Original scientific paper

A Comparative Study of Deep Learning and Decision Tree Based Ensemble Learning Algorithms for Network Traffic Identification

Nedeljko Nikolić, Slavica Tomović, *Member, IEEE*, and Igor Radusinović*, Member, IEEE*

Abstract — **In this paper, we apply Deep Learning (DL) and decision-tree-based ensemble learning algorithms to classify network traffic by application. Various Deep Learning (DL) models for network traffic identification have been presented, implemented and compared, including 1D convolutional, stacked autoencoder, multi-layer perceptron, and combination of the aforementioned. Then the results of DL models have been compared to those obtained with two popular ensemble learning models based on decision trees - Random Forest and XGBoost. To train and test the classification models, a dataset containing both encrypted and unencrypted traffic has been collected in a real network, under normal operating conditions, and pre-processed in a way that ensures non-biased results. The classification uncertainties of the models have been also quantified on publicly available ISCX VPN-nonVPN dataset. The models have been compared in terms of precision, recall, F1 score and accuracy, for different levels of complexity and training dataset sizes. The evaluation results indicate that the decision-tree ensemble learning algorithms provide more accurate results and outperform the DL algorithms. The performance gap reduces with the dataset complexity.**

Keywords — **computer networks, decision-tree algorithms, deep learning, ensemble machine learning, traffic identification.**

I. Introduction

Accurate and real-time traffic identification is of increasing importance for modern network management. The ability to promptly recognize different applications in the network is especially relevant for enforcing Quality of Service (QoS) requirements, identifying anomalies, applying routing policies, etc. [1]-[3].

Due to their high importance, traffic identification techniques have been widely studied for more than two decades [4]. One of the basic approaches is to identify network applications by their registered port numbers (defined by the Internet Assigned Authority - IANA) in TCP or UDP header [3]. By the time, this approach has become unreliable because many applications today use unregistered ports or even random generated ports that can overlap with ports used by other applications [4]. The presence of NAT (Network Address Translation) routers is another limiting factor. According to some studies [5], port-based methods can classify only 30-70% of the current Internet traffic. The aforementioned issues have been addressed by Deep Packet Inspection (DPI) approaches, which analyze the whole packet payload and match it with a set of manually identified patterns and keywords (so-called signatures) to determine a traffic class the packet belongs to. DPI approach currently provides the best classification accuracy, and thus it is widely used in systems for network traffic analysis [4]. However, DPI requires an up-to-date database of signatures to be maintained, and manual extraction of signatures for each new protocol is time-consuming and labor-intensive [6]. Furthermore, DPI cannot deal with proprietary protocols and has limited usability when it comes to encrypted traffic [4]. These limitations have been addressed by statistics-based approaches which rely on statistical features of the entire traffic flow to make a classification decision. As statistics are related to packet lengths, inter-arrival times, flow rates, packet header content, etc., they can also be calculated for encrypted packets [7]. However, statistics-based classification cannot be performed in real-time, i.e., after observing only the flow beginning [4].

Recent developments of Deep Learning (DL) have motivated researchers to apply advanced DL capabilities in communication networks [8], [9]. Since DL techniques have the ability to automatically learn complex patterns in data, various deep Neural Network (NN) models have been proposed to classify both unencrypted and encrypted data traffic at the application level [4], [10]. Once DL models are trained to associate pre-defined application labels to network packets, they can be relatively easily upgraded and re-trained with the transfer learning method to recognize new applications, without the need to engage DPI experts [11]. However, from the current literature, it is hard to draw a conclusion about the most suitable deep NN architectures for network traffic identification because datasets and dataset pre-processing techniques used by individual researchers differ substantially. We have partially addressed this issue in [12], where the most commonly used deep NN architectures have been applied for network traffic identification and compared on the dataset collected in the Institute of Public Health in Montenegro. Here, we extend that analysis by providing more comprehensive results, obtained on two datasets, that

Paper received May 16, 2021; revised October 2, 2022; accepted October 2, 2022. Date of publication December 26, 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Grozdan Petrović.

This paper is revised and expanded version of the paper presented at the 29th Telecommunications Forum TELFOR 2021 [12].

Nedeljko Nikolić is with the Institute for Public Health, 81000 Podgorica, Montenegro (e-mail: nedeljko.nikolic@ijzg.me).

