `SSAE` has been proposed, to take advantage of both labeled and unlabeled data during the training process. Stacked

AutoEncoder (SAE) was used in order to better learn more complex and abstract features. In particular, several AutoEncoder (AE) layers are stacked together and form an unsupervised pre-training stage where the encoder layer computed by an AE will be used as the input to its next AE layer. Each layer in this stage is trained like an AE by minimizing its reconstructing error. When all the layers are pre-trained, the network goes into the supervised fine-tuning stage. Then, to further avoid the over-fitting problem, improve the classification performance `SSAE` model, and extract robust features some hyper-parameters including the sparse, dropout, and denoising coding, are injected into the model. Finally, a performance evaluation on both non-encrypted, encrypted, binary, and multi-classification scenarios. Also, a comparison against semi-supervised (e.g., AE model), and supervised models (well-known supervised models) have been conducted.

- In Chapter 6, an FL-based solution has been developed to enhance the security and privacy of the network data, called `FLUIDS`. A semi-supervised model as proposed in Chapter 5 was employed, which allows the edge nodes to participate in the training process without the need to label and share their local data. In particular, the nodes perform the model pre-training and AE model through their unlabeled data and then not only generate the global AE model by also fine-tuning the global parameters using its limited labeled data and during the supervised learning process. Using different scenarios and datasets, the experimental results demonstrate that the support of unlabeled data and FL can enhance the performance of the learned model and decrease the communication overhead, respectively. Also, the numerical simulations showed that `FLUIDS` outperforms its non-FL setting as well as with a limited amount of labeled data can achieve competitive results, compared to some state-of-the-art schemes.

Future directions

Based on the contributions proposed in this thesis and the limitations mentioned in each chapter, in this section, we will summarize some possible future directions of our contributions.

- **Future work on data privacy:** Despite the FL approach, exchanging the model parameters instead of the row data, the recent attacks demonstrate that such an approach does not provide sufficient privacy guarantee [169]. This is because the FL training process is based on the communication between the clients and the FL server that can expose the model and in turn be a target for several security threats, such as membership inference attack [170] [171]. Therefore, the proposed `FLUIDS` model requires a new solution with further security and privacy improvement. As a next step, we can use

a meta-model to securely aggregate the models' parameters [142]. In particular, the federated meta-model will be trained on the meta-data instead of user-sensitive data and in turn to further enhance privacy.

- **Future work on robust FLUIDS with constrained devices:** The clients with `FLUIDS` were simulated with the same computational resources which is not the case in a real environment. Some clients may take much longer to train the model than other nodes, and these nodes are often referred to as stragglers. These devices, even though are not limited in size, they come with limitations in resources. Their limited power and computational resources make the participation of such straggler clients in the FL process almost impossible. To mitigate the impact of stragglers clients, we can explore some techniques like *Federated Dropout* to reduce the computation load of local training [172]. Another option is to explore the client selection based on their resource conditions in order to maximize the efficiency of model training. In particular, instead of selecting random clients, we can apply *FedCS* [173].

- **Deep Learning in Network Traffic Prediction:** Prediction of network traffic plays a vital role in other networking-related challenges such as resource allocation. It aims to forecast the total amount of traffic expected [174] in order to avoid future congestion and maintain high network quality. Traffic prediction enables the network operator to present resource-allocation strategies, hence optimizing the network at various hours during the day. As this field grows, more and more models have been proposed, and choosing the appropriate model is a daunting task. Thus, there is a requirement for an extensive comparative analysis in terms of performance and training/test time needed for the different models and techniques used for traffic prediction.

- **Future work on explainability of the proposed models:** Although our proposed models are efficient, using them to make decisions (i.e., classify the traffic) is crucial to gain insights into the decision-making process of the models. The black-box nature of models raises a lack of trust an explanation solution is required for ML-based systems [175]. An eXplainable AI (XAI) has drawn significant attention from academia as it is essential and aims at explaining the outcomes of ML models. In other words, it enables the operators to understand why and why not a model makes such a decision, when it can fail and when can succeed [176]. XAI is defined by Gunning *et al.* [176] as *"XAI will create a suite of machine learning techniques that enables human users to understand, appropriately trust, and effectively manage the emerging generation of artificially intelligent partners."* An explanation is a way to generate better results, make confidence in the final decision, and consequently, make the network operators more comfortable with applying ML-based models. Moreover, in the case of the models yielding poor performance, it helps to understand the source and the reason for the

underlying problem [12]. In this context, we need to extend our proposed models by adding an explainability module. For example, we can explore the attention mechanism [177] to highlight the most important features that were mainly considered by the model when calculating the final class.

