A Semi-supervised Stacked Autoencoder Approach for Network Traffic Classification

Ons Aouedi
University of Nantes, LS2N
2 Chemin de la Houssinière
Nantes, France
ons.aouedi@ls2n.fr

Kandaraj Piamrat
University of Nantes, LS2N
2 Chemin de la Houssinière
Nantes, France
kandaraj.piamrat@ls2n.fr

Dhruvjyoti Bagadthey
Department of Electrical Engineering
IIT Madras, Chennai 600036
Chennai, India
djbagadthey@gmail.com

***Abstract*—Network traffic classification is an important task in modern communications. Several approaches have been proposed to improve the performance of differentiating among applications. However, most of them are based on supervised learning where only labeled data are used. In reality, a lot of datasets are partially labeled due to many reasons and unlabeled portions of the data, which can also provide informative characteristics, are ignored. To handle this issue, we propose a semi-supervised approach based on deep learning. We deployed deep learning because of its unique nature for solving problems, and its ability to take into account both labeled and unlabeled data. Moreover, it can also integrate feature extraction and classification into a single model. To achieve these goals, we propose an approach using stacked sparse autoencoder (SSAE) accompanied by denoising and dropout techniques to improve the robustness of extracted features and prevent the over-fitting problem during the training process. The obtained results demonstrate a better performance than traditional models while keeping the whole procedure automated.**

***Index Terms*—Traffic classification, Feature extraction, Deep learning, Machine learning , Stacked Autoencoder, Stacked Denoising Autoencoder, Dropout, Semi-supervised learning.**

I. Introduction

Traffic classification is one of the principal issues in automated intrusion detection systems since learning user behaviors can help network operators to know what is flowing in their network so they can provide appropriate services. Therefore, traffic classification has profound implications on network management for Internet Service providers (ISP) [1]. 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 [2]. Traditional techniques such as port-based classification and deep packet inspection are becoming less efficient to handle and classify this heterogeneous traffic (i.e. application).

Over the past few years, machine learning (ML) and especially deep learning (DL) have advanced considerably and being widely adopted in several domains. In contrast to traditional ML models that rely heavily on features defined by domain experts, DL algorithms can extract knowledge from data through multiple layers of nonlinear processing units. Also, in most research on traffic classification, only the labeled data can be used, while many unlabeled data (i.e unknown applications) are not considered. In fact, it is difficult and time-consuming to label all the existing applications of the dataset manually; therefore, there may exist a large number of unlabeled instances within the dataset. At the same time, as new applications emerge every day, it is also not possible to have all the flow labeled in a real-time manner. Within the field of ML/DL, a broad distinction could be made between supervised, unsupervised, and semi-supervised learning. The main idea behind *supervised learning* is to identify a mapping from the input features to an output class. On the other hand, for *unsupervised learning* the objective is to find a structure (i.e. pattern) in the inputs without the output class. *Semi-supervised* is a combination of supervised and unsupervised approaches and is used when the dataset consists of input-output pairs but the outputs values are not known for certain observations. This reflects the situation of most of the network datasets. To handle this important issue, this paper presents a stacked sparse autoencoder (SSAE) based semi-supervised deep learning model for traffic classification. In this context, unsupervised feature extraction and a supervised classification algorithm are combined so as to utilize information from both unlabeled and labeled data. In brief, the advantages of our approach compared to other schemes are stated as follows.

- Our model takes advantage of both labeled and unlabeled data to implement a classification task. Making use of unlabeled data is of significance for the network-traffic classification.
- By taking advantage of unsupervised learning, our model can extract robust features automatically; hence, there is no need for an expert to extract features manually. Moreover, the dropout and denoising coding hyper-parameters are introduced into the model to avoid the over-fitting problem and extract robust features.

The rest of the paper is organized as follows. Section II,

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

provides related works while Section III presents the essential background on deep learning, autoencoder, and stacked autoencoder. Section IV introduces our algorithm of traffic classification based on semi-supervised deep learning. Section V discusses the details of the experiments, their corresponding results, and the dataset used during this work. Finally, a conclusion of this paper is given in section VI.