Slavica Tomović is with the Faculty of Electrical Engineering, University of Montenegro, Dzordza Vasingtona bb, 81000 Podgorica, Montenegro (e-mail: slavicat@ucg.ac.me).

Igor Radusinović is with the Faculty of Electrical Engineering, University of Montenegro, Dzordza Vasingtona bb, 81000 Podgorica, Montenegro (e-mail: igorr@ucg.ac.me).

reveal how the performance of various DL models is affected by the NN complexity, dataset size and a number of application classes. Moreover, we compare the DL models with two decision-tree-based ensemble machine learning methods – Random Forests and XGBoost. Interestingly, the study reveals that the decision-tree-based ensemble learning solutions outperform the analyzed DL algorithms. The performance gap decreases with the dataset complexity.

The rest of the paper is organized as follows. Section II provides a review of the most relevant DL models for traffic identification from the literature. In Section III we provide a brief overview of Random Forests and XGBoost machine learning algorithms. Section IV describes the datasets preparation and the training procedure. Section V presents results of the performance analysis. Conclusion remarks are given in Section VI.

II. Deep Learning Models for Network Traffic Identification

In this section, we present several most common deep NN models for *online* identification of network applications. These models take a data packet as input and produce a vector of *N* real numbers as an output, where N is the number of application classes. Since raw packet content cannot be fed into the NN models properly, packets are truncated or expanded with zero padding such that a fixed-size input is always provided to NN. The input is formatted as a vector of packet bytes. How this input will be mapped to the output vector depends on the NN architecture. In the rest of the section, we describe those most commonly used in the literature.

A. Multi-Layer Perceptron (MLP)

MLP is a class of feedforward NNs that consists of three or more layers. The first layer receives the input byte vector that needs to be classified. The classification result is contained in the last layer. One or more hidden layers, composed of multiple neurons, are placed between input and output layers, as illustrated in Fig. 1.

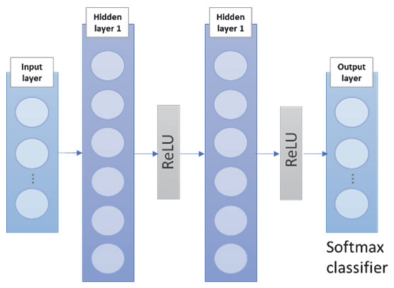

Fig. 1. MLP NN architecture.

The hidden layers represent the computational engine of the NN. The computations taking place in each hidden layer and output layer are as follows:

$$f(x) = \sigma(W^{(i)}x + b^{(i)}) \quad (1)$$

where σ is the activation function, x is input from the previous layer, $W^{(i)}$ is a weight matrix and $b^{(i)}$ is a bias vector for the i-th layer.

B. Stacked AutoEnconder (SAE)

As the name suggests, SAE architecture is composed of multiple autoencoders. Autoencoders are unsupervised NNs that are typically used for automatic feature extraction from the data. As illustrated in Fig. 2, the autoencoder is composed of three parts: encoder, bottleneck layer (or code), and decoder. The encoder extracts the main features from the input data and stores them in the bottleneck layer. The decoder attempts to reconstruct the original data by using features from the bottleneck layer. The bottleneck has a smaller number of neurons than the input layer. Thus, it limits the amount of information that can traverse the decoder, forcing a learned compression of the input data. Since both encoder and decoder are feed-forward NNs, autoencoder can be trained using standard procedures for feed-forward NNs. The optimization goal is to get output identical to the input. In classification tasks, the content of the bottleneck layer is used as input features [10].

In practice, using only one autoencoder usually is not sufficient to learn a good feature extraction function. Thus, multiple autoencoders are often stacked on top of one another, forming SAE architecture as illustrated in Fig. 2. In SAE, the bottleneck layer of each autoencoder is an input of the subsequent autoencoder in the stack. In the first phase, autoencoder layers are trained one at a time. Then, to achieve more accurate results, a backpropagation algorithm is applied to the whole stack and all layers' weights are fine-tuned [7]. For the traffic classification task, the final layer with *N* neurons and *SoftMax* activation function have to be appended to the last bottleneck layer.