- **ML enabled green networking technology:** As DL-based models require massive computations to achieve acceptable performance, green deep learning becomes an increasingly hot research field. Recently, big attention is given to energy consumption during model training and inference [178]. Green learning, a term first proposed by Schwartz *et al.* [179], aims to use the ML/DL-based models without increasing computational cost rather, than ideally reducing it. In this context, to reduce the computation cost of our proposed models, we can explore several mechanisms such as model compression, transfer learning, and using fewer data to train our model while keeping the model performance. In particular, some effective pre-processing mechanisms can be applied to filter out the unrelated/noisy data that can reduce the dimensionality of the data [180]. Since the model size is an essential factor in the training and inference computation, using lightweight ML/DL models or model compression mechanisms is helpful to get faster prediction as well as to achieve the trade-off between the energy consumption of certain IoT devices and the final model performance. Moreover, to avoid the redundant computations required for training a new model, learning from scratch as well as solving the problem of insufficient training data, transfer learning can be a promising solution [181].

Alongside all these future directions, an important direction that can be explored further is network resources management. Indeed, using our proposed models and some of the future directions can provide efficient, automated, and fast convergence in network resource management in large-scale systems.

References

[1] *Cisco edge-to-enterprise IoT analytics for electric utilities solution overview, 2020.* https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/big-data/solution-overview-c22-740248.html.

[2] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper, Cisco, 2019.* https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[3] Mahmoud Abbasi, Amin Shahraki, and Amir Taherkordi. "Deep learning for network traffic monitoring and analysis (NTMA): a survey". In: *Computer Communications* 170 (2021), pp. 19–41.

[4] Min Zhang et al. "State of the art in traffic classification: A research review". In: *PAM Student Workshop*. 2009, pp. 3–4.

[5] Fannia Pacheco et al. "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey". In: *IEEE Communications Surveys & Tutorials* 21.2 (2018), pp. 1988–2014.

[6] Alessandro D'Alconzo et al. "A survey on big data for network traffic monitoring and analysis". In: *IEEE Transactions on Network and Service Management* 16.3 (2019), pp. 800–813.

[7] Raouf Boutaba et al. "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities". In: *Journal of Internet Services and Applications* 9.1 (2018), pp. 1–99.

[8] Ons Aouedi, Kandaraj Piamrat, and Benoît Parrein. "Intelligent Traffic Management in Next-Generation Networks". In: *Future internet* 14.2 (2022), p. 44.

[9] Alberto Dainotti, Antonio Pescape, and Kimberly C Claffy. "Issues and future directions in traffic classification". In: *IEEE network* 26.1 (2012), pp. 35–40.

[10] Ons Aouedi, Kandaraj Piamrat, and Benoît Parrein. "Ensemble-based Deep Learning model for network traffic classification". In: *IEEE Transactions on Network and Service Management* (July 2022). URL: https://hal.archives-ouvertes.fr/hal-03736603.

[11] Ons Aouedi et al. "Federated Semi-Supervised Learning for Attack Detection in Industrial Internet of Things". In: *IEEE Transactions on Industrial Informatics* (2022).

[12] Shaashwat Agrawal et al. "Federated learning for intrusion detection system: Concepts, challenges and future directions". In: *arXiv preprint arXiv:2106.09527* (2021).

[13] Ons Aouedi, Kandaraj Piamrat, and Dhruvjyoti Bagadthey. "Handling partially labeled network data: a semi-supervised approach using stacked sparse autoencoder". In: *Computer Networks* (2022).

[14] Ons Aouedi et al. "Intrusion detection for Softwarized Networks with Semi-supervised Federated Learning". In: *ICC 2022-IEEE International Conference on Communications*. IEEE. 2022, pp. 1–6.

[15] Ons Aouedi, Kandaraj Piamrat, and Benoît Parrein. "Decision tree-based blending method using deep-learning for network management". In: *IEEE/IFIP Network Operations and Management Symposium*. 2021.

[16] Ons Aouedi, Kandaraj Piamrat, and Benoît Parrein. "Performance evaluation of feature selection and tree-based algorithms for traffic classification". In: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE. 2021, pp. 1–6.

[17] Ons Aouedi, Kandaraj Piamrat, and Dhruvjyoti Bagadthey. "A semi-supervised stacked autoencoder approach for network traffic classification". In: *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE. 2020, pp. 1–6.

[18] Pedro Domingos. "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[19] Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260.

[20] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[21] Chollet Francois. *Deep learning with Python*. 2017.

[22] Guoqiang Peter Zhang. "Neural networks for classification: a survey". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.4 (2000), pp. 451–462.

[23] Ons Aouedi et al. "Network Traffic Analysis using Machine Learning: an unsupervised approach to understand and slice your network". In: *Annals of Telecommunications* (2021), pp. 1–13.

[24] Xiaojin Jerry Zhu. "Semi-supervised learning literature survey". In: (2005).