II. Related Work

As the importance of traffic classification increases, several techniques have been proposed and used over the years. Some representative techniques are summarized below.

A. Port based traffic classification

Port-based classification is the most simple technique since an analysis of the packet header is used to identify only the port number and its correspondence to the well-known port numbers. Network applications register their ports via the Internet Assigned Network Authority (IANA). However, this approach has limitations, for example, applications can use dynamic port number or ports associated with other protocols to hide from network security tools [3].

B. Deep Packet Inspection (DPI)

To avoid total reliance on the semantics of port numbers, the DPI technique has been proposed to inspect the payload of the packets searching for patterns that identify the application. It checks all packets data, which consumes a lot of CPU resources and can cause a scalability problem. Moreover, it fails to classify encrypted traffic [4].

C. ML-based traffic classification

The emergence of new applications, as well as the development of smart devices, has increased the complexity and diversity of online applications, which makes the traffic classification a difficult task. In this context, many research works have already used ML methods in network application classification in order to avoid the limitation of DPI and port-based traffic classification [1]. It is used as an alternative approach to classify the traffic by exploiting the distinctive characteristics of applications when they communicate on a network. Therefore, several works have been done using ML.

In [5], the authors introduced a traffic classification architecture based on software-defined networking (SDN) deployed in an enterprise network using different ML methods i.e. Random Forest, Stochastic Gradient Boosting, and Extreme Gradient Boosting. Before the classification task, they used Principal component analysis (PCA) as a dimensionality reduction technique to reduce over-fitting of the classifier. In [6], two feature selection methods have been used for traffic classification followed by six supervised learning, which are Naive Bayes, Bayes Net, Random Forest, Decision Tree, Naive Bayes Tree, and Multilayer Perceptron. In [7], the authors have used two common machine learning algorithms, which are C4.5 and KNN (K-nearest neighbor), for traffic classification. Also, the authors in [19] used the Symmetric uncertainty feature selection method then XGBoost for traffic classification.

It can be noticed that many works have been investigated using ML; however, most of them select the network traffic features through the conventional feature selection method and have focused only on labeled data and supervised learning. In contrast, this paper focuses on semi-supervised learning to benefit from labeled and unlabeled data and integrates the feature learning and model training in one architecture to avoid human intervention and time-wasting as maximum as possible.

III. Background

DL is a novel research direction in the ML field, which has achieved great success in many applications in comparison to classical ML algorithms. Building models using traditional ML is bottlenecked by the amount of features engineering required, there are limits to how much human effort can be thrown at the problem [8]. In contrast, DL algorithms hierarchically extract knowledge from raw data through multiple layers of nonlinear processing automatically, in order to make them flexible in modeling complex relationships [9]. In recent years, with the rapid growth of computational power and the availability of graphical processing units (GPUs), training DL algorithms has become easier. Here, one of the typical unsupervised feature learning algorithms, which is Stacked Sparse AutoEncoder (SSAE), is investigated. For a better understanding, we will describe in the following some important concepts.

A. Autoencoder

Autoencoder (AE) is one of the several architectures of artificial neural networks with a symmetrical structure. It is an unsupervised learning algorithm and can be divided into three parts (encoder, code, and decoder blocks) as shown in Fig. 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 [10].

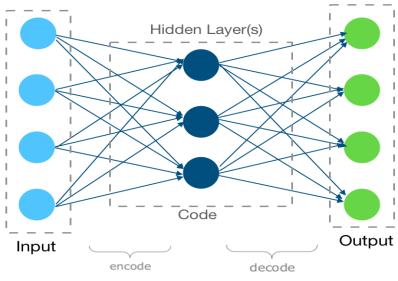

Fig. 1. General autoencoder process

In many cases, Autoencoder outperforms conventional feature selection methods and linear feature extraction such as principal component analysis (PCA) since it consists of several layers with non-linear activation functions to extract features intelligently [12].