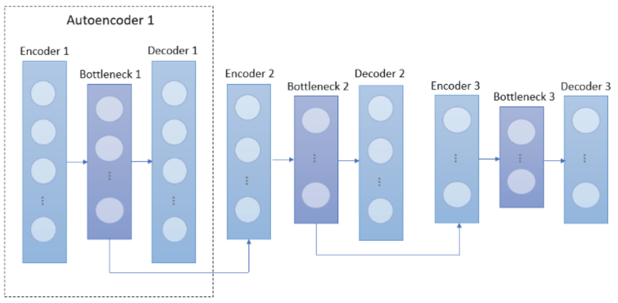

Fig. 2. SAE: encoder training.

C. Convolutional Neural Network (CNN)

CNN is a popular deep learning model which extracts features from input data by performing convolutional operations in each layer. Conventional CNNs have been originally designed to operate on 2D data such as images and videos, where they can efficiently capture spatial and temporal dependencies and extract complex feature maps by applying relevant kernels (convolution matrices). As an alternative, 1D CNNs have been recently developed [13]. In numerous applications dealing with 1D data, 1D CNNs proved to be advantageous over their 2D counterparts [13]. Unlike conventional CNNs, 1D CNNs process 1D data, and use 1D arrays for both kernels and feature maps. Thus, one of the most significant differences between 1D and 2D convolutions is in terms of computational complexity. This makes 1D CNN models much easier to train.

To perform the classification task, the features extracted by 1D CNN must be provided to one or more fully-connected layers. Fig. 3 shows the 1D CNN model for traffic classification which is used in our benchmarking study in Section V. Two convolutional layers are followed by a *max-pooling* layer, a *flatten* layer, and three *fully-connected* layers. In each 1D convolutional layer, the non-linearity is introduced with *ReLU* activation function. The *max-pooling* layer is used to reduce the dimensionality of features and avoid overfitting. A dropout is applied as an additional measure to avoid overfitting and improve generalization. Finally, the flattened 1D CNN output is provided as input to the stack of three fully-connected layers. The output layer uses the *SoftMax* activation function to enable classification [14].

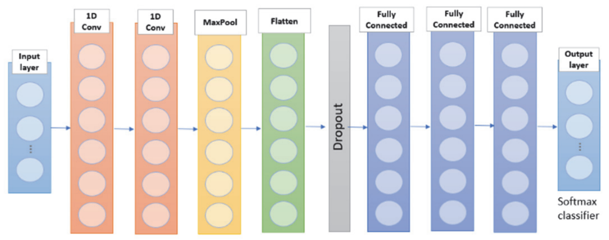

Fig. 3. 1D CNN-based model proposed in [14].

III. Decision-Tree Based Ensemble Learning Classification Models

Although DL techniques outperform the other ML methods in many fields today, it is well known that there is no single algorithm that works best across all possible scenarios. In particular, it has been shown that decision-tree-based ensemble machine learning methods often outperform DL solutions use-cases with structured data [15]. Therefore, we include Random Forest and eXtreme Gradient Boost (XGBoost) algorithms in our study as they are two of the best-performing decision-tree-based classification models.

A. Random Forest

Random Forest is an ensemble learning algorithm that operates by constructing a collection of uncorrelated decision trees at training time. In the context of the analyzed problem, each decision tree represents a network traffic classification model in a tree structure. Each node in the tree evaluates some feature of the input data, branches represent values the feature can take, while leaves contain the corresponding classification results. RF algorithm uses multiple trees to overcome the problem of overfitting. The final classification decision is the decision selected by most trees. The learning process constructs a decision tree step by step by identifying the feature that best divides the data at each node going down the tree. However, each decision tree is trained with a different, randomly chosen subset of the dataset and a randomly chosen subset of the features. In this way, the similarities between the decision trees are reduced, and the potential errors of individual trees are efficiently compensated.

B. eXtreme Gradient Boost (XGBoost)

XGBoost is the computationally efficient and highly accurate implementation of the gradient boosting decision-tree technique [16]. It builds an ensemble of decision-tree models incrementally, by adding one model at a time and by adjusting it to correct the classification errors made by previously added models. When adding new decision-trees, XGBoost uses the gradient descent algorithm to minimize an objective function that combines a convex loss function, which measures how well the model fits the training data, and a penalty for the model complexity. The decision-trees used by XGBoost are much smaller than those used by the Random Forest algorithm, and they usually perform slightly better than Random Forest in many classification problems with a low-to-moderate number of classes. However, the training process is much slower because each decision-tree in the ensemble is optimized. Hyperparameter tuning is also more complex.