[25] Richard S Sutton, Andrew G Barto, et al. "Introduction to reinforcement learning". In: (1998).

[26] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3 (1992), pp. 279–292.

[27] Alexis Bitaillou, Benoît Parrein, and Guillaume Andrieux. "Q-routing: from the algorithm to the routing protocol". In: *International Conference on Machine Learning for Networking*. Springer. 2019, pp. 58–69.

[28] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[29] Menuka Perera Jayasuriya Kuranage, Kandaraj Piamrat, and Salima Hamma. "Network traffic classification using machine learning for software defined networks". In: *International Conference on Machine Learning for Networking*. Springer. 2019, pp. 28–39.

[30] Mikel Galar et al. "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4 (2011), pp. 463–484.

[31] Vicente García, José Salvador Sánchez, and Ramón Alberto Mollineda. "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance". In: *Knowledge-Based Systems* 25.1 (2012), pp. 13–21.

[32] Xinjian Guo et al. "On the class imbalance problem". In: *2008 Fourth international conference on natural computation*. Vol. 4. IEEE. 2008, pp. 192–201.

[33] Yoav Freund and Llew Mason. "The alternating decision tree learning algorithm". In: *icml*. Vol. 99. Citeseer. 1999, pp. 124–133.

[34] Tasnim Abar, Asma Ben Letaifa, and Sadok El Asmi. "Machine learning based QoE prediction in SDN networks". In: *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE. 2017, pp. 1395–1400.

[35] Pascal Soucy and Guy W Mineau. "A simple KNN algorithm for text categorization". In: *Proceedings 2001 IEEE international conference on data mining*. IEEE. 2001, pp. 647–648.

[36] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. "Data clustering: a review". In: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323.

[37] Robi Polikar. "Ensemble based systems in decision making". In: *IEEE Circuits and systems magazine* 6.3 (2006), pp. 21–45.

[38] Michał Woźniak, Manuel Grana, and Emilio Corchado. "A survey of multiple classifier systems as hybrid systems". In: *Information Fusion* 16 (2014), pp. 3–17.

[39] Sergey Tulyakov et al. "Review of classifier combination methods". In: *Machine learning in document analysis and recognition* (2008), pp. 361–386.

[40] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[41] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[42] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. "A comparative analysis of gradient boosting algorithms". In: *Artificial Intelligence Review* 54.3 (2021), pp. 1937–1967.

[43] Yoav Freund, Robert Schapire, and Naoki Abe. "A short introduction to boosting". In: *Journal-Japanese Society For Artificial Intelligence* 14.771-780 (1999), p. 1612.

[44] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. "CatBoost: gradient boosting with categorical features support". In: *arXiv preprint arXiv:1810.11363* (2018).

[45] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017).

[46] Andreas Töscher, Michael Jahrer, and Robert M Bell. "The bigchaos solution to the netflix grand prize". In: *Netflix prize documentation* (2009), pp. 1–52.

[47] Zhice Fang et al. "A comparative study of heterogeneous ensemble-learning techniques for landslide susceptibility mapping". In: *International Journal of Geographical Information Science* 35.2 (2021), pp. 321–347.

[48] Chia-Hsiu Chen et al. "Comparison and improvement of the predictability and interpretability with ensemble learning models in QSPR applications". In: *Journal of cheminformatics* 12.1 (2020), pp. 1–16.

[49] Tuan A Tang et al. ""Deep learning approach for network intrusion detection in Software Defined Networking"". In: *2016 international conference on wireless networks and mobile communications (WINCOM)*. IEEE. 2016, pp. 258–263.

[50] Qingchen Zhang et al. "A survey on deep learning for big data". In: *Information Fusion* 42 (2018), pp. 146–157.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[52] Ruben Mayer and Hans-Arno Jacobsen. "Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools". In: *ACM Computing Surveys (CSUR)* 53.1 (2020), pp. 1–37.

[53] Ronan Collobert and Samy Bengio. "Links between perceptrons, MLPs and SVMs". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 23.

[54] Yanming Guo et al. “Deep learning for visual understanding: A review”. In: *Neurocomputing* 187 (2016), pp. 27–48.

[55] Asifullah Khan et al. “A survey of the recent architectures of deep convolutional neural networks”. In: *Artificial intelligence review* 53.8 (2020), pp. 5455–5516.

[56] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[57] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[58] Chuanting Zhang et al. “Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data”. In: *IEEE Journal on Selected Areas in Communications* 37.6 (2019), pp. 1389–1401.

[59] Samira Pouyanfar et al. “A survey on deep learning: Algorithms, techniques, and applications”. In: *ACM Computing Surveys (CSUR)* 51.5 (2018), pp. 1–36.

[60] Alexandra L’heureux et al. “Machine learning with big data: Challenges and approaches”. In: *Ieee Access* 5 (2017), pp. 7776–7797.