B. Stacked Autoencoder (SAE)

To obtain a better performance than classical autoencoder, there exists a more complex architecture and training procedure, known as stacked autoencoder (SAE) [11]. 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 (Fig.2). Since the number of units in hidden layers is large, we impose a sparse constraint on the hidden layers to capture high-level representations of the data. A 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. Hence, each neuron in the hidden layer typically gets associated with a useful feature of the input. Generally, we define a neuron as "active" when its output value is close to 1 and "inactive" when its output value is close to 0. Consequently, by the use of sparsity constraint, we get an algorithm known as Stacked Sparse AutoEncoder (SSAE). Then, to prevent the identity transformation and avoiding the over-fitting, denoising noise and dropout hyper-parameters can be used in the SSAE.

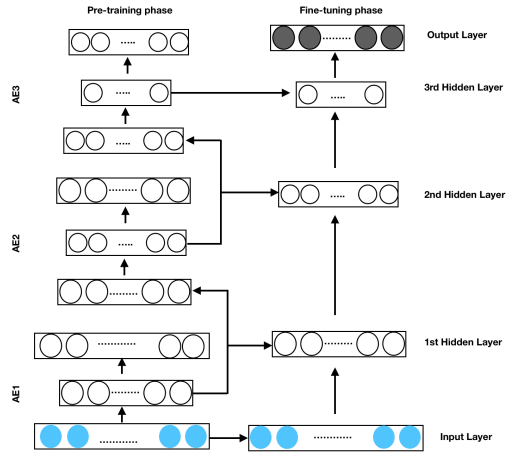

Fig. 2. General Stacked AutoEncoder process

C. Dropout

Dropout is a technique that aims to help a neural network model to learn more robust features and reduces the interdependent learning among the neurons [13]. The term "dropout" refers to dropping out units in a neural network. By dropping a unit 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. 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 (over-fit). It should be noted that the dropout is turned off during testing and used just within the training stage.

D. Denoising autoencoder

Denoising autoencoder was proposed in [14] to improve the robustness of feature representation. It is trained to reconstruct a clean input from a corrupted version of it. 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'. 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"* [14]. The main reasons for using denoising SSAE are: (i) it is expected that a higher-level representation should be rather 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.

IV. Methodology

In this work, we develop an SSAE based semi-supervised classification method for traffic classification as shown in Fig.3. 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.

A. Data Preprocessing

In order to simulate a partially-labeled dataset, we select a portion of the known applications randomly and remove the application labels of their instances. Further, to reduce impacts from an imbalanced dataset [16], we create a subset with more balanced data samples for each application. For that, we have used a random over-sampling technique for the minority classes and random under-sampling technique for the majority classes and we got a fair distribution of each class. In the experiment, we separate the labeled data into training (80%), validation (10%), and testing (10%).

When the dataset has several features containing different data types but some ML models can only work with numeric values, it is 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 the distribution mean value for x. Standardized features have approximately zero mean and unit standard deviation thus eliminating high variability and scaling effects.

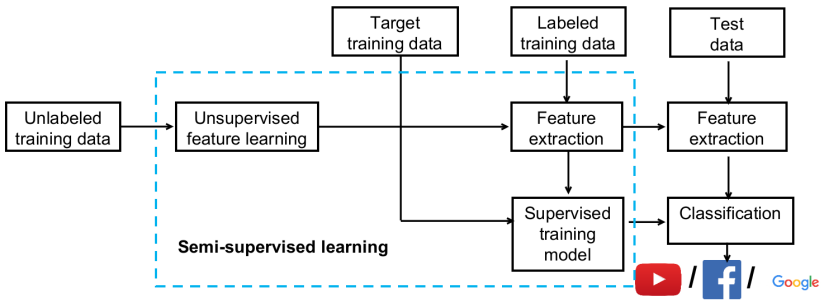

Fig. 3. Structure of the semi-supervised network traffic classification model

B. Semi-supervised traffic classification

By taking advantage of supervised and unsupervised learning, a semi-supervised classification model has been proposed as illustrated in Fig. 3. 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. To do so, 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. Then, the backpropagation algorithm is employed to fine-tune 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. Next, to improve the performance of our model, we also injected and varied hyper-parameters such as denoising coding and dropout.

