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

Ons Aouedi, Kandaraj Piamrat, Dhruvjyoti Bagadthey

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

Ons Aouedi[a], Kandaraj Piamrat[a,*], Dhruvjyoti Bagadthey[b]

[a] *University of Nantes, LS2N, 2 Chemin de la Houssinière, Nantes, France*
[b] *Department of Electrical Engineering, IIT Madras, Chennai 600036, Chennai, India*

Abstract

Network traffic analytics has become a crucial task in order to better understand and manage network resources, especially in the network softwarization era where the implementation of this concept can be done easily with network function virtualization. Currently, many approaches have been proposed to improve the performance of traffic classification. However, as new types of traffic emerge every day (and they are generally not labeled), this opens a new challenge to be handled. Moreover, the question of how to accurately classify traffic using a limited amount of labeled data or partially labeled data hence becomes another important concern. In fact, labeling data is often difficult and time-consuming. In order to tackle the previously described issues, we reformulate traffic classification into a semi-supervised learning where both supervised learning (using labeled data) and unsupervised learning (no label data) are combined. To do so, this paper presents a stacked sparse autoencoder (SSAE) based semi-supervised deep-learning model for traffic classification. The main motivations of this approach are: (i) unlabeled data is often abundant 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. To investigate the performance of our approach, an empirical study has been conducted on a real dataset and results indicate that using a large amount of unlabeled data in the SSAE pre-trained phase can improve significantly the classification performance of the whole model. The proposed approach is compared against other representative machine-learning and deep-learning models, which are Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), Multi-Layer Perceptron (MLP), eXtreme Gradient Boosting (XGBoost), and Autoencoder. Furthermore, we have also conducted experiments on a well-known dataset including encrypted traffic (containing only time-related features) to evaluate the generalization performance of the proposed model.

Keywords: Machine learning, deep learning, stacked sparse autoencoder, feature extraction, traffic classification, semi-supervised learning, partial information.

1. Introduction

Network traffic classification provides a wide variety of management in today's Internet, such as resource allocation, Quality of Service (QoS) provisioning, anomaly detection, etc. According to the latest Cisco forecast, by 2022 the number of devices connected to mobile networks will exceed the world's population, reaching 12.3 billion. Meanwhile, mobile data traffic will be 77 exabytes per month, which is 7 times that in 2017 [1]. Traditional techniques such as port-based classification and deep packet inspection (DPI) are becoming less efficient to handle and classify this heterogeneous traffic (i.e. application) [2] [3].

In this line, Machine Learning (ML) is opening the ways to develop network traffic classifiers, which achieve an acceptable trade-off between computation complexity and accuracy [4]. Most of the classifiers are based on supervised learning where only labeled data are used as well as this 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 [5]. Therefore, a semi-supervised learning that uses a large amount of unlabeled data together with a limited amount of labeled data in order to build a better learner is a promising solution and has attracted more and more attention in network traffic classification [6] [7].

Apart from the previous issue, building models using traditional ML is also bottle-necked by the amount of features engineering effort required since there are limits to how much human effort can be thrown at the problem

*Corresponding author
Email address: kandaraj.piamrat@univ-nantes.fr (Kandaraj Piamrat)

as well [8]. With this regard, Deep Learning (DL) has gained popularity in the machine learning community because of its unique nature for solving complex problems, and it outperforms the other machine-learning methods in several fields such as healthcare, computer vision, network resource management, and has shown success in network traffic classification [9] [10] [11]. DL provides a variety of algorithms that allows exploiting unlabeled data to learn useful patterns in an unsupervised manner; for example, an autoencoder is one of the most popular and most widely used models for feature extraction [12]. Specifically, Stacked Sparse Autoencoder (SSAE) is an efficient unsupervised feature learning algorithm. In fact, SSAE 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 [13] [14].

1.1. Motivations and Key challenges

Applying semi-supervised learning and deep-learning 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 architecture and model hyper-parameters that should learn robust features and classify new network traffic very well.

1.2. Key Contributions

Facing the above challenges, we propose our semi-supervised model, of which some preliminary results appeared in [6]. In this paper, we investigate deeper into the variable ratios of unlabeled data as well as their impact on the accuracy. Extensive experiments have been conducted to obtain a robust model, which is then compared against representative models using supervised machine-learning as well as deep-learning approaches. In brief, the contributions of this paper can be summarized as follows:

- A robust SSAE model based on both unlabeled and labeled traffic has been proposed.
- Performance evaluation and comparison against semi-supervised learning (e.g., Autoencoder) as well as supervised learning (well-known supervised models) have been conducted.
- Performance evaluation of the proposed model using both non-encrypted and encrypted network traffic.

The rest of the paper is organized as follows. Section 2 summarizes the work related to this paper. Section 3.1, firstly presents essential backgrounds for a better comprehension of this work, then Section 3.2 introduces our algorithm for traffic classification and feature extraction based on semi-supervised deep-learning. Experimental settings and results as well as the datasets are presented in Section 4. Discussion and analysis of the results are provided in Section 5. Finally, conclusions and future works are given in Section 6.

2. Related Work

This section first provides an overview of state-of-the-art methods that adopt the Stacked Sparse AutoEncoder (SSAE) in several domains. Then, it also reviews recent achievements of state-of-the-art approaches that have been proposed to solve the traffic classification problem using ML/DL models.

2.1. SSAE related work

Sagheer and Kotb [14] used LSTM-stacked autoencoders for unsupervised time-series event prediction including bike rent demands and air quality. The proposed approach is tested and validated using two different case studies and two public datasets. Their results show that the unsupervised pre-training approach improves the performance and leads to better and faster convergence than Deep LSTM.

Xiao et al. [15] proposed a semi-supervised deep-learning strategy called the stacked sparse auto-encoder (SSAE) to classify and predict cancer tumor. The proposed SSAE-based method employs the pre-training layer approach and a sparsity penalty term to capture and extract important information from the high-dimensional data and then classify the samples. The proposed SSAE model was tested on three public RNA-seq data sets of three types of cancers. The proposed SSAE-based semi-supervised learning model achieves the best classifications compared with several commonly used classification methods such as Autoencoder, SVM, and Random Forest.

Sun et al. [16] proposed SSAE-DNN model using SSAE combined with DNN for bearing fault diagnosis. First, the sparse denoising autoencoder was used to learn more robust features from the unlabeled vibration data. Then, the learnt feature representations were used to train a DNN classifier for identifying induction motor running status. A comparison with traditional neural network shows that SSAE-based DNN achieves superior performance for feature learning and classification in induction motor fault diagnosis.