[61] Isabelle Guyon et al. “Gene selection for cancer classification using support vector machines”. In: *Machine learning* 46.1 (2002), pp. 389–422.

[62] Mahesh Pal and Giles M Foody. “Feature selection for classification of hyperspectral data by SVM”. In: *IEEE Transactions on Geoscience and Remote Sensing* 48.5 (2010), pp. 2297–2307.

[63] Huan Liu and Lei Yu. “Toward integrating feature selection algorithms for classification and clustering”. In: *IEEE Transactions on knowledge and data engineering* 17.4 (2005), pp. 491–502.

[64] Alexandros Kalousis, Julien Prados, and Melanie Hilario. “Stability of feature selection algorithms: a study on high-dimensional spaces”. In: *Knowledge and information systems* 12.1 (2007), pp. 95–116.

[65] Manoranjan Dash and Huan Liu. “Feature selection for classification”. In: *Intelligent data analysis* 1.3 (1997), pp. 131–156.

[66] Jasmina Novakovic. "Using information gain attribute evaluation to classify sonar targets". In: *17th Telecommunications forum TELFOR*. Citeseer. 2009, pp. 1351–1354.

[67] Fadi Salo, Ali Bou Nassif, and Aleksander Essex. "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection". In: *Computer Networks* 148 (2019), pp. 164–175.

[68] Mark Andrew Hall et al. "Correlation-based feature selection for machine learning". In: (1999).

[69] Andreas Janecek et al. "On the relationship between feature selection and classification accuracy". In: *New challenges for feature selection in data mining and knowledge discovery*. PMLR. 2008, pp. 90–105.

[70] Ian T Jolliffe. "Principal components in regression analysis". In: *Principal component analysis*. Springer, 1986, pp. 129–155.

[71] Guoqiang Peter Zhang. "Neural networks for classification: a survey". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.4 (2000), pp. 451–462.

[72] Ons Aouedi, Mohamed Anis Bach Tobji, and Ajith Abraham. "An ensemble of deep auto-encoders for healthcare monitoring". In: *International Conference on Hybrid Intelligent Systems*. Springer. 2018, pp. 96–105.

[73] Francisco J Pulgar et al. "AEkNN: An AutoEncoder kNN-based classifier with built-in dimensionality reduction". In: *arXiv preprint arXiv:1802.08465* (2018).

[74] David Charte et al. "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines". In: *Information Fusion* 44 (2018), pp. 78–96.

[75] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.

[76] Pascal Vincent et al. “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.” In: *Journal of machine learning research* 11.12 (2010).

[77] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[78] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

[79] Sawsan AbdulRahman et al. “A survey on federated learning: The journey from centralized to distributed on-site learning and beyond”. In: *IEEE Internet of Things Journal* 8.7 (2020), pp. 5476–5497.

[80] Qiang Yang et al. “Federated machine learning: Concept and applications”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19.

[81] Brendan McMahan et al. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[82] Stacey Truex et al. “A hybrid approach to privacy-preserving federated learning”. In: *Proceedings of the 12th ACM workshop on artificial intelligence and security*. 2019, pp. 1–11.

[83] Teodora Sandra Buda et al. “Can machine learning aid in delivering new use cases and scenarios in 5G?” In: *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2016, pp. 1279–1284.

[84] *(2020, February) IANA, Port Numbers. [Online]. Available: http://www.iana.org/assignments/port-numbers.*

[85] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. “Accurate, scalable in-network identification of p2p traffic using application signatures”. In: *Proceedings of the 13th international conference on World Wide Web*. 2004, pp. 512–521.

[86] Andrew W Moore and Konstantina Papagiannaki. “Toward the accurate identification of network applications”. In: *International workshop on passive and active network measurement*. Springer. 2005, pp. 41–54.

[87] Alok Madhukar and Carey Williamson. "A longitudinal study of P2P traffic classification". In: *14th IEEE international symposium on modeling, analysis, and simulation*. IEEE. 2006, pp. 179–188.

[88] Matthew Hayes et al. "Scalable architecture for SDN traffic classification". In: *IEEE Systems Journal* 12.4 (2017), pp. 3203–3214.

[89] Zafar Ayyub Qazi et al. "Application-awareness in SDN". In: *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. 2013, pp. 487–488.

[90] Yunchun Li and Jingxuan Li. "MultiClassifier: A combination of DPI and ML for application-layer classification in SDN". In: *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*. IEEE. 2014, pp. 682–686.

[91] Mostafa Uddin and Tamer Nadeem. "TrafficVision: A case for pushing software defined networks to wireless edges". In: *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE. 2016, pp. 37–46.

