A New Transfer Learning-Based Traffic Classification Algorithm for a Multi-Domain SDN Network

Conference Paper · December 2023

DOI: 10.1145/3628797.3628804

7 authors, including:

Son Dương
Dong Thap University
12 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Van Tong
Hanoi University of Science and Technology
36 PUBLICATIONS **530** CITATIONS
SEE PROFILE

Hai-Anh Tran
Hanoi University of Science and Technology
71 PUBLICATIONS **787** CITATIONS
SEE PROFILE

A New Transfer Learning-Based Traffic Classification Algorithm for a Multi-Domain SDN Network

Nam-Thang Hoang
thanghn@huce.edu.vn
School of Information and Communication Technology (SOICT), Hanoi University of Science and Technology (HUST)
Hanoi, Vietnam

Cong-Son Duong
son167464@huce.edu.vn
Faculty of Information Technology, Hanoi University of Civil Engineering (HUCE)
Hanoi, Vietnam

Minh-Ngoc Vu
minh0197266@huce.edu.vn
Faculty of Information Technology, Hanoi University of Civil Engineering (HUCE)
Hanoi, Vietnam

Huy-Hieu Nguyen
hieu1520065@huce.edu.vn
Faculty of Information Technology, Hanoi University of Civil Engineering (HUCE)
Hanoi, Vietnam

Truong X. Tran
truong.tran@psu.edu
School of Science, Engineering and Technology, Penn State University
Penn State Harrisburg, USA

Van Tong
vantv@soict.hust.edu.vn
School of Information and Communication Technology (SOICT), Hanoi University of Science and Technology (HUST)
Hanoi, Vietnam

Hai-Anh Tran*
anhth@soict.hust.edu.vn
School of Information and Communication Technology (SOICT), Hanoi University of Science and Technology (HUST)
Hanoi, Vietnam

ABSTRACT

To enhance the efficiency and resource utilization of a computer network, it is imperative to classify network traffic and implement distinct priority policies. Network traffic classification plays a pivotal role across various domains, including network administration, cybersecurity, and network resource optimization. As encrypted network data undergoes diverse evolution, as evident in datasets from tech giants like Google, Facebook, and Youtube, traditional traffic classification methods have given way to machine learning-based approaches. Given that computer networks are primarily deployed as distributed multi-domain systems, employing machine learning for traffic classification becomes challenging when a new network domain appears with a limited dataset. One potential remedy is to employ transfer learning, allowing knowledge transfer from a pre-trained model in an established domain to a new one. In this paper, we introduce an algorithm named Multi-class TrAdaBoost-CNN for a distributed and multi-domains SDN network. This algorithm combines a variant of the Multi-class TrAdaBoost approach with a Convolutional Neural Network (CNN) as a learner model. Our experimental results demonstrate that our proposed algorithm surpasses the performance of the traditional CNN model, achieving accuracy improvements of up to 16% even with extremely limited data.

CCS CONCEPTS

• **Networks → Data path algorithms**.

KEYWORDS

Encrypted Network Traffic Classification, Transfer Learning, Imbalanced Data, TrAdaBoost, Distributed Software-defined Network

ACM Reference Format:
Nam-Thang Hoang, Cong-Son Duong, Minh-Ngoc Vu, Huy-Hieu Nguyen, Truong X. Tran, Van Tong, and Hai-Anh Tran. 2023. A New Transfer Learning-Based Traffic Classification Algorithm for a Multi-Domain SDN Network. In *The 12th International Symposium on Information and Communication Technology (SoICT 2023), December 7–8, 2023, Hochiminh, Vietnam.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3568562.3568632

*Corresponding author

SoICT 2023, December 7–8, 2023, Hochiminh, Vietnam

ACM ISBN 979-8-4007-0891-6…$15.00
https://doi.org/10.1145/3568562.3568632

1 INTRODUCTION

The problem of network traffic classification [14, 17] has been extensively studied for nearly 20 years due to its importance in various aspects, such as improved performance monitoring, bandwidth optimization, QoS (Quality of Service) improvement, capacity planning

& scaling, troubleshooting & diagnostics, policy enforcement, resource optimization, billing & accounting, and forecasting & trend analysis [3].