Yan and Han [17] applied the stacked sparse autoencoder to generate a low-dimension feature subset, which is then used with three basic classifiers (KNN, Random Forest, and SVM). Also, several hyper-parameters (sparse constraint, hidden layers, number of output layer neurons) have been tested in order to improve the performance of the final model.

2.2. Network traffic classification related work

Most of the existing traffic classification solutions are based on supervised learning algorithms [18] and have focused only on the totally labeled data without taking advantage of the unlabeled data. Consequently, in such conditions, the models tend to learn over-fit spaces to the labeled samples. In the following, we briefly review some representative contributions in the literature that are used to solve the traffic classification problems using ML/DL models.

Uddin and Nadeem [19] introduced a traffic classification framework, called TrafficVision that identifies the applications and their corresponding flow-type in real-time. *TrafficVision* is deployed on the controller, and one of the kernel modules of TrafficVision is named TV engine, which has three major tasks: 1) Collecting, storing and extracting flow statistics and ground-truth training data from end devices and access devices. 2) Building the classifiers from the training data. 3) 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 TrafficVision framework. The classification task of the TV engine has two modules, which are application detection using Decision Tree (C5.0) and flow-type detection using KNN. However, this solution uses only totally labeled data.

Amaral et al. [20] presented a traffic classification architecture based on SDN environment deployed in an enterprise network using ML. In this architecture, the controller collected the flow statistics from the switches, and then the preprocessing step is used through principal component analysis (PCA). Then, several supervised classifiers were applied (Random Forest, Stochastic Gradient Boosting, and Extreme Gradient Boosting). The accuracy of each application is used as an evaluation metric. However, PCA assumes that the relations between variables are linear, which is not always the case. Also, only labeled data have been used during the training process.

Wang et al. [21] have developed encrypted data classification framework called *DataNet*, which is embedded in SDN home gateway. This classification was achieved through the use of several DL algorithms, which are multilayer perceptron (MLP), stacked autoencoder (SAE), and convolutional neural networks (CNN). Although this framework used several DL algorithms, it did not benefit from the unlabeled data.

Zhao et al. [22] proposed an unsupervised feature extraction and clustering algorithm to obtain pure clusters. To build feature extraction model, autoencoder has been deployed. Then, K-means has been applied as clustering algorithm on the extracted features in order to identify the unknown traffic. However, the experiments were carried out on private dataset with only non-labeled data.

In the network domain, Stacked Autoencoder (SAE) has recently been applied to the field of traffic classification. For instance, Zhang et al. [23] proposed a deep learning network in the SDN-based environment to classify the traffic to one of several classes (Bulk, Database, Interactive, Mail, Services, WWW, P2P, Attack, Games, Multimedia). It consists of a SAE and Softmax classifier. SAE was used as an unsupervised-learning based feature extractor and Softmax was used as a supervised classifier. The experimental results demonstrate that the proposed method outperforms SVM in terms of accuracy, recall, and F1-score. Also, Vincent et al. [24] proposed a DL-based traffic classification framework, namely, deep packet. Deep packet used deep-learning (CNN and SAE) for encrypted traffic classification working at packet-level. However, here only labeled data were used. Moreover, Li et al. [25] presented an Improved Stacked AutoEncoder (ISAE) model for network traffic classification. To do so, SAE and Bayesian theory have been used where the Bayesian probability is applied to predict the final parameters of each autoencoder. The performance of this model is tested on only six applications and using only labeled data. Last but not least, D'Angelo and Palmieri [26] proposed a new model using SAE with convolutional and recurrent neural network in order to extract the spatial and temporal features. Their solution was used for coarse-grained network traffic classification. However, to evaluate the performance of their solution, they used UNIBS-2009, which is quite an old dataset and can miss modern network behavior. Also, only labeled data have been used.

Furthermore, a semi-supervised learning framework of network traffic classification to QoS classes was proposed in [7]. 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, which is Laplacian SVM to classify the traffic flows of unknown applications. However, this solution uses shallow learning model and hence cannot scale very well.

The related work papers with their key contributions/findings and limitations are summarized in Table 1.

2.3. Novelties of this paper

The current literature shows that SSAE has been used to solve several problems in different domains and it gives a promising result as well as outperforms the performance of traditional machine learning algorithms. However, it is not yet been well investigated for traffic classification. Furthermore, as presented previously, most 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.

In addition, even a few semi-supervised models and SSAE-based models have been used in traffic classification; they did not study the impact of unlabeled ratio on the performance of the final model, neither the effect of hyper-parameters (i.e., dropout and corruption noise) on the generalization performance of the SSAE model nor

Table 1: Summary of related works

Ref.	Methods used	Contributions/Findings	Limitations
[19]	Decision Tree and KNN	An application detection module and flow-type detection module.	The classification rule is learned using only the labeled training data.
[20]	PCA, Random Forest, Stochastic Gradient Boosting, and Extreme Gradient Boosting	A method using PCA for dimensionality reduction and Decision tree-based models for classification.	Dimensionality reduction is done using the linear transformation technique and only labeled data have been used for model training.
[21]	Multilayer perceptron (MLP), stacked autoencoder (SAE), and convolutional neural networks (CNN)	A method using deep learning models to achieve more than 95% for all evaluation metrics.	Only the labeled data have been used to train the deep learning models.
[22]	Autoencoder and K-means	A method using autoencoder for features extraction and clustering algorithm for partitioning the traffic packets into clusters.	Only unsupervised learning has been used.
[23]	Stacked autoencoder (SAE)	The results demonstrate that the SAE model outperforms SVM in terms of all evaluation metrics.	Only labeled data have been used for model training as well as the comparison was carried out only with SVM.
[24]	SAE and Convolutional Neural Network (CNN)	A method using SAE and CNN for encrypted traffic classification.	Only labeled data have been used.
[25]	SAE and Bayesian theory	The results demonstrate that the improved stacked auto-encoder outperforms the traditional one in terms of classification accuracy.	Only labeled data have been used as well as the comparison has carried out only with traditional SAE (i.e., without Bayesian theory).
[26]	SAE, CNN, and Recurrent Neural Network (RNN)	A method using SAE with CNN and RNN to extract the spatial and temporal feature for traffic classification.	Only labeled data have been used and only four traffic classes were considered.
[7]	Wrapper feature selection and Laplacian SVM	A method using semi-supervised learning algorithm, which is Laplacian SVM to classify the traffic flows of unknown applications.	Using shallow learning model and hence cannot scale very well.

a comparative analysis with several machine-learning and deep-learning models, which have been provided in this study.

3. Proposition

This section presents the main components integrated into the proposition of this paper as well as the architecture of the proposed DL-based semi-supervised learning.

3.1. Components

To better understand our proposition, first of all, each concept of which compose our proposition is explained and the arguments of why they are deployed in the solution are provided.