[92] Iyad Lahsen Cherif and Abdesselem Kortebi. "On using extreme gradient boosting (XGBoost) machine learning algorithm for home network traffic classification". In: *2019 Wireless Days (WD)*. IEEE. 2019, pp. 1–6.

[93] Pramitha Perera et al. "A comparison of supervised machine learning algorithms for classification of communications network traffic". In: *International Conference on Neural Information Processing*. Springer. 2017, pp. 445–454.

[94] Razan M AlZoman and Mohammed JF Alenazi. "A comparative study of traffic classification techniques for smart city networks". In: *Sensors* 21.14 (2021), p. 4677.

[95] Chuangchuang Zhang et al. "Deep learning–based network application classification for SDN". In: *Transactions on Emerging Telecommunications Technologies* 29.5 (2018), e3302.

[96] Mohammad Lotfollahi et al. "Deep packet: A novel approach for encrypted traffic classification using deep learning". In: *Soft Computing* 24.3 (2020), pp. 1999–2012.

[97] Qing Lyu and Xingjian Lu. "Effective media traffic classification using deep learning". In: *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*. 2019, pp. 139–146.

[98] Santiago Egea Gómez et al. "Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal". In: *Computer Networks* 127 (2017), pp. 68–80.

[99] Ting Yang et al. "Achieving Robust Performance for Traffic Classification Using Ensemble Learning in SDN Networks". In: *ICC 2021-IEEE International Conference on Communications*. IEEE. 2021, pp. 1–6.

[100] Won-Ju Eom et al. "Network traffic classification using ensemble learning in software-defined networks". In: *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE. 2021, pp. 089–092.

[101] Xin Wang, Shuhui Chen, and Jinshu Su. "App-Net: a hybrid neural network for encrypted mobile traffic classification". In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2020, pp. 424–429.

[102] Kunda Lin, Xiaolong Xu, and Honghao Gao. "TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT". In: *Computer Networks* 190 (2021), p. 107974.

[103] ThankGod Obasi and M Omair Shafiq. "An experimental study of different machine and deep learning techniques for classification of encrypted network traffic". In: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, pp. 4690–4699.

[104] ThankGod Obasi and M Omair Shafiq. "CARD-B: A stacked ensemble learning technique for classification of encrypted network traffic". In: *Computer Communications* (2022).

[105] Kazuki Hara and Kohei Shiomoto. "Intrusion detection system using semi-supervised learning with adversarial auto-encoder". In: *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2020, pp. 1–8.

[106] Jonas Höchst et al. "Unsupervised traffic flow classification using a neural autoencoder". In: *2017 IEEE 42Nd Conference on local computer networks (LCN)*. IEEE. 2017, pp. 523–526.

[107] Shahbaz Rezaei and Xin Liu. "How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets". In: *arXiv preprint arXiv:1812.09761* (2018).

[108] Pu Wang, Shih-Chun Lin, and Min Luo. "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs". In: *2016 IEEE international conference on services computing (SCC)*. IEEE. 2016, pp. 760–765.

[109] Sara Ayoubi et al. "Machine learning for cognitive network management". In: *IEEE Communications Magazine* 56.1 (2018), pp. 158–165.

[110] Albert Mestres et al. "Knowledge-defined networking". In: *ACM SIGCOMM Computer Communication Review* 47.3 (2017), pp. 2–10.

[111] Meenaxi M Raikar et al. ""Data traffic classification in Software Defined Networks (SDN) using supervised-learning"". In: *Procedia Computer Science* 171 (2020), pp. 2750–2759.

[112] Pedro Amaral et al. "Machine learning in software defined networks: Data collection and traffic classification". In: *2016 IEEE 24th International conference on network protocols (ICNP)*. IEEE. 2016, pp. 1–5.

[113] Pan Wang et al. "Datanet: Deep learning based encrypted network traffic classification in sdn home gateway". In: *IEEE Access* 6 (2018), pp. 55380–55391.

[114] Gerard Draper-Gil et al. "Characterization of encrypted and vpn traffic using time-related". In: *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*. sn. 2016, pp. 407–414.

[115] Umer Majeed, Latif U Khan, and Choong Seon Hong. "Cross-silo horizontal federated learning for flow-based time-related-features oriented traffic classification". In: *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE. 2020, pp. 389–392.

[116] Hyunsu Mun and Youngseok Lee. "Internet traffic classification with federated learning". In: *Electronics* 10.1 (2020), p. 27.

[117] Umer Majeed, Sheikh Salman Hassan, and Choong Seon Hong. "Cross-silo model-based secure federated transfer learning for flow-based traffic classification". In: *2021*

International Conference on Information Networking (ICOIN). IEEE. 2021, pp. 588–593.

[118] R Base and P Mell. "Special publication on intrusion detection systems". In: *NIST Infidel, Inc., National Institute of Standards and Technology, Scotts Valley, CA* (2001).