IV. Dataset Preparation

For this analysis, a dataset is created from real traffic in the network of the Institute for Public Health in Montenegro. We will refer to this dataset as "IJZCG dataset" in the rest of the paper. The traffic is collected over a period of one week by using the SPAN (Switched Port Analyzer) feature on a single 48-port Allied Telesis AT-8000S network switch. One of the switch ports is configured as a monitoring port, while the others were access ports of data VLANs. The Internet connection was used by 36 PCs.

On the host connected to the monitoring port, a Python script was used to capture the network traffic and periodically store it in *pcap* files. From the captured traffic, individual packets were firstly extracted using *scapy* program [17] and then pre-processed for training, taking care to remove information that can lead to biased results. In the first step, packets smaller than 1500B (Maximum Transmission Unit size) were zero-padded to this value. As proposed in [7], the Ethernet header was removed since it does not contain information relevant for traffic identification. The UDP headers were zero-padded to 20B, to match TCP header size. The packets without payload, such as TCP segments with SYN, ANK or FIN flags (generated while establishing or terminating TCP connections) were discarded. Further on, DNS segments are considered irrelevant for this study and thus were omitted from the dataset. Since the dataset was created by generating traffic from a relatively small number of hosts, we masked IP addresses in IP headers to prevent overfitting, i.e., to prevent DL model to learn classifying based on IP addresses. The last pre-processing step implies conversion of the hexadecimal packet representation (provided by the *scapy*) into an integer array (0-255 range) and normalization of the array in the 0-1 range for faster training.

The dataset labeling was performed with nDPI tool [17], which applies DPI technique to detect application protocols. Since nDPI has a high precision rate, it can be safely assumed that the labels generated in this way are the best guess of the ground-truth labels [14]. The flows not detected by nDPI were removed from the dataset. The packets are grouped in N pcap files, which correspond to different applications detected. In total, 27 application protocols were included in the dataset. The number of detected protocols was much larger, but those with less than 1MB of traffic were discarded.

The final dataset included 497296 samples (3.92 GB of data), from which 60% was used for training, 20% for validation and 20% for testing.

The classification uncertainties of the machine learning models have been also quantified on the publicly available ISCX VPN-nonVPN dataset [18]. This dataset includes 16 classes of traffic generated by various Web browsing, email, chat, streaming, file transfer, VOIP and P2P applications.

V. Performance Evaluation

In this section, we present the results of the experimental evaluation. The performance of different models has been compared in terms of accuracy, precision, recall and F1 score [10]. The accuracy is calculated as a ratio of correctly classified samples and the total samples. The precision metric measures the percentage of packets that are properly classified to the target application. The recall corresponds to the percentage of packets in an application class that are correctly identified. F1-score is a widely used metric in classification tasks that combines both recall and precision in a single score, as follows:

$$F_1 = \frac{2\,precision \cdot recall}{precision + recall} \quad (2)$$

In an ideal case, F1-score score should be 1. We consider this metric the most important indicator of the models' overall performance.

Firstly, we evaluate the performance of DL models by varying the number of trainable parameters. The models were implemented using the Keras Python library [20]. To ensure a fair comparison, the models have been designed to have the same number of parameters. In particular, three different configurations for each model were tested - with 3M, 6M and 9M parameters, respectively. The MLP models consist of three hidden layers of the same size. The number of neurons in the hidden layers has been chosen to approximately fit the target number of parameters, i.e., 900 for 3M parameters, 1200 for 6M parameters, and 1800 for 9M parameters. The SAE models were implemented based on the model proposed in [7]. The encoder and decoder parts of SAE have the same NN architecture, with the only difference in hidden layers being reversely ordered. In the SAE configuration with 3M parameters, the encoder consists of three fully connected layers with 600, 500, 400, and 300 neurons. The size of the bottleneck layer has been set to 200. In configurations with a larger number of parameters the bottleneck size is doubled, and the number and dimensions of encoder/decoder layers are increased. The 1D CNN models are based on the architecture proposed in [14]. They consist of two 1D CNN layers, followed by *max-pooling* layer, *flatten* layer and 3 *fully connected* layers with 200, 100 and 50 neurons. The used kernel sizes for the 1D CNN layers are 5 and 4, respectively. The number of filters (convolutional matrices) in configuration with 3M parameters is 5. This parameter was being adjusted to achieve the target NN complexity. We also examined performance of the 1D CNN model without hidden fully connected layers (CNN-NFC).