3.1.1. Autoencoder

Autoencoder (AE) is an unsupervised three-layer neural network, including an *input layer*, a *hidden layer*, and an *output layer* (also referred to as reconstruction layer). The typical structure of AE is shown in Figure 1. More specifically, the encoder obtains the input and converts it into an abstraction, which is generally known as a *code*, then the input can be reconstructed from the code layer through the decoder. It uses non-linear hidden layers to perform dimensionality reduction [27]. Its working process consists of two core segments placed back-to-back that have the same number of layers.

1. The *encoder* takes input data and maps it to a hidden representation (code), when the hidden layer has a lesser dimension than the input data, the encoder reduces/compresses the initial data.
2. The *decoder* uses the hidden representation (code) to reconstruct the input.

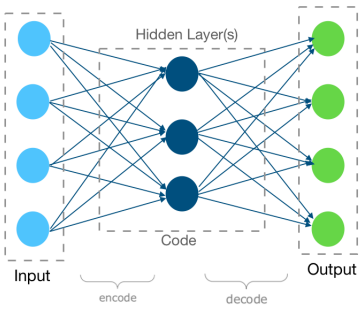


Figure 1: General autoencoder process

This structure is formulated as below where equation (1) presents the encoder and (2) the decoder respectively:

$$Z = f(W_1 X + b_1) \tag{1}$$

$$X' = f(W_2 Z + b_2) \tag{2}$$

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 is the weight matrix and bias between the second and the output layer; $f(.)$ 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 3, where the parameter set is denoted by $\theta = \{W_1, b_1, W_2, b_2\}$.

$$J(\theta) = \sum_{i=1}^{n} RE(x_i, x'_i) \tag{3}$$

We have deployed the concept of autoencoder because unlike supervised deep neural networks, it is an unsupervised feature learning neural network that can extract features from unlabeled data automatically.

3.1.2. Stacked Autoencoder

To obtain a better performance and learn more complex and abstract features than classical autoencoder, we deploy a more complex architecture and training procedure, known as stacked autoencoder (SAE) [24].

Several autoencoder layers are stacked together and form an unsupervised pre-training stage where the encoder layer computed by an autoencoder will be used as the input to its next autoencoder layer. Each layer in this stage is trained like an autoencoder 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 is added to the encoding layer of the unsupervised pre-training stage for the classification task and discarding the decoding layers of SAE (Figure 2).

3.1.3. Sparse Autoencoder

Since the number of units in hidden layers is large in the Stacked Autoencoder, we impose a sparse constraint on the hidden layers to 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", and thus the stacked autoencoder is converted into Stacked Sparse AutoEncoder (SSAE). 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:

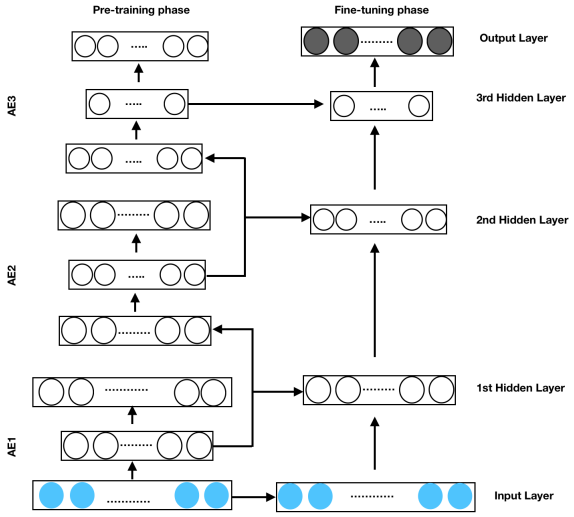


Figure 2: General Stacked Autoencoder process

$$\hat{\rho}_j = \frac{1}{n}\sum_{i=1}^{n}[f_j(x(i)] \tag{4}$$

$$\rho_{penalty} = \sum_{j=1}^{S} KL(\rho||\hat{\rho}_j) \tag{5}$$

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

$$KL(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j} \tag{6}$$

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:

$$J_{sparse}(\theta) = J(\theta) + \beta\sum_{j=1}^{S} KL(\rho||\hat{\rho}_j) \tag{7}$$

3.1.4. 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 [28]. The term "dropout" refers to dropping out units in a neural network (as shown in Figure 3). 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 [29].

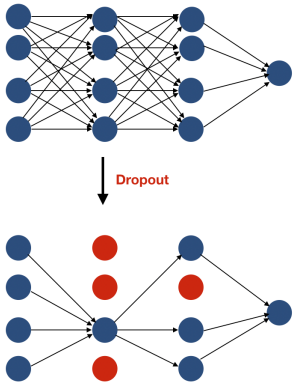


Figure 3: Dropout technique

In this study, the dropout technique is applied to train our semi-supervised learning in order to avoid the extraction of the same features repeatedly (overfit). It should be noted that the dropout is turned off during testing and used just within the training stage.

3.1.5. Denoising autoencoder

Denoising autoencoder was proposed in [30] to improve the robustness of feature representation. It is trained to reconstruct a clean input from a corrupted version of it (Figure 4). Therefore, similar to the conventional autoencoder network, it is trained in order to learn a hidden representation that allows it to reconstruct its input. However, the main difference with denoising autoencoder 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 this study, we set a certain percentage of random units of each sparse autoencoder (neurons) to zero (i.e., a fraction of the input is deleted randomly). To train a stacked sparse denoising autoencoder, each denoising sparse autoencoder 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"* [30].

The main reasons for deploying denoising SSAE are: (i) it is expected that a higher-level representation should be stable and robust under corruptions of the input, and (ii) performing the denoising can help our model to capture useful structure in the input data. The experiments demonstrate that the denoising autoencoders can improve the generalization performance of the network.

3.2. Methodology

In this section, we describe our proposed methodology: SSAE based semi-supervised method for traffic classification. Figure 5 presents the structure of the proposed

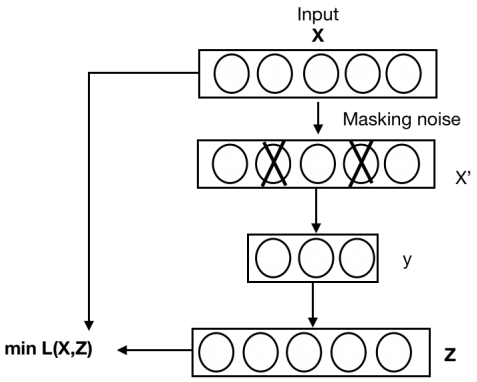


Figure 4: Training process of individual denoising autoencoders

model. It consists of the *unsupervised feature extraction task* and the *supervised classification task*. Therefore, both *unlabeled* and *labeled* data have been used to extract more valuable information and make a better classification. We present the methodology in the following.