[119] Nasrin Sultana et al. "Survey on SDN based network intrusion detection system using machine learning approaches". In: *Peer-to-Peer Networking and Applications* 12.2 (2019), pp. 493–501.

[120] Nadia Chaabouni et al. "Network intrusion detection for IoT security based on learning techniques". In: *IEEE Communications Surveys & Tutorials* 21.3 (2019), pp. 2671–2701.

[121] Sang-Woong Lee et al. "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review". In: *Journal of Network and Computer Applications* 187 (2021), p. 103111.

[122] Robert A Bridges et al. "A survey of intrusion detection systems leveraging host data". In: *ACM Computing Surveys (CSUR)* 52.6 (2019), pp. 1–35.

[123] Elike Hodo et al. "Shallow and deep networks intrusion detection system: A taxonomy and survey". In: *arXiv preprint arXiv:1701.02145* (2017).

[124] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z Emam. "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues". In: *Knowledge-Based Systems* 189 (2020), p. 105124.

[125] Syeda Manjia Tahsien, Hadis Karimipour, and Petros Spachos. "Machine learning based solutions for security of Internet of Things (IoT): A survey". In: *Journal of Network and Computer Applications* 161 (2020), p. 102630.

[126] Vibekananda Dutta et al. "Hybrid model for improving the classification effectiveness of network intrusion detection". In: *Computational Intelligence in Security for Information Systems Conference*. Springer. 2019, pp. 405–414.

[127] Saeid Soheily-Khah, Pierre-François Marteau, and Nicolas Béchet. "Intrusion detection in network systems through hybrid supervised and unsupervised machine learning

process: A case study on the iscx dataset". In: *2018 1st International Conference on Data Intelligence and Security (ICDIS)*. IEEE. 2018, pp. 219–226.

[128] Parag Verma et al. "Network intrusion detection using clustering and gradient boosting". In: *2018 9th International conference on computing, communication and networking technologies (ICCCNT)*. IEEE. 2018, pp. 1–7.

[129] Sharmila Kishor Wagh and Satish R Kolhe. "Effective intrusion detection system using semi-supervised learning". In: *2014 International Conference on Data Mining and Intelligent Computing (ICDMIC)*. IEEE. 2014, pp. 1–5.

[130] Thien Duc Nguyen et al. "DÏoT: A federated self-learning anomaly detection system for IoT". In: *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*. IEEE. 2019, pp. 756–767.

[131] Zhuo Chen et al. "Intrusion detection for wireless edge networks based on federated learning". In: *IEEE Access* 8 (2020), pp. 217463–217472.

[132] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. "Describing multimedia content using attention-based encoder-decoder networks". In: *IEEE Transactions on Multimedia* 17.11 (2015), pp. 1875–1886.

[133] Ekta Sood et al. "Interpreting attention models with human visual attention in machine reading comprehension". In: *arXiv preprint arXiv:2010.06396* (2020).

[134] Ming Li et al. "A deep learning method based on an attention mechanism for wireless network traffic prediction". In: *Ad Hoc Networks* 107 (2020), p. 102258.

[135] Ruijie Zhao et al. "Intelligent intrusion detection based on federated learning aided long short-term memory". In: *Physical Communication* 42 (2020), p. 101157.

[136] Viraaji Mothukuri et al. "Federated learning-based anomaly detection for IoT security attacks". In: *IEEE Internet of Things Journal* (2021).

[137] Dinesh Chowdary Attota et al. "An Ensemble Multi-View Federated Learning Intrusion Detection for IoT". In: *IEEE Access* (2021).

[138] Sawsan Abdul Rahman et al. "Internet of things intrusion detection: Centralized, on-device, or federated learning?" In: *IEEE Network* 34.6 (2020), pp. 310–317.

[139] Othmane Friha et al. “FELIDS: Federated learning-based intrusion detection system for agricultural Internet of Things”. In: *Journal of Parallel and Distributed Computing* 165 (2022), pp. 17–31.

[140] Beibei Li et al. “DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber–Physical Systems”. In: *IEEE Transactions on Industrial Informatics* 17.8 (2020), pp. 5615–5624.

[141] Thomas Morris and Wei Gao. “Industrial control system traffic data sets for intrusion detection research”. In: *International Conference on Critical Infrastructure Protection*. Springer. 2014, pp. 65–78.

[142] Noor Ali Al-Athba Al-Marri, Bekir S Ciftler, and Mohamed M Abdallah. “Federated mimic learning for privacy preserving intrusion detection”. In: *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. IEEE. 2020, pp. 1–6.

[143] Quamar Niyaz, Weiqing Sun, and Ahmad Y Javaid. “”A deep learning based DDoS detection system in software-defined networking (SDN)””. In: *arXiv preprint arXiv:1611.07400* (2016).