C. Evaluation metrics

We evaluated the classification performance of our model in terms of several metrics: accuracy, precision, recall, and F-measure, which are calculated respectively as:

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

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

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

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

where:

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

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

To calculate these metrics, there are four important terms: TP: True Positive, FP: False Positive, TN: True Negative, FN: False Negative.

There are different evaluation models as training and testing on the same data: *k-fold cross-validation* and *train/validation/test split*. In this research, we used the train/validation/test split because it is simple and flexible.

V. Experimental Study

The purpose of this experiment is to compare the effect of our model with different parameters to find the optimal architecture. In this section, we assess the performance of our semi-supervised based deep learning against other classification algorithms. We have used Scikit-learn and Keras libraries, and Python as a programming language. The Python scripts used for implementing our model are made available online for public access at [18].

A. Dataset

The dataset used in our experiment was presented in a research project [15]. It consists of 87 features, 3,577,296 instances, and 78 classes (Facebook, Google, YouTube, Yahoo, Dropbox, and so on). 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 instances and 54 applications. Then, we have removed the target label of some traffic in order to be used as unlabeled data. In this context, we used 283,186 instances (70%) as labeled data and 121,342 instances (30%) as unlabeled data.

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

Our SSAE based semi-supervised model is a DL approach that consists of stacked sparse autoencoders and softmax classifier for unsupervised feature learning and classification, respectively. Therefore, to find the optimal unsupervised feature learning model, we need to find different hyper-parameters that maximize the accuracy of the classification. In this context, several experiments have been done, some of the best configurations are presented in Table I. For each model, the test accuracy, the number of hidden layers, as well as the number of neurons in each layer, are presented.

It can be seen that the configuration in bold text with four hidden layers (100-200-400-50) obtains the best results with a learning rate equal to 0.001. Similarly, for the selection

TABLE I
CONFIGURATIONS USED IN THE EXPERIMENTATION

SSAE Model	#Hidden layers	Number of neurons					Test accuracy
		L1	L2	L3	L4	L5	
SSAE 1	2	100	50	-	-	-	79.4%
SSAE 2	2	100	100	-	-	-	83%
SSAE 3	2	100	200	-	-	-	84.2%
SSAE 4	2	100	400	-	-	-	82%
SSAE 5	2	70	50	-	-	-	77.4%
SSAE 6	2	70	30	-	-	-	76.8%
SSAE 7	3	100	200	50	-	-	85.4%
SSAE 8	3	100	200	100	-	-	85.7%
SSAE 9	3	100	200	200	-	-	85.3%
SSAE 10	3	200	400	60	-	-	85.2%
SSAE 11	3	100	200	400	-	-	85.8%
SSAE 12	4	100	200	400	100	-	86.3%
SSAE 13	**4**	**100**	**200**	**400**	**50**	-	**86.89%**
SSAE 14	5	100	200	400	50	30	84.8%

of the sparsity parameter, we tested different values ranging from 0 to 0.1 with a step of 0.01. Then, we have selected the best configuration that maximizes the performance of our model (we stopped when the performance started to decrease). Besides, the selected activation of hidden layers is ReLU (rectified linear unit) because it shows better convergence performance than sigmoid and tanh [17].

To improve the performance of our model, we combine the sparse coding with denoising coding in the autoencoder to learn robust features for the NN classifier (i.e. Softmax layer), and we used a dropout technique to overcome over-fitting during the training process. As presented in Fig. 4 and Fig. 5, we have tested the effect 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 showed that the best classification performance was obtained at a dropout rate and corruption equal to 0.02 and 0.03 respectively. Based on the results shown in these figures, we can interpret that heavy corruption noise and too much dropout can decrease the classification performance.