3.2.1. Data Preprocessing

Our model uses flow and packet-based features (attribute) 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 [10]. These features 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.

Further, to reduce impacts from an imbalanced dataset of the labeled set [31], we create a new set with more balanced data samples for each application. For that, we have used a random *over-sampling* technique for the minority classes and a random *under-sampling* technique for the majority classes and we got a fair distribution of each class. As a result, the balanced subset has a total of 504,731 observations. Then, we have separated the re-sampled data into unlabeled and labeled sets, where the labeled set split into training (80%), validation (10%), and testing (10%).

When the flow has several features containing different data types whereas some ML models can only work with numeric values, it is hence necessary to convert or reassign numeric values. In this work, we have converted initial values of *timestamp* and *IP address* to numerical values. Moreover, when the dataset consists of different features with values in different scales, it needs to be scaled and to center the feature values. This can be done by calculating the standard scores for each data feature. The standard score x' of a data feature x is given by:

$$x' = \frac{x - \mu}{\sigma(x)}$$

where $\sigma(x)$ is the standard deviation and μ is the distribution mean value for x. Standardized features have approximately zero mean and unit standard deviation thus eliminating high variability and scaling effects.

3.2.2. Semi-supervised traffic classification

When the number of labeled data is limited compared to the unlabeled data, sometimes supervised models cannot obtain an accurate classification. A solution to overcome this limitation is to apply a semi-supervised classification. Semi-supervised learning uses unlabeled data for training-typically a limited amount of labeled data with a large amount of unlabeled data. In the network environment, the data X=[X_l, X_u] is composed of labeled data X_l and unlabeled data X_u. Semi-supervised learning classification problem benefits from unlabeled data to extract relevant information for new data discrimination X_{new}. This problem is showed in Table 2.

Table 2: Classification problem based semi-supervised learning

Learning Model	**Training data**	**Test data**
Supervised learning	$\{(X_l, y)\}$	X_{new}
Semi-supervised learning	$\{(X_l, y), X_u\}$	X_{new}

By taking advantage of unlabeled and labeled data, a semi-supervised classification model has been proposed as illustrated in Figure 5. Our semi-supervised classification model consists of (i) the unsupervised feature extraction stage using unlabeled data, and (ii) the supervised classification stage using labeled data. As shown in Figure 2, the unsupervised learning algorithm pre-trained in a bottom–up way. Then, the decoder layers of the SSAE model have been ignored and 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. In this study, the fine-tuning was done in a top-down fashion by training the pre-trained layers as a single model using a supervised learning process. The backpropagation algorithm is employed to get the gradient to update the parameters of the whole model.

In order to find the optimal model architecture, several experiments have been conducted with different architectures varying hidden layers and hidden nodes. Then, under different ratios of unlabeled data, we measure the performance of the proposed model. Next, using the optimal model architecture and optimal unlabeled data ratio, we also varied the sparse hyper-parameters in order to improve the performance of the SSAE model. Finally, to extract more robust features and prevent the over-fitting problem during the training process, we injected other hyper-parameters such as denoising coding and dropout.

4. Experimental study and results analysis

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

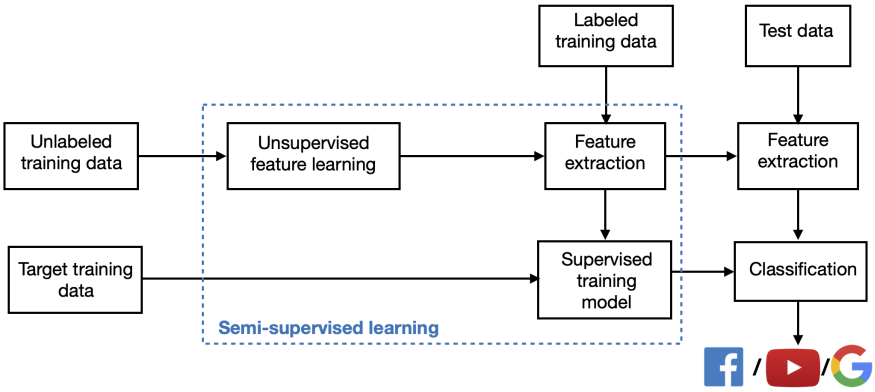


Figure 5: Structure of the semi-supervised network traffic classification model

4.1. Objectives

- Evaluate the performance of the SSAE under different ratios of unlabeled data.
- Evaluate the impact of the sparse parameter on the SSAE performance.
- Evaluate the impact of dropout and corruption noise hyper-parameters on the SSAE performance.
- Evaluate the impact of features extracted by the SSAE on the performance of other learning models.
- Evaluate SSAE against well-known models including simple autoencoder (deep-learning) and supervised machine-learning models that use 100% labeled data.
- Evaluate SSAE against supervised SSAE using the same architecture and hyper-parameters.
- Evaluate the classification performance on the well-known applications through 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)*.

4.2. Dataset

The dataset used in our experiment was presented in a research project [32]. It consists of 87 meta-data features, 3,577,296 instances, and 75 classes (Facebook, Google, YouTube, Yahoo, Dropbox, and so on)[1]. This dataset was collected in a network section from Universidad Del Cauca, Popayàn, Colombia. It was constructed by performing packet captures at different hours, during the morning and afternoon over six days in 2017. We choose 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. 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 404,528 flows and 54 applications. However, we used the balanced set that has a total of 504,731 observations.

As presented in our previous paper [6], the simulation of a partially-labeled dataset has been done through the selection of a portion of the known applications randomly and remove the application labels of their instances. Table 3 presents the statistical information (unlabeled/labeled observations along with train/validation/test split) used in this work after applying the re-sampling methods.

Table 3: The statistical information of dataset.

Unlabeled	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

4.3. Evaluation metrics

After the model training, it needs to be tested to verify its performance. To determine the quality of our proposed model and to evaluate the classification quality, several performance metrics have been used; time-related metrics (classification and training time), accuracy, F1-score, precision, and recall, which are calculated respectively as:

Accuracy is the proportion of correct classification (TP and TN) from the overall number of cases.

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

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

F1-Score is the harmonic mean of precision and recall. If, its value is high and closer to accuracy, the performance of classification is better.

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

where:

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

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

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.

4.4. SSAE based semi-supervised architecture and hyper-parameters

In this section, we study the impact of several hyper-parameters on the performance of the whole model. At 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 architecture 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 and as used in our previous work [6], 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 [33].