Over the past decade, traffic classification has grappled with two primary challenges. Firstly, the widespread adoption of encrypted traffic, driven by the surge in mobile devices, has been instrumental in safeguarding user privacy and anonymity online [12, 23]. However, this surge in encryption has also led to a complex landscape of concealed applications and protocols, making accurate detection and classification a formidable task [22]. Secondly, the Internet is currently experiencing a shift towards autonomous network systems characterized by dispersed resources and centralized control. Modern network systems are segregated into independent and diverse domains, all centrally managed by a controller in accordance with the Software-defined network (SDN) model. This decentralized and heterogeneous network environment poses a considerable challenge in classifying network traffic.

These two challenges have rendered conventional traffic classification methods, such as port-based and payload-based techniques, ineffective [14]. Consequently, there has been a notable shift towards employing Machine Learning (ML)-based service classification methods. The significant progress in ML algorithms, particularly those utilizing semi-supervised learning, has propelled the popularity of these approaches. Previous research in this domain has primarily focused on leveraging both traditional ML and deep learning algorithms, yielding promising outcomes [1, 2, 4, 5, 11, 15, 18, 20, 24]. Nonetheless, these methods have not taken into account the dynamic nature of network systems, where new domains continually emerge with limited data sets. This scarcity of data poses a challenge in training machine learning models for these nascent domains. To address this, this paper employs a transfer learning mechanism to transfer knowledge from pre-trained models of established domains to the newly emerging ones.

The main contributions of this paper are as follows:

- A new **Multi-class TrAdaBoost-CNN algorithm** is proposed to address the issue of cross-domain classification of encrypted network services. Our method extended the Multi-Class TrAdaBoost algorithm but a convolution neural network (CNN) is used as a weak learner.
- The extensive **experiments on two different domains with imbalanced data distributions** are conducted to evaluate the effectiveness of our method.

Outline: The remainder of this paper is organized as follows. Section 2 provides an overview of related work and background in the field of encrypted traffic classification TrAdaBoost algorithm and KDN. Section 3 presents the methodology and details of our proposed approach. In Section 4, we describe the experimental setup and present the results and analysis. Finally, Section 5 concludes the paper and discusses future research directions.

2 RELATED WORK AND BACKGROUND

2.1 Encrypted traffic classification

The task of network traffic classification in the past used port-based methods and payload-based methods for classification[14]. Port-based methods use the header of the data packet containing TCP or UDP port number which uniquely identifies the application. This approach is not suitable anymore since most applications now use dynamic ports. Payload-based methods examine the payload part of the packet and match it with a set of stored patterns. However, with the growth of encrypted data, this method becomes impossible to classify. With the rise of Machine Learning algorithms, most recent studies have mainly focused on applying semi-supervised algorithms to the task of encrypted traffic classification and have achieved promising results.

Sudad Abdulrazzaq et al. [1] conducted experiments with basic machine learning algorithms such as K-NN, SVM, DT, RF, DNN, and CNN on an SDN topology. Additionally, the paper also proposed the Synthetic Minority Over Sampling Technique (SMOTE) to address the issue of imbalanced data. SMOTE is a technique that increases the minority labels by combining samples of nearby data points in the feature space rather than replacing them. Final experiments show that all approaches achieved good results on both encrypted and unencrypted data.

Lopez-Martin et al. [13] introduced a novel approach to traffic classification using CNN with a single-layer LSTM model. Their goal was to detect and classify 108 different services. In their research, each flow consisted of 20 packets, and each packet contained 6 features such as source port, destination port, packet payload size, TCP window size, inter-arrival time, and packet direction. Interestingly, the authors relied on the destination port, which was often the same for different services, to differentiate between the various services in the network.