[144] Muhammad Aamir and Syed Mustafa Ali Zaidi. “Clustering based Semi-Supervised Machine Learning for DDoS attack classification”. In: *Journal of King Saud University-Computer and Information Sciences* (2019).

[145] Prabhakar Krishnan, Subhasri Duttagupta, and Krishnashree Achuthan. “”VARMAN: Multi-plane security framework for software defined networks””. In: *Computer Communications* 148 (2019), pp. 215–239.

[146] Jingyi Li et al. “FIDS: Detecting DDoS Through Federated Learning Based Method”. In: *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE. 2021, pp. 856–862.

[147] Thuy TT Nguyen and Grenville Armitage. “A survey of techniques for internet traffic classification using machine learning”. In: *IEEE communications surveys & tutorials* 10.4 (2008), pp. 56–76.

[148] Yawen Xiao et al. “A deep learning-based multi-model ensemble method for cancer prediction”. In: *Computer methods and programs in biomedicine* 153 (2018), pp. 1–9.

[149] Isadora P Possebon et al. "Improved network traffic classification using ensemble learning". In: *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2019, pp. 1–6.

[150] Prabhat Kumar, Govind P Gupta, and Rakesh Tripathi. "An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks". In: *Computer Communications* 166 (2021), pp. 110–124.

[151] Alessio Zappone, Marco Di Renzo, and Mérouane Debbah. "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" In: *IEEE Transactions on Communications* 67.10 (2019), pp. 7331–7376.

[152] M Paz Sesmero, Agapito I Ledezma, and Araceli Sanchis. "Generating ensembles of heterogeneous classifiers using stacked generalization". In: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 5.1 (2015), pp. 21–34.

[153] Juan Sebastián Rojas, Álvaro Rendón Gallón, and Juan Carlos Corrales. "Personalized service degradation policies on OTT applications based on the consumption behavior of users". In: *International Conference on Computational Science and Its Applications*. Springer. 2018, pp. 543–557.

[154] Ilhan Firat Kilincer, Fatih Ertam, and Abdulkadir Sengur. "Machine learning methods for cyber security intrusion detection: Datasets and comparative study". In: *Computer Networks* 188 (2021), p. 107840.

[155] Haipeng Yao et al. "Identification of encrypted traffic through attention mechanism based long short term memory". In: *IEEE Transactions on Big Data* (2019).

[156] Miguel Camelo et al. "A semi-supervised learning approach towards automatic wireless technology recognition". In: *2019 IEEE international symposium on dynamic Spectrum access networks (DySPAN)*. IEEE. 2019, pp. 1–10.

[157] Dumitru Erhan et al. "Why does unsupervised pre-training help deep learning?" In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 201–208.

[158] Alaa Sagheer and Mostafa Kotb. "Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems". In: *Scientific reports* 9.1 (2019), pp. 1–16.

[159] Shahbaz Rezaei and Xin Liu. "Multitask learning for network traffic classification". In: *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE. 2020, pp. 1–9.

[160] Nivedita Mishra and Sharnil Pandya. "Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review". In: *IEEE Access* (2021).

[161] Djallel Hamouda et al. "Intrusion Detection Systems for Industrial Internet of Things: A Survey". In: *2021 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS)*. IEEE. 2021, pp. 1–8.

[162] Mohamed Amine Ferrag et al. "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis". In: *IEEE Access* 9 (2021), pp. 138509–138542.

[163] Nour Moustafa and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In: *2015 military communications and information systems conference (MilCIS)*. IEEE. 2015, pp. 1–6.

[164] Yi Liu et al. "Privacy-preserving traffic flow prediction: A federated learning approach". In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7751–7763.

[165] Tian Li et al. "Federated learning: Challenges, methods, and future directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60.

[166] Simon D Duque Anton, Sapna Sinha, and Hans Dieter Schotten. "Anomaly-based Intrusion Detection in industrial data with SVM and Random Forests". In: *2019 International conference on software, telecommunications and computer networks (SoftCOM)*. IEEE. 2019, pp. 1–6.

[167] Shamsul Huda et al. "Securing the operations in SCADA-IoT platform based industrial control system using ensemble of Deep Belief Networks". In: *Applied soft computing* 71 (2018), pp. 66–77.

[168] Chun-Pi Chang, Wen-Chiao Hsu, and I-En Liao. "Anomaly Detection for industrial control systems using k-means and Convolutional AutoEncoder". In: *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE. 2019, pp. 1–6.

[169] Viraaji Mothukuri et al. “A survey on security and privacy of federated learning”. In: *Future Generation Computer Systems* 115 (2021), pp. 619–640.

[170] Hongsheng Hu et al. “Membership inference attacks on machine learning: A survey”. In: *ACM Computing Surveys (CSUR)* (2021).