4.4.1. Trade-off between performance and unlabeled ratio

One of the importance concerns in this paper is to study the impact of the unlabeled data ratio on the performance of the proposed model. It is important to note that during this experiment, we fix the amount of labeled data and vary only the amount of unlabeled data (Table 3).

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.

$$R_u = \frac{nb\ unlabeled\ data}{nb\ labeled\ data} \quad (8)$$

The accuracy of the model while varying R_u is presented in Figure 6. 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 the unlabeled data provides more informative characteristics and deep learning can benefit from this data in the pre-training process and in turn improves the classification performance for unseen samples X_{new}. As a result, we use Ru=3.2 (where the number of unlabeled observations are 384,366) for the rest of the experiments .

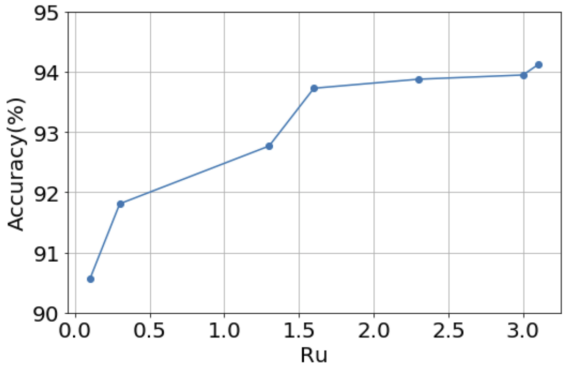


Figure 6: Performance of model with different unlabeled ratios.

4.4.2. Impact of the sparse parameter

Since the number of neurons in hidden layers is large, 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 for a better traffic classification. As presented in Figure 7, 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 SSAE model gives the best accuracy (94.40%) and training time. Larger than this value, the training time of the model begins to increase.

4.4.3. Impact of dropout and denoising coding

Although SSAE works well, we can further improve its generalization performance through other hyper-parameters. Our previous work [6] shows that dropout and denoising code hyper-parameters can improve the performance of the classification. As presented in Figure 8 and Figure 9, 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 equal to 0.02 and 0.02 respectively. Based on the results shown in these figures, it can be seen that

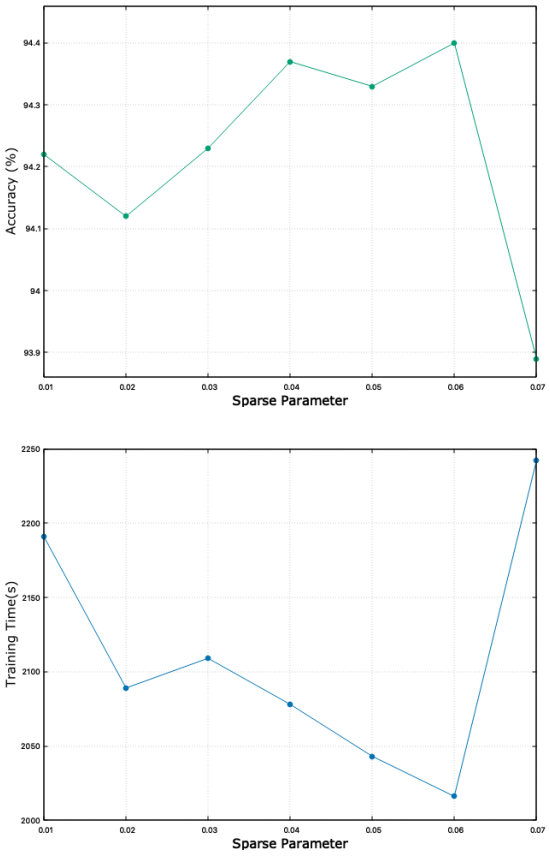


Figure 7: Accuracy and training time of different sparse parameter

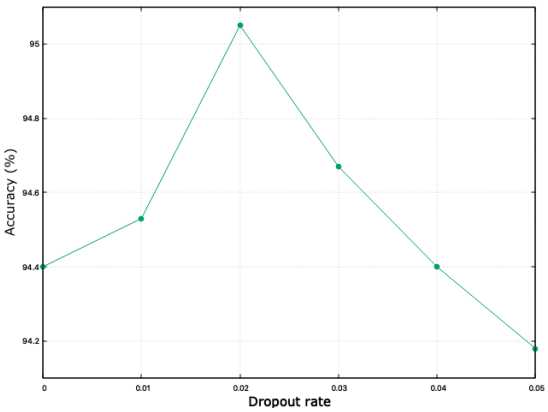


Figure 8: Effect of dropout

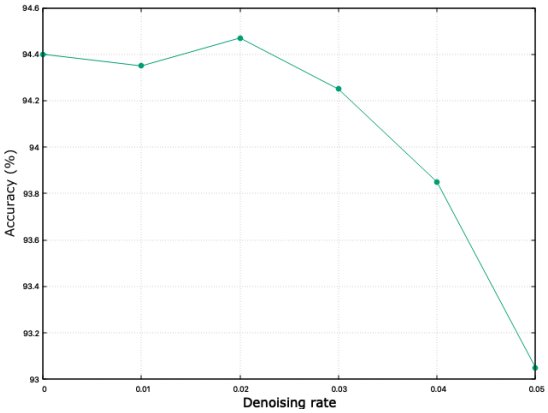


Figure 9: Effect of denoising coding

the SSAE has the ability to restore a good reconstruction from a corrupted input version even with a high corruption level. Moreover, we can interpret that too much dropout can decrease the classification performance.

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 10. 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.

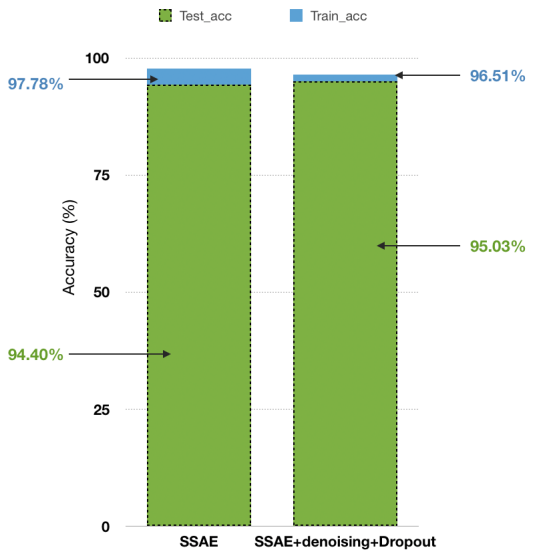


Figure 10: Accuracy of our model without/with enforcement (dropout/denoising)

4.5. Comparison Analysis

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

4.5.1. Comparison with semi-supervised learning models

To verify the classification efficiency of the proposed model labeled as SSAE* (for SSAE+denoising+dropout),

we compared it to four reference ML classification algorithms namely Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), and eXtreme Gradient Boosting (XGBoost) as well as simple SSAE (without dropout+denoising). To create semi-supervised models *(SSAE+DT, SSAE+RF, SSAE+SVM, SSAE+XGBoost)*, these algorithms are built on top of our proposed model (feature extraction part) and benefit from its automatic feature extraction and unlabeled data. In fact, the labeled data is passed through the SSAE that is trained by the unlabeled data and obtain 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.