Nowadays, approaches utilizing transfer learning techniques are becoming increasingly promising in the problem of encrypted network traffic classification. In the paper [4], data was transferred from a pre-trained CNN model to another model used for classifying Google applications. The initial model was trained on one dataset and then transferred and fine-tuned on a smaller dataset. However, this model was limited in accuracy, reaching only **85%** due to the dissimilarity between the two datasets. This dissimilarity can arise mainly from differences in label distribution, data deficiencies, and other factors.

Sun et al. [21] introduced a solution to the issue of different data distributions assumed by the traditional ML method using the TrAdaBoost algorithm. This method utilizes a Maxent model as a base classifier and updates sample weights based on the error rate. The method achieved high classification accuracy on a new dataset that is not completely similar to the training dataset, with an accuracy of up to **98.7%**.

Fengjun Shang et al. [19] use an improvement of the TrAdaBoost algorithm called the Dynamic-TrAdaBoost algorithm for the task of non-encrypted network classification. They address a drawback of the TrAdaBoost algorithm, where the source versions converge before the transfer learning. They resolve it using an additional dynamic cost function. The paper experiments with both the original TrAdaBoost algorithm and Dynamic TrAdaBoost algorithm on the same dataset. Results show that Dynamic-TrAdaBoost has achieved **99.3%** accuracy compared to **97.5%** for original TrAdaBoost.

Hanxian He et al. [9] proposed the VoxNet framework for the classification of mobile lidar data. The framework used the Multi-class TrAdaBoost algorithm, an extension of the TrAdaBoost algorithm for multi-label classification to address the lack of sufficient training samples for different object categories. They experiment with the

algorithm by using two lidar datasets with different distributions. As a result, Multi-class TrAdaBoost has achieved higher accuracies in most of the categories in terms of precision, recall, and F1-score compared to other models, specifically with an unbalanced dataset.

2.2 TrAdaBoost Algorithm

TrAdaBoost [6] is an algorithm that is built upon the AdaBoost ensemble learning algorithm [7] for transfer learning. AdaBoost like many other Machine Learning algorithms, usually assumes that the distributions of training data and test data are identical. However, this assumption is not true in most real-life scenarios. In addition, the process of transferring a Machine Learning model becomes ineffective. To resolve this problem, TrAdaBoost algorithm is proposed with a sample weight update strategy to increase the accuracy of the transfer process.

AdaBoost, or Adaptive Boosting, is a boosting algorithm that aims to improve the performance of classifiers by combining many weak classifiers into strong classifier. In each iteration, Adaboost increases the accuracy of selecting the next weak learner, such as a Decision Tree, by boosting the importance of misclassified labels. TrAdaboost assumes that the source and target domain data have small correlation differences and imbalanced distributions. Therefore, the algorithm allows the use of a small amount of target domain data D_T, combined with the source domain data, D_S, to enhance the classifier $h_f(x)$.

The key point of TrAdaBoost algorithm is that it uses a sample weight update strategy for the issue of imbalanced distributions dataset. After every boosting iteration, this method adjusts the weights of the source dataset by sequentially increasing the weights of misclassified source samples based on a constant factor. The current weights of correctly classified source samples are maintained. This strategy makes the learner in the next iteration focus more on the wrong classified data.

2.3 Knowledge-defined Network

SDN [16], a programmable network enables network advancements, where the forwarding devices are decoupled from the control plane, known as the control layer. This architecture makes the behavior of the network core more flexible in meeting the Quality of Service requirements posed by different access technologies, applications, and devices. The Distributed Knowledge-Defined Networking model, which combines SDN, Network Analytics (NA), and ML is applied to create an intelligent network system that integrates machine learning models more effectively in the classification process. KDN architecture is deployed with three planes: the data plane, the control plane, and the knowledge plane for implementation of large-scale multi-domain SDN.

The data layer includes switches, which are programmable network devices responsible for forwarding or dropping packets. Controllers send forwarding rules to switches to enable these actions. The control layer acts as a bridge between applications and network devices, allowing them to communicate. To enhance the scalability of SDN, multiple controller entities are deployed and can communicate with each other through an East-West interface, which was introduced in a recent study [10].