In each model, fully connected layers and 1D CNN layers (if existing) are regularized with 0.05 dropout rate to avoid overfitting. The models were trained in 10 epochs, with the batch size 32 and the learning rate 0.001. The NN parameters with the best validation result during 10 epochs are used in the testing phase.

The most suitable optimizer was chosen for each DL model based on the experimental evaluation. For 1D CNN and CNN-NFC models, the Adam optimizer was used. SAE and MCP models achieved the best performance with SGD-Momentum optimizer.

The obtained results are shown in Table 1. The best results are achieved with 1D CNN model in configuration with 6M parameters. This model achieves a similar performance with 3M parameters as well. More complex 1D CNN models have not led to further performance improvement. It is interesting to note that CNNNFC performs the worst among the models analysed, although it contains two 1D CNN layers like 1D CNN. The results indicate the importance of fully connected layers at the output for the successful feature extraction. A decent performance is achieved with MLP. However, the results suggest that the other NN designs extract more useful features from the input data.

Table 1: Performance of DL models with different number of training parameters (IJZG dataset)

Algorithm	Precision	Recall	F1 score	Accuracy
CNNNFC 3M	0.8099	0.8116	0.8094	0.8116
CNNNFC 6M	0.8162	0.8145	0.8134	0.8145
CNNNFC 9M	0.8067	0.8096	0.8054	0.8096
1D CNN 3M	0.8661	0.8605	0.8584	0.8605
1D CNN 6M	0.8671	0.8621	0.8608	0.8621
1D CNN 9M	0.8642	0.8570	0.8547	0.8570
MLP 3M	0.8439	0.8365	0.8338	0.8364
MLP 6M	0.8464	0.80	0.8314	0.8370
MLP 9M	0.8430	0.8358	0.8322	0.8358
SAE 3M	0.8428	0.8338	0.8284	0.8338
SAE 6M	0.8563	0.8485	0.8422	0.8485
SAE 9M	0.8450	0.8404	0.8348	0.8404

When compared with the results from [14], our 1D CNN model achieves a better performance on the dataset of higher complexity. Although in [14] only one 1D CNN layer was used, our analysis has shown that an additional CNN layer is not the most significant factor of the performance improvement. While we were using the same dataset labeling procedure as in [14], we have implemented a more comprehensive dataset pre-processing procedure. This probably contributed to the improved CNN generalization ability.

More detailed evaluation results for 1D CNN (6M parameters) are shown in Fig. 4. It can be seen that the model achieves very high precision, recall, and F1 score for application protocols typical for the LAN (Local Area Network), such as UPnP, SSDP, SNMP. The satisfying results are achieved for HTTP traffic as well. On the other side, the performance indicators are relatively low for most of the encrypted traffic (using TLS and QUIC protocols). In general, encryption makes classification difficult for any real-time technique. The results can be potentially improved by considering the entropy of the encrypted packets and flow-level features [14].

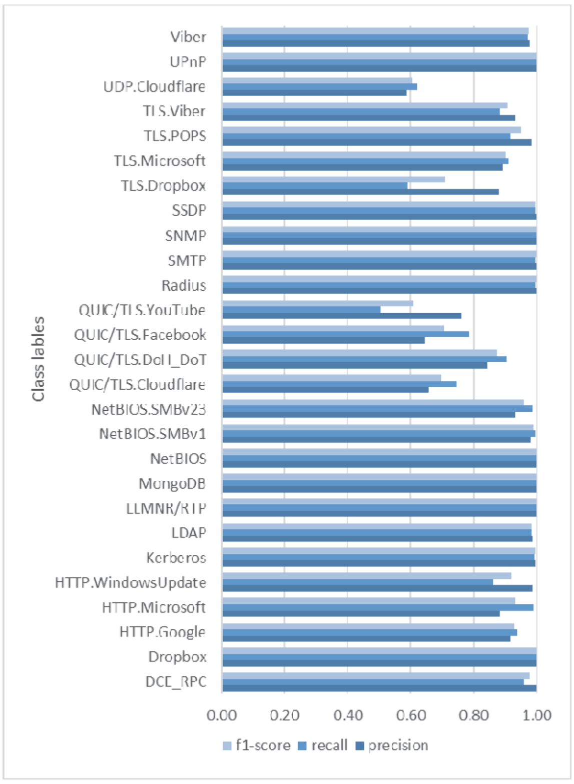

Fig. 4. The 1D CNN performance review for IJZG dataset.