Furthermore, we also used the autoencoder (*AE*) model for the comparison as it has a deep-learning architecture similar to SSAE. After learning with the autoencoder (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 11, the autoencoder here reconstructs the input then classifies through the encoder part. It should be noted that the used AE has the same network structures as SSAEs except that there is no sparse constraint as well as the noise and dropout hyper-parameters.

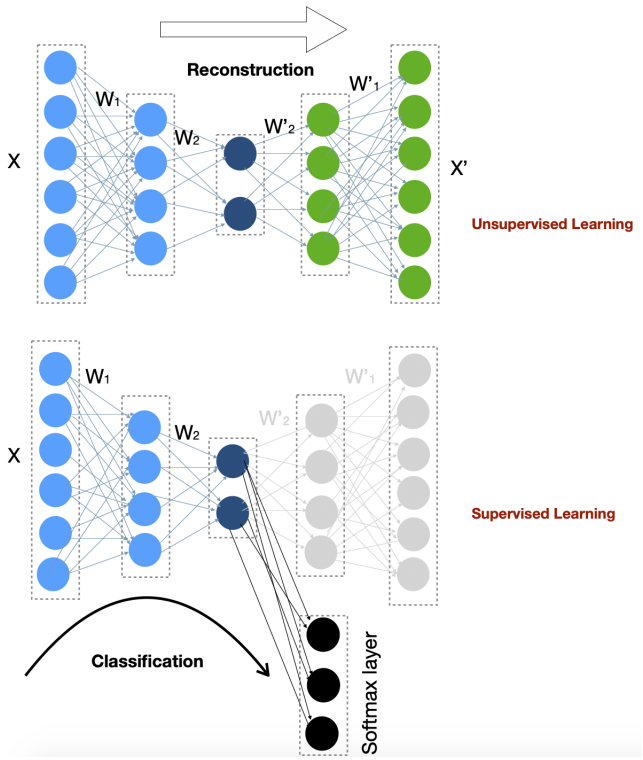


Figure 11: Classification process with Autoencoder

Table 4 summarizes the experimental results of the five semi-supervised models (previously described), SSAE, and SSAE*. It can be seen that our proposed model outperforms every 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, it can be seen that deep-learning based models (i.e., SSAE and AE) perform better than the conventional machine learning models (i.e., RF, DT, XGBoost, and SVM). 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 each AEs layers (Figure 2) and the sparse constraint used with the SSAE models.

Table 4: Comparison with semi-supervised models.

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**

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

To verify the efficiency of our model for traffic classification, we also compared it with five references supervised classification algorithms including: (i) simple classifiers, which are Decision Tree and Support Vector Machine (SVM), (ii) ensemble learning such as XGBoost and Random Forest, (iii) neural network classifier, which is Multi-layer Perceptron (MLP) classifier. We select these classifiers as our baselines because Random Forest and Decision Tree are easy to train [34], SVM is widely used and proved to be useful in several applications [35], 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 for their learning process and use all the original feature X. In fact, they used three times more labeled data (i.e., 100% labeled data) compared to our proposed model.

Table 5 presents the results achieved by our proposed model compared with the supervised classifiers. It is very clear that our proposed model outperforms the ensemble models (i.e., XGBoost and RF), simple deep-learning model (i.e., MLP) as well as the simple classifiers (SVM and DT). Specifically, although the competitive results of XGBoost with 100% labeled data, our model gets the best results with less amount of labeled data. It means that the proposed model with a limited labeled data can get competitive accuracy compared with the well-known supervised models. This may be attributed to the fact that

the proposed semi-supervised model based on SSAE generates deeply learned features that yield far superior results compared to the initial statistical features. Moreover, it uses pre-trained process that can boost the accuracy instead of the supervised models that trained from scratch using all labeled data. From this, we can conclude that our model is a robust model, extracts relevant features as well as can differentiate the applications very well.

Table 5: Comparison with supervised models.

Model	**Accuracy (%)**	**Precision (%)**	**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**

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

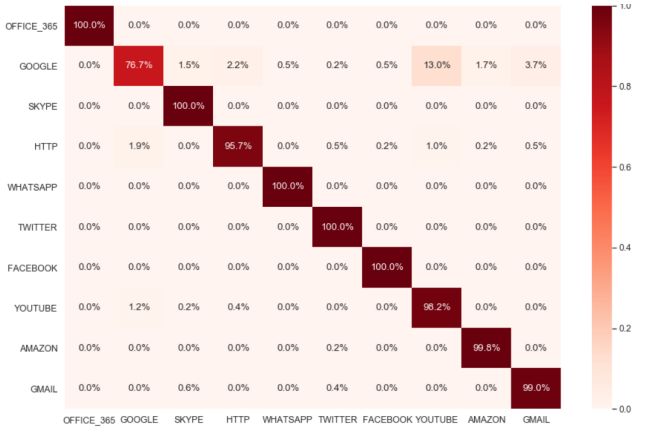
In order to further validate the efficiency of the proposed solution, which demonstrated the best performance against the semi-supervised model (using labeled and unlabeled data) and shallow supervised model (using more labeled data), we compared our model with itself in a supervised manner. In other 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 6 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 with the one using unlabeled data. This is because unlabeled data can provide informative characteristics and hence can boost the performance of traffic classification. This advantage will become even more significant in the case of larger training data.

Table 6: Comparison with supervised SSAE*.