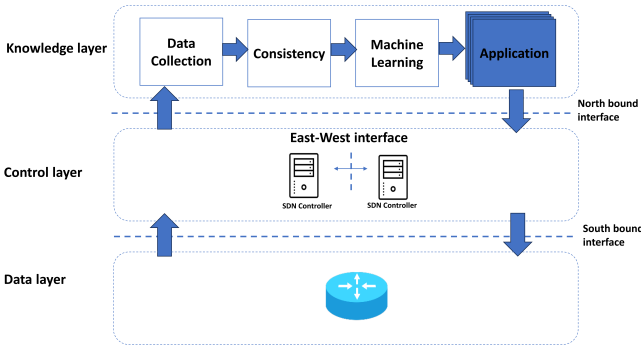

Figure 1: KDN architecture

The knowledge plane consists of four primary modules: data collection, consistency, machine learning, and application. Its aim is to enable applications to improve their performance and become intelligent systems. The data collection module observes and collects packet data from the data plane, which is then sent to the consistency module. This module ensures that data is transmitted to other SDN domains, maintaining consistency across multiple controllers. The machine learning module plays a crucial role in the knowledge plane by gathering and deducing information about network behavior. This knowledge enhances the decision-making capabilities of SDN controllers. Lastly, the application module allows ISPs to offer services to end-users.

3 METHODOLOGY

3.1 Problem Statement

Giving two separate SDN domains, source domain S and target domain T, with corresponding data D_S and D_T, representing network traffic and their labels. The task is to create a classification model that can categorize data from the target domain into K different network service groups based on the patterns learned from the source domain. The problem is modeled as follows:

Having $D_S = \{(x_{s1}, y_{s1}), (x_{s2}, y_{s2}), \ldots, (x_{sm_1}, y_{sm_1})\}$ and $D_T = \{(x_{t1}, y_{t1}), (x_{t2}, y_{t2}), \ldots, (x_{tm_2}, y_{tm_2})\}$ representing the datasets from the source and target domains, respectively, where:

- $x_{ti} \in \mathbb{R}^{n_p \cdot v}$ and $x_{si} \in \mathbb{R}^{n_p \cdot v}$ are the payload encoding with v bytes for each service group in the source and target domains.
- y_{ti} and $y_{si} \in \{0, 1, \ldots, K\}$ are the corresponding labels indicating the network service group.
- m_1 and m_2 are the number of samples in the source and target domains, respectively.
- n_p are the number of packets of each flow.
- K is the number of network service groups.

Objective: The objective is to build a classification model using the combined dataset from both domains, D_S and D_T, with the ultimate goal of minimizing the error rate ε when classifying the target data.

Subject to: The target domain dataset D_T is plagued by a scarcity of data.

To address this problem, we propose an algorithm called Multi-Class TrAdaBoost-CNN, which is an improved version of the original Multi-Class TrAdaBoost algorithm.

3.2 Proposed Algorithm

In this section, we introduce an extended version of the Multi-Class TrAdaBoost-CNN algorithm, designed for network traffic classification in KDN based on CNN.

Algorithm 1 Multi-Class TrAdaBoost-CNN

Input: Two labeled datasets T_d and T_s, the unlabeled dataset S, a CNN model **CNNModel**, and the maximum number of iterations N.

Output: The multi-label hypothesis:

$$h_f(x) = \underset{k}{\operatorname{argmax}} \left(\sum_{t=1}^{N} \alpha_t \cdot I(h_t(x) = k) \right)$$

1: At $t = 1$, set $\beta^1 = (\beta_1^1, ..., \beta_{n+m}^1)$, $\alpha = \ln\left(\frac{1}{1+\sqrt{2ln n/N}}\right)$

2: **for** $t = 1, ..., N$ **do**

3: Set $p_t = \frac{\beta_t}{\sum_{i=1}^{n+m} \beta_{ti}}$.

4: Train the **CNNModel** on the combined training set T with the distribution p_t and the unlabeled dataset S.