To better understand the effectiveness of DL models, we evaluate their performance on smaller datasets which include 25%, 50% and 75% of the full training dataset. As shown in Table 2, the performance of all DL models deteriorates with the reduction of the training dataset. It is interesting to note that on the small datasets, that account for 25% and 50% of total training data, a simple MLP model outperforms the others. However, on larger datasets MLP achieves less competitive scores. On the same dataset we evaluated the performance of Random Forest and XGBoost algorithms. We used the implementation of Random Forest algorithm available in scikit-learn Python library [21]. There are four main hyperparameters that need to be tuned for the Random Forest algorithm: the number of decision-trees, the maximum depth of the trees, the maximum number of features to be considered at each node and the minimum number of data samples required for a data split to occur. The first two hyperparameters have the biggest impact on the classification speed because they determine the "forest" dimensions. From the experiments we have concluded that increasing number of trees above 50 and increasing maximum tree depth above 30 does not lead to a significant improvement of classification efficiency. Therefore, we choose to use 50 decision-trees with a maximum depth of 30 node layers. The minimum number of samples required for a split is set to 8, while the maximum number of features per node is configured with the default limitation (square root of total number of features). For XGBoost algorithm the maximum tree depth is also set to 30. The obtained results are given in Table 3. The table shows that both algorithms outperform the analyzed DL models. The results also suggest that Random Forest and XGBoost algorithms can be trained with a relatively small amount of data. DL models need more data to achieve the same level of accuracy. The Random Forest is the least computationally expensive among the analyzed solutions, and does not require a GPU for training, which can be considered as an important advantage.

TABLE 2: PERFORMANCE OF DL MODELS AS A FUNCTION OF THE IJZG TRAINING DATASET SIZE

Algoritam	Precision	Recall	F1	Accuracy
1D CNN 100%	0.8671	0.8621	0.8608	0.8621
1D CNN 75%	0.8561	0.8537	0.8486	0.8536
1D CNN 50%	0.8177	0.8147	0.8085	0.8147
1D CNN 25%	0.7775	0.7703	0.7663	0.7703
CNNNFC 100%	0.8162	0.8145	0.8134	0.8145
CNNNFC 75%	0.7940	0.7978	0.7867	0.7978
CNNNFC 50%	0.7956	0.7933	0.7902	0.7933
CNNNFC 25%	0.7872	0.7841	0.7820	0.7841
MLP 100%	0.8464	0.8370	0.8314	0.8370
MLP 75%	0.8456	0.8419	0.8381	0.8419
MLP 50%	0.8456	0.8410	0.8372	0.8410
MLP 25%	0.8392	0.8304	0.8260	0.8304
SAE 100%	0.8563	0.8485	0.8422	0.8485
SAE 75%	0.8427	0.8329	0.8238	0.8329
SAE 50%	0.8155	0.8086	0.8042	0.8086
SAE 25%	0.7798	0.7734	0.7715	0.7734

TABLE 3: PERFORMANCE OF DECISION-TREE BASED ENSEMBLE MODELS AS A FUNCTION OF THE IJZCG DATASET SIZE

Algorithm	Precision	Recall	F1	Accuracy
RF 100%	0.8835	0.8772	0.8757	0.8772
RF 75%	0.8555	0.8514	0.8436	0.8514
RF 50%	0.8503	0.8425	0.8361	0.8425
RF 25%	0.8323	0.8254	0.8233	0.8254
XGBoost 100%	0.9002	0.8941	0.8927	0.8941
XGBoost 75%	0.8737	0.8667	0.8621	0.8667
XGBoost 50%	0.8682	0.8694	0.8650	0.8694
XGBoost 25%	0.8471	0.8469	0.8452	0.8469