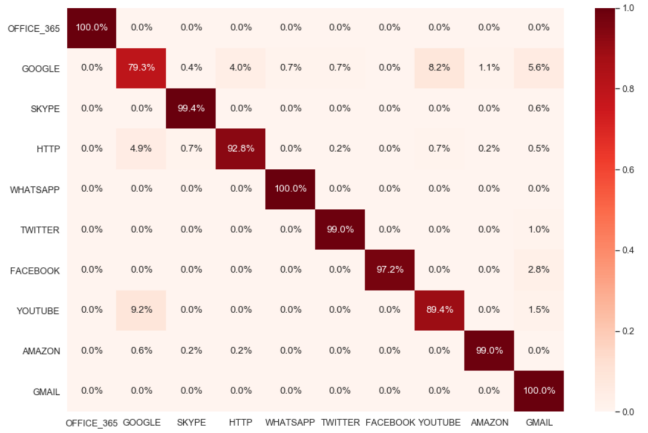
Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Supervised SSAE*	83.74	84.54	84.98	83.86
SSAE*	**95.03**	**96.40**	**94.01**	**94.40**

4.5.4. Confusion matrix (CM) comparison

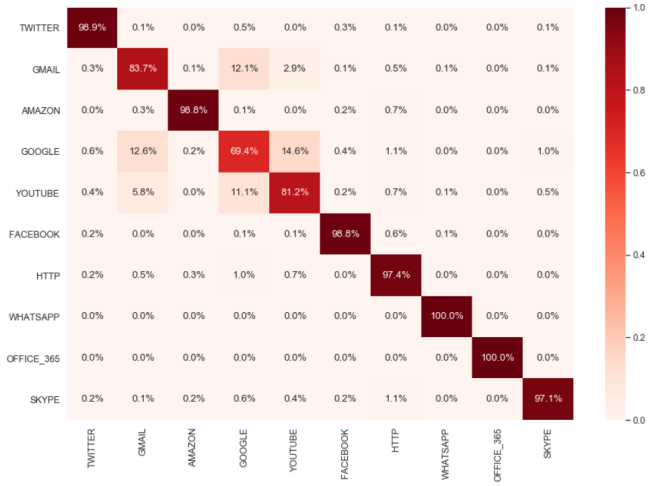
We now analyze a subset of results limited to some **well-known applications** using confusion matrix. CM compared the efficiency of the proposed model against two methods (one based on non deep-learning and another based on deep-learning). According to the above results, we selected XGBoost as supervised (non-deep learning)



(a) Confusion matrix for the proposed model



(b) Confusion matrix for Autoencoder



(c) Confusion matrix for XGBoost

Figure 12: A confusion matrix of the proposed model against AE and XGBoost under the most popular applications.

classifiers because it gives the best results and we select Autoencoder (deep learning) as a semi-supervised model.

Figure 12 shows the classification results of these 3 models using CM where columns correspond to the true labels, rows refer to the predicted labels, and the elements on the diagonal present the accurately classified results. The CM provides information about the classes that are correctly or incorrectly classified and the type of misclassification.

It can be seen that with XGBoost, many Youtube and Gmail flows, are incorrectly classified as Google, and vice versa. Moreover, we can see some interesting confusions between Google and Youtube, Gmail and Google, and Facebook and Gmail with AE. Finally, our proposed model incorrectly classifies Google as Youtube but not the opposite. As a result, we can conclude that our proposed model provides a slightly better classification results compared to AE. It also performs slightly better than XGBoost but without the need to label all the data as with XGBoost.

4.5.5. Cost in terms of training and testing times

As a continuation from the previous subsection, we also compared the computational efficiency of the proposed model against XGBoost and Autoencoder (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 Intel® Core™ i5-7200U CPU@2.50GHz processor.**

As shown in Figure 13 and Figure 14 that in terms of training time, the deep-learning based model (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 3.2, 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 on the real-time utilisation of the classification process.

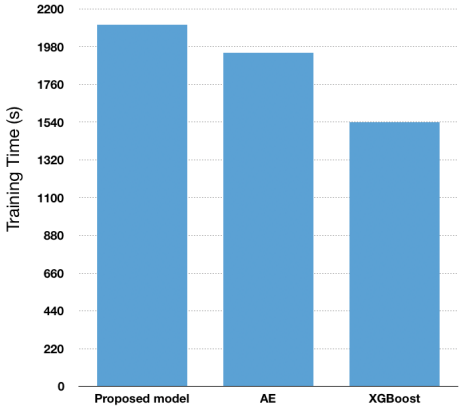


Figure 13: Training time comparison

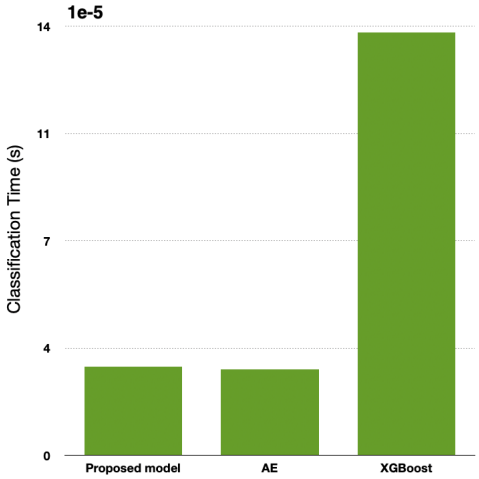


Figure 14: Classification time comparison per_sample

4.6. 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). This is one of the most popular encrypted traffic classification datasets. It contains only time-related features. For more details on the captured traffic and the traffic generation process, please refer to [36].

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 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 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 7 and Table 8 that with half amount of labeled observations the accuracy of our model can achieve accuracy over 88%, 84% for Sc_A and Sc_B, respectively. Also, it can be seen that with few labeled observations as well as the fewer amount of features, our model performs better than the simple classifiers like SVM and MLP. However, ensemble-based models (RF and XGBoost) using totally labeled data, give better accuracy.

In order to further validate the efficiency of our model, we compare it with some state-of-the-art approaches. The experimental results are presented in Table 9. It can be seen that the DL-based supervised model such as [37] and [38] outperforms all the approaches and specifically our model because they are more complex. However, this is not the case with the DT-based approach [36], where our model gives better results. This is maybe attributed to the

Table 7: 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 8: 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

deep architecture used in our case. Also, our model outperforms the DL-based semi-supervised learning proposed by [39], and this is because of the pre-training process of each AEs layer as well the use of the hyper-parameters such as denoising and dropout rate.

Table 9: The classification accuracy (%) of baseline and ensemble methods on VPN-nonVPN Dataset.

Approach	Model	Sc_A	Sc_D
Supervised([36])	DT	89.7	81.77
Supervised([37])	CNN-LSTM	99.7	91.7
Supervised([38])	Attention Based LSTM	99.7	91.2
Semi-supervised([39])	Multi-task model based CNN	N/A	80.67
Semi-supervised(SSAE*)	SSAE+NN	88.04	84.31

5. Discussion

In this study, we propose 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 limited amount of labeled to get an accuracy over 95%. Next, hyper-parameters tuning has been done in order to improve the 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 our model may be attributed to the use of all the data contained in the dataset as well as the layer-wise pre-training 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

- Our 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 improving 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 machine learning approach.

6. Conclusion

In this paper, a semi-supervised network traffic classification system based on Stacked Sparse Autoencoder (SSAE) using two real network dataset has been proposed. It extracts features from unlabeled data and trains the classification model with 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 the optimal performance based on the accuracy of the whole model. Next, dropout and denoising code hyper-parameters have been injected to improve the generalization performance of the SSAE.