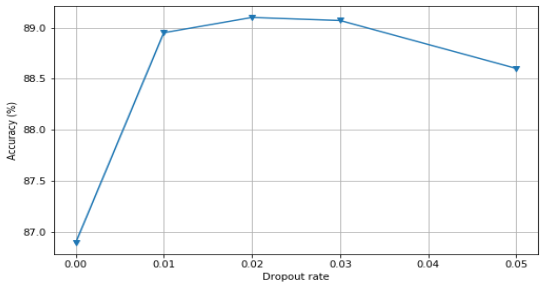

Fig. 4. Effect of dropout

Moreover, a comparison has been carried out between the SSAE (used as a baseline) and SSAE+denoising+dropout, as presented in Fig. 6. It can be seen that the injection of dropout and denoising hyper-parameters can improve classification performance. This implies that our model is more stable and generates more relevant information with these hyper-parameters.

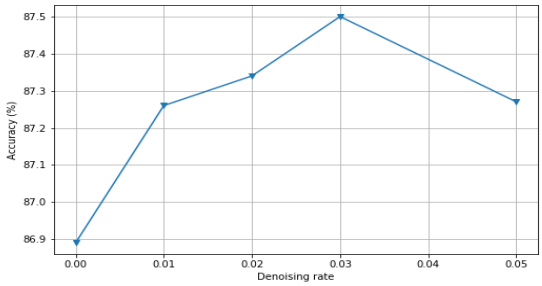

Fig. 5. Effect of denoising coding

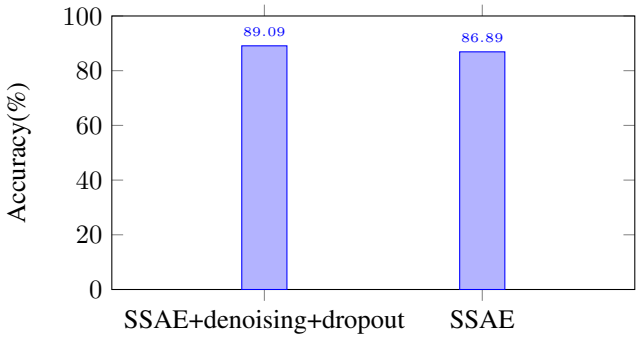

Fig. 6. Accuracy of our model with/without enforcement (dropout/denoising).

C. Comparison with traditional approaches

To verify the efficiency of our model for traffic classification, we compared it with three reference classification algorithms namely decision tree (DT), random forest (RF), and support vector machine (SVM). These algorithms are built on top of our SSAE and benefit from its automatic feature extraction. However, there is no fine-tuning process of features extracted by SSAE and their learning process only uses labeled data. In fact, the labeled data is passed through the SSAE to obtain X', the transformed data. Since the last layer of our model has 50 neurons only, this X' has a smaller dimension than X (i.e 87 features). Finally, these features are used to be

TABLE II
COMPARISON OF ML CLASSIFICATION RESULTS

Model	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
SSAE+RF	87.13	88.54	87.13	87.49
SSAE+SVM	55	63.22	55	56.79
SSAE+DT	84.37	86.60	84.37	85.13
Our model	**89.09**	**89.51**	**88.35**	**89.05**

learned with the aforementioned classifiers.

The result shown in Table II that our proposed model outperforms the others in terms of accuracy, precision, and recall. Also, it achieves a good tradeoff between precision and recall (i.e. F-measure). Therefore, this verifies that our model is stable and can give satisfying results.