In order to gain a better insight into the performance of analysed machine learning models for network traffic identification, we repeated the experiments described above on the ISCX VPN-nonVPN dataset [18]. The performance results as a function of the dataset size are shown in Table 4, for all models. It can be seen that classification efficiency of all models is much higher on the ISCX VPN-nonVPN dataset than on the IJZCG dataset. This can be contributed to the lower dataset complexity, i.e., a lower number of application classes and lower application similarity. The performance gap between decision-tree based ensemble learning algorithms and DL algorithms is now more significant. There is no noticeable difference in the performance of 1D-CNN, SAE and MLP DL models. Among DL models, SAE performed the best on the full dataset, with F1 score of 0.93. On the other side, XGBoost algorithm achieves almost the same F1 score on the smallest dataset, and F1 score of 0.96 on the full dataset. Thus, the results suggest that decision-tree based ensemble methods learn more efficient features for network traffic identification.

TABLE 4: CLASSIFICATION PERFORMANCE ON ISCX VPN-NONVPN DATASET

Algorithm	Precision	Recall	F1	Accuracy
1D CNN 100%	0.9190	0.9144	0.9138	0.9144
1D CNN 75%	0.9161	0.9161	0.9161	0.9161
1D CNN 50%	0.8967	0.8967	0.8967	0.8967
1D CNN 25%	0.8587	0.8587	0.8587	0.8587
CNNNFC 100%	0.8999	0.8964	0.8971	0.8964
CNNNFC 75%	0.8828	0.8847	0.8823	0.8847
CNNNFC 50%	0.8713	0.8693	0.8690	0.8693
CNNNFC 25%	0.8422	0.8353	0.8348	0.8353
MLP 100%	0.9310	0.9307	0.9299	0.9307
MLP 75%	0.9271	0.9277	0.9268	0.9277
MLP 50%	0.9033	0.9017	0.9012	0.9017
MLP 25%	0.8531	0.8535	0.8507	0.8535
SAE 100%	0.9340	0.9345	0.9335	0.9345
SAE 75%	0.9288	0.9285	0.9284	0.9285
SAE 50%	0.9044	0.9049	0.9039	0.9049
SAE 25%	0.8727	0.8725	0.8709	0.8725
RF 100%	0.9586	0.9579	0.9574	0.9579
RF 75%	0.9556	0.9552	0.9543	0.9552
RF 50%	0.9431	0.9433	0.9416	0.9433
RF 25%	0.9257	0.9260	0.9240	0.9260
XGBoost 100%	0.9631	0.9631	0.9625	0.9631
XGBoost 75%	0.9529	0.9518	0.9511	0.9518
XGBoost 50%	0.9454	0.9465	0.9446	0.9465
XGBoost 25%	0.9366	0.9372	0.9356	0.9372

The confusion matrix of XGboost model is shown in Fig. 5. From the figure it can be seen that XGBoost identifies applications with minor confusion. The model mostly struggles to distinguish chat applications. This problem has been already identified in [7].

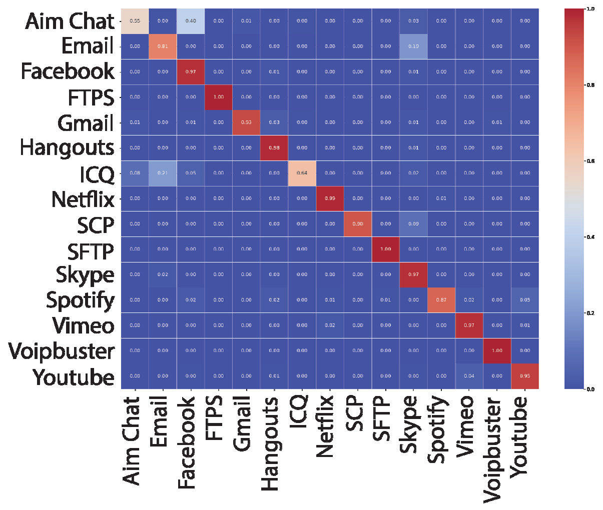

Fig. 5. Confusion matrix of XGBoost model.