The simulation results show that our 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), traditional machine learning models (DT, SVM), ensemble

learning models (RF and XGBoost), as well as supervised SSAE* (using only the labeled ratio). Moreover, we have evaluated this model for computation efficiency 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).

In the future, we will focus on further improvement by using Federated Learning as a way to collaboratively train learning model with privacy-preservation. Additionally, the model's training time can be further reduced by the implementation of the model in GPU acceleration.

References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper, Cisco., 2019.

[2] O. Barut, Y. Luo, T. Zhang, W. Li, P. Li, Netml: A challenge for network traffic analytics, arXiv preprint arXiv:2004.13006 (2020).

[3] H. Shi, H. Li, D. Zhang, C. Cheng, X. Cao, An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification, Computer Networks 132 (2018) 81–98.

[4] Y. Li, J. Li, Multiclassifier: A combination of DPI and ML for application-layer classification in SDN, in: Proceedings of the 2nd International Conference on Systems and Informatics (ICSAI), Shanghai, China, 2014, pp. 682–686.

[5] M. F. A. Hady, F. Schwenker, Semi-supervised learning, in: Handbook on Neural Information Processing, Springer, 2013, pp. 215–239.

[6] O. Aouedi, K. Piamrat, D. Bagadthey, A semi-supervised stacked autoencoder approach for network traffic classification, in: Proceedings of the 28th International Conference on Network Protocols (ICNP) HDR-Nets Workshop, Madrid, Spain, 2020.

[7] P. Wang, S.-C. Lin, M. Luo, A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs, in: Proceedings of the IEEE international conference on services computing (SCC), San Francisco, CA, USA, 2016, pp. 760–765.

[8] Q. Zhang, L. T. Yang, Z. Chen, P. Li, A survey on deep learning for big data, Information Fusion 42 (2018) 146–157.

[9] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, J. Aguilar, Towards the deployment of machine learning solutions in network traffic classification: A systematic survey, IEEE Communications Surveys & Tutorials 21 (2018) 1988–2014.

[10] M. Abbasi, A. Shahraki, A. Taherkordi, Deep learning for network traffic monitoring and analysis (ntma): A survey, Computer Communications (2021).

[11] T. De Schepper, M. Camelo, J. Famaey, S. Latré, Traffic classification at the radio spectrum level using deep learning models trained with synthetic data, International Journal of Network Management 30 (2020) e2100.

[12] M. Camelo, A. Shahid, J. Fontaine, F. A. P. de Figueiredo, E. De Poorter, I. Moerman, S. Latre, 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.

[13] D. Erhan, A. Courville, Y. Bengio, P. Vincent, 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.

[14] A. Sagheer, M. Kotb, Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems, Scientific Reports 9 (2019) 1–16.

[15] Y. Xiao, J. Wu, Z. Lin, X. Zhao, A semi-supervised deep learning method based on stacked sparse auto-encoder for cancer prediction using rna-seq data, Computer methods and programs in biomedicine 166 (2018) 99–105.

[16] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, X. Chen, A sparse auto-encoder-based deep neural network approach for induction motor faults classification, Measurement 89 (2016) 171–178.

[17] B. Yan, G. Han, Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system, IEEE Access 6 (2018) 41238–41248.

[18] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, Y. Liu, A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges, IEEE Communications Surveys & Tutorials 21 (2018) 393–430.

[19] M. Uddin, T. Nadeem, TrafficVision: A case for pushing software defined networks to wireless edges, in: Proceedings of the 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Brasilia, Brazil, 2016, pp. 37–46.

[20] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, H. S. Mamede, Machine learning in software defined networks: Data collection and traffic classification, in: Proceedings of the 24th International Conference on Network Protocols (ICNP), Singapore, 2016, pp. 1–5.

[21] P. Wang, F. Ye, X. Chen, Y. Qian, Datanet: Deep learning based encrypted network traffic classification in sdn home gateway, IEEE Access 6 (2018) 55380–55391.

[22] S. Zhao, Y. Zhang, Y. Sang, Towards unknown traffic identification via embeddings and deep autoencoders, in: 2019 26th International Conference on Telecommunications (ICT), IEEE, 2019, pp. 85–89.

[23] C. Zhang, X. Wang, F. Li, Q. He, M. Huang, Deep learning-based network application classification for SDN, Transactions on Emerging Telecommunications Technologies 29 (2018).

[24] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th international conference on Machine learning, 2008, pp. 1096–1103.

[25] P. Li, Z. Chen, L. T. Yang, J. Gao, Q. Zhang, M. J. Deen, An improved stacked auto-encoder for network traffic flow classification, IEEE Network 32 (2018) 22–27.

[26] G. D'Angelo, F. Palmieri, Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction, Journal of Network and Computer Applications 173 (2021) 102890.

[27] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M. S. Lew, Deep learning for visual understanding: A review, Neurocomputing 187 (2016) 27–48.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (2014) 1929–1958.

[29] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv:1207.0580 (2012).

[30] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, L. Bottou, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion., Journal of machine learning research 11 (2010).

[31] H. He, E. A. Garcia, Learning from imbalanced data, IEEE Transactions on knowledge and data engineering 21 (2009) 1263–1284.

[32] J. S. Rojas, Á. R. Gallón, J. C. Corrales, Personalized service degradation policies on OTT applications based on the consumption behavior of users, in: Proceedings of the International Conference on Computational Science and Its Applications, Melbourne, Australia, 2018, pp. 543–557.

[33] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classifica-

tion with deep convolutional neural networks, Communications of the ACM 60 (2017) 84–90.

[34] O. Aouedi, K. Piamrat, B. Parrein, Performance evaluation of feature selection and tree-based algorithms for traffic classification, in: 2021 IEEE International Conference on Communications (ICC) DDINS Workshop, Montreal, Canada, 2021.

[35] I. F. Kilincer, F. Ertam, A. Sengur, Machine learning methods for cyber security intrusion detection: Datasets and comparative study, Computer Networks (2021) 107840.

[36] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, A. A. Ghorbani, Characterization of encrypted and vpn traffic using time-related, in: Proceedings of the 2nd international conference on information systems security and privacy (ICISSP), 2016, pp. 407–414.

[37] K. Lin, X. Xu, H. Gao, Tscrnn: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of iiot, Computer Networks 190 (2021) 107974.

[38] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, S. Yu, Identification of encrypted traffic through attention mechanism based long short term memory, IEEE Transactions on Big Data (2019).

[39] S. Rezaei, X. Liu, Multitask learning for network traffic classification, in: 2020 29th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2020, pp. 1–9.