[171] Milad Nasr, Reza Shokri, and Amir Houmansadr. “Comprehensive privacy analysis of deep learning”. In: *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 1–15.

[172] Sebastian Caldas et al. “Expanding the reach of federated learning by reducing client resource requirements”. In: *arXiv preprint arXiv:1812.07210* (2018).

[173] Takayuki Nishio and Ryo Yonetani. “Client selection for federated learning with heterogeneous resources in mobile edge”. In: *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE. 2019, pp. 1–7.

[174] Iraj Lohrasbinasab et al. “From statistical-to machine learning-based network traffic prediction”. In: *Transactions on Emerging Telecommunications Technologies* 33.4 (2022), e4394.

[175] Arun Das and Paul Rad. “Opportunities and challenges in explainable artificial intelligence (xai): A survey”. In: *arXiv preprint arXiv:2006.11371* (2020).

[176] David Gunning. “Explainable artificial intelligence (xai)”. In: *Defense advanced research projects agency (DARPA), nd Web* 2.2 (2017), p. 1.

[177] Djamila Beddiar, Mourad Oussalah, and Seppänen Tapio. “Explainability for Medical Image Captioning”. In: *2022 Eleventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE. 2022, pp. 1–6.

[178] Jingjing Xu et al. “A survey on green deep learning”. In: *arXiv preprint arXiv:2111.05193* (2021).

[179] Roy Schwartz et al. “Green ai”. In: *Communications of the ACM* 63.12 (2020), pp. 54–63.

[180] G Thippa Reddy et al. “Analysis of dimensionality reduction techniques on big data”. In: *IEEE Access* 8 (2020), pp. 54776–54788.

[181] Aditya Khamparia et al. “Internet of health things-driven deep learning system for detection and classification of cervical cells using transfer learning”. In: *The Journal of Supercomputing* 76.11 (2020), pp. 8590–8608.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

Nantes Université

Titre : Analyse du trafic réseau basée sur l'apprentissage automatique

Mot clés : Apprentissage automatique, Apprentissage fédéré, analyse du trafic

Résumé : L'Internet des Objets entraînent par son nombre de terminaux une explosion du trafic de données. Pour augmenter la qualité globale de réseau, il est possible d'analyser intelligemment le trafic réseau afin de détecter d'éventuel comportement suspect ou malveillant. Les modèles d'apprentissage automatique et d'apprentissage profond permettent de traiter ce très grand volume de données. Néanmoins, il existe certaines limites dans la littérature, notamment la confidentialité des données, le surapprentissage (manques de diversité dans les données) ou tout simplement le manque de jeu de données labélisées. Dans cette thèse, nous proposons de nouveaux modèles s'appuyant sur l'apprentissage automatique et l'apprentissage profond afin de traiter une grande quantité de données tout en préservant la confidentialité. Notre première approche utilise un modèle d'ensemble. Les résultats montrent une diminution du surapprentissage, tout en augmentant de 10% la précision comparé à des modèles de l'état de l'art. Notre seconde contribution s'attache aux problèmes de disponibilité des données labélisées. Nous proposons un modèle d'apprentissage semi-supervisé capable d'améliorer la précision de 11% par rapport à un modèle supervisé équivalent. Enfin, nous proposons un système de détection d'attaque s'appuyant sur l'apprentissage fédéré. Nommé FLUIDS, il permet de réduire la surcharge réseau de 75% tout en préservant de très haute performance et la confidentialité.

Title: Machine Learning-Enabled Network Traffic Analysis

Keywords: Machine Learning, Federated Learning, traffic analysis

Abstract: Recent development in network communication along with the drastic increase in the number of smart devices leads to an explosion in data generation. To this end, intelligent network traffic analysis can help to understand the behavior of connected smart devices and applications as well as provides defense against cyber-attacks. In this line, Machine Learning (ML) and Deep Learning (DL) models have the ability to model and uncover hidden patterns using training data or environment. Despite their benefits, major challenges need to be addressed such as model generalization (due to model overfitting), lack of label (due to the difficulty to label all the data), and privacy (due to recent regulations). In this thesis, new ML/DL-based models are proposed for tackling these challenges. The first contribution focuses on improving the generalization and classification performance by proposing an ensemble blending model. The simulation results show that the accuracy of the proposed ensemble model is 10%, better than some state-of-the-art models. Second, a semi-supervised model has been proposed and the experiment results show that unlabeled data boost the classification accuracy by 11% in comparison to its supervised version. Finally, a Federated Learning (FL) based Intrusion Detection System (IDS) has been proposed. It allowed the clients to learn an efficient intrusion detection model without the need to label their local data as well as to achieve high classification performance and improvement in terms of communication overhead (reduction by almost 75%).