VI. CONCLUSION

In this paper, we analyzed the performance of deep learning and decision-tree based ensemble learning models for network traffic identification. A dataset with around half a million packets from 27 applications has been collected and pre-processed in a way that enables faster convergence and prevents biased learning. The experimental results on two different datasets showed that all the models can achieve relatively high classification accuracy and F1 score when dealing with individual packets. Thereby, they automate feature extraction and thus eliminate the need for costly expert interventions, which are inherent in conventional DPI approaches. The study showed that Random Forest and XGBoost models outperform deep learning models. The best performance was achieved with XGBoost model. However, the study suggests that more sophisticated approaches are needed to deal with the classification of encrypted traffic. In our future work, we will try to overcome this issue by considering both flow-level and packet-level features. In that context, it is interesting to explore the capabilities of recurrent neural networks.

REFERENCES

[1] S. Tomovic et al., “An architecture for QoS-aware service deployment in software-defined IoT networks,” 2017 20th *WPMC*, Bali, 2017, pp. 561-567.
[2] S. Tomovic, et al., “SDN Control Framework for QoS Provisioning,” *Proc. of 22nd Telecommunication Forum TELFOR 2014*, pp. 111-114, Belgrade, Serbia, November 2014.
[3] M. Lopez-Martin, et al., “Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things,” in *IEEE Access*, vol. 5, pp. 18042-18050, 2017.
[4] F. Pacheco et al., “Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey,” in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988-2014, 2019.
[5] A.W. Moore, K. Papagiannaki, “Toward the accurate identification of network applications”, In: PAM, Springer, vol 5, pp 41-54, 2015.
[6] S. Rezaei, B. Kroencke, X. Liu, “Large-scale Mobile App Identification Using Deep Learning,” *IEEE Access*, vol. 8, 2020.
[7] M. Lotfollahi, R. Shirali, M. Jafari Siavoshani, and M. Saberian, “Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning,” ArXiv e-prints, Sep. 2017.
[8] N. Cong Luong, et al. “Applications of Deep Reinforcement Learning in Communications and Networking: A Survey,” *Commun. Surveys Tuts*. 21, 4, 2019, pp. 3133–3174.
[9] S. Tomovic and I. Radusinovic, “A Novel Deep Q-learning Method for Dynamic Spectrum Access,” *2020 28th Telecommunications Forum (TELFOR), 2020*, pp. 1-4, doi: 10.1109/TELFOR51502.2020.9306591.
[10] P. Wang, F. Ye, X. Chen and Y. Qian, “Datanet: Deep Learning Based Encrypted Network Traffic Classification in SDN Home Gateway,” in *IEEE Access*, vol. 6, pp. 55380-55391, 2018, doi: 10.1109/ACCESS.2018.2872430.
[11] U. Trivedi and M. Patel, “A fully automated deep packet inspection verification system with machine learning,” *IEEE Int. Conf. on Advanced Networks and Telecomm. Systems*, 2016, pp. 1-6.
[12] N. Nikolić, S. Tomović and I. Radusinović, “Performance Analysis of Deep Learning Models for Network Traffic Identification,” *2021 29th Telecommunications Forum (TELFOR)*, 2021, pp. 1-4.
[13] S. Kiranyaz, et al., “1D convolutional neural networks and applications: A survey,” *Mechanical Systems and Signal Processing*, Volume 151, 2021.
[14] A. V. Jain, “Network Traffic Identification with Convolutional Neural Networks,” *IEEE Intl Conf on Big Data Intel. and Comp. and Cyber Science and Technology*, 2018, pp. 1001-1007.
[15] I. P. Possebon, et al., “Improved Network Traffic Classification Using Ensemble Learning,” *2019 IEEE Symposium on Computers and Communications (ISCC)*, 2019, pp. 1-6.
[16] Chen, Tianqi, and Carlos Guestrin, “Xgboost: A scalable tree boosting system,” *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.
[17] Scapy program: https://scapy.readthedocs.io/en/latest/usage.html.
[18] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, “ndpi: Open-source high-speed deep packet inspection,” *Int. Wireless Comm. and Mobile Computing Conference*, pp. 617–622, Aug 2014.
[19] Chollet F, et al (2017) Keras. https://github.com/fchollet/keras.
[20] G. D. Gil, et. al., “Characterization of Encrypted and VPN Traffic Using Time-Related Features,” *International Conference on Information Systems Security and Privacy (ICISSP 2016)*, pp 407-414, Rome, Italy.
[21] Scikit-learn Random Forest implementation: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html