D. Discussion

The experimental results presented in the previous sections demonstrate that the proposed SSAE based semi-supervised deep learning can outperform the conventional classification algorithms. The results may be attributed to the layer-wise pre-training where each single AE is trained to exploit the relationship between high-level features and target outputs and helps the deep neural network models to yield much better local initialization than random initialization. Then, the global fine-tuning process optimizes the parameters of the entire model, which greatly improves the classification task. Also, the integration of dropout and denoising code hyper-parameters into our model improves the performance of classification and leads to learn more robust and informative features with minimal risk of over-fitting. Moreover, our model benefits from the unlabeled traffic in the training process to improve the classification performance over other conventional classifiers. Finally, our model can extract features automatically from network traffic and discover the nonlinear relationships behind the data as well as it can help to save the cost of using experts to extract good features from the traffic, which is one of the most important contributions of our model.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have presented a stacked sparse auto-encoder (SSAE) based semi-supervised classification model for network-traffic classification. This model is closer to reality as it profits through the use of labeled and unlabeled data. In this way, an unsupervised algorithm based on the SSAE is first pre-trained to obtain layer-by-layer initialization parameters. Then, a supervised neural network classifier is linked to the code layer of the SSAE and the whole is fine-tuned by backpropagation. Furthermore, to improve the performance of the feature extracted through our model and to avoid over-fitting, we injected dropout and denoising code hyper-parameters. Experimental results and analysis have shown that our model outperforms other conventional classifiers while benefiting from automatic feature extraction.

For future works, we plan to use a much larger amount of unlabeled data to verify its impact on the classification performance of our proposed model. Also, we can use other hyper-parameters to improve our model and test the time needed for the classification tasks.

REFERENCES

[1] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," IEEE communications surveys and tutorials, vol. 10, no. 4, pp. 56–76, 2008.

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

[3] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in network identification of P2P traffic using application signatures," In WWW2004, New York, NY, USA, May 2004.

[4] Z. A. Qazi et al., "Application-awareness in SDN," In Proceeding ACMSIGCOMM, Hong Kong, 2013, pp. 487–488.

[5] P. Amaral et al., "Machine learning in software defined networks: Data collection and traffic classification", IEEE International Conference on Network Protocols (ICNP), Singapore, Nov. 2016, pp.1–5.

[6] P. Perera , YC Tian , C. Fidge, and W. Kelly "A Comparison of Supervised Machine Learning Algorithms for Classification of Communications Network Traffic". In: Neural Information Processing, Springer, Cham, Lecture Notes in Computer Science, 2017, pp 445–454,

[7] GD. Gil, AH. Lashkari, M. Mamun, and AA. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features". In: Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), 2016, pp 407–414.

[8] N. Ketkar, "Deep Learning with Python" Apress, pp 159–194, 2017.

[9] G.P. Zhang, "Neural networks for classification: a survey", IEEE Trans.Syst. Man Cybern. C, Appl. Rev., 30, (4), pp. 451–462, 2000.

[10] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M.S. Lew, "Deep learning for visual understanding: a review", Neuro computing 187, 27–48, 2016.

[11] P. Vincent, H. Larochelle, Y. Bengio, and PA. Manzagol, "Extracting and composing robust features with denoising autoencoders". In: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp 1096–1103.

[12] O. Aouedi, M.A.B. Tobji, and A. Abraham, "An Ensemble of Deep Auto-Encoders for Healthcare Monitoring," in Hybrid Intelligent Systems. Springer International Publishing, mar 2019, pp. 96–105.

[13] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, no. 1, 2014, pp. 1929–1958.

[14] P. Vincent, H. Larochelle, I. Lajoie, et al. "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion". Journal of Machine Learning Research, Vol. 11, 2010, p. 3371-3408.

[15] J.S. Rojas, A. Rendon, and J. Corral, "Personalized Service Degradation Policies on OTT Applications Based on the Consumption Behavior of Users", International Conference on Computational Science and Its Applications, 2018, pp. 543–557.

[16] H. He and E. A. Garcia, "Learning from imbalanced data", IEEE Trans. Knowl. Data Eng., vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks". In NIPS, 2012.

[18] Dhruvjyoti Bagadthey, Traffic_Classification, GitHub, 08 July 2020, [Online]. Available: https://github.com/Dhruvjyoti-Bagadthey/Traffic_Classification

[19] I. L. Cherif and A. Kortebi, "On using eXtreme Gradient Boosting (XGBoost) Machine Learning algorithm for Home Network Traffic Classification," in 2019 Wireless Days (WD), 2019-April, pp. 1–6