5: Calculate the error of **CNNModel** on T_s:

$$\varepsilon_t = \sum_{i=n+1}^{n+m} \frac{\beta_i^t \cdot (\mathrm{I}(h_t(x_i) \neq c(x_i)))}{\sum_{i=n+1}^{n+m} \beta_i^t}$$

6: Set $\alpha_t = \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) + \ln(K-1)$, $C^t = K \cdot (1 - \varepsilon_t)$ where $\varepsilon_t < \frac{K-1}{K}$

7: Update the new weight:

$$\beta_i^{t+1} = \begin{cases} \beta_i^t \cdot e^{\alpha_t \cdot \mathrm{I}(h_t(x_i) \neq c(x_i))}, & \text{for } 1 \leq i \leq n \\ C^t \cdot \beta_i^t \cdot e^{\alpha \cdot \mathrm{I}(h_t(x_i) \neq c(x_i))}, & \text{for } n+1 \leq i \leq n+m \end{cases}$$

8: **end for**

In our proposed algorithm, denoted as Algorithm 1, each input flow can be conceptualized as a **2D matrix**, each input packet can be visualized as a vector with varying lengths. We achieve 2d matrix representation by **concatenating** the first n_p packets of each flow together, trimming packets exceeding v bytes and padding 0, forming the foundation for our 2D matrix format(note that, v, n_p represents the dimension of matrix). Instead of using a **Decision Tree** as the weak classifier, we opt for a Convolutional Neural Network (CNN). This choice is motivated by the suitability of **CNNs** for processing input data represented as 2D matrices. This characteristic seamlessly aligns with our data format, allowing us to leverage the power of **CNNs** in efficiently analyzing the input flows. This aspect is particularly crucial because **Multi Class-Adaboost** requires an error rate below **(K-1)/K** to prevent weight updating parameter α_t negative [8].

Fig. 2 illustrates the flowchart of the algorithm. When the algorithm does not terminate prematurely, the step by step of the algorithm is as follow:

(1) Pre-process the target and source data.
(2) (Line 1) Initialize weights β_1 at $t = 1$, Calculate α.
(3) (Line 4) Train the CNN model using the combined source and target data.
(4) (Line 5 - 6) Calculate the error rate, α_t, and C_t for the CNN model on the target data.
(5) (Lines 7) If not all N iterations have been completed, use the CNN model from the previous iteration as the initialization for the next iteration and update the aggregated weights for the final model.
(6) End the iterations and generate predictions for the target.

Table 1: Notations and Descriptions of Algorithms

Notation	Description
T_d	Labeled data set for the source domain
T_s	Labeled data set for the target domain
S	Unlabeled data set
CNNModel	Convolution Neural Network Learner
N	Maximum number of iterations
β	Weight vector
n	Number of samples in the target domain
m	Number of samples in the source domain
p_t	Probability distribution at iteration t
ε_t	Error on T_s at iteration t
K	Number of labels
$h_t(x_i)$	Prediction for sample x_i at iteration t
$c(x_i)$	True class label for sample x_i
$h_f(x)$	Multi-label hypothesis output
$\mathrm{I}(a \neq b)$	return 1 if $a \neq b$, else return 0
C_t	Correction factor
α	Weight updating parameter based on max iterations
α_t	Weight updating parameter based on multiclass loss

4 EXPERIMENTAL RESULTS

This section presents the evaluation results of the proposed method.

Our evaluation metrics are the Accuracy and F1-score on different testing conditions. The testing data sets are obtained by using the holdout method.

4.1 Data Prepossessing and CNN Model Configuration

The ETC datasets (D_1 and D_2) consist of three different services (e.g., E-commerce, Video-on-Demand, and Interactive Data). This dataset was captured recently between January and March 2023 in Southeast Asia. In this paper, we consider a variety of services encompassing Thegiodidong, Shopee, Tiki, eBay, Facebook video, YouTube, Chat, and Amazon. The description of datasets is depicted in Tab 2.

The data prepossessing of these datasets (captured at the KDN's data layer) include four main steps: Header removal, conversion to bytes, Normalization, Padding with zero, and Packet aggregation. Following the approach of Tong et al[22] for classifying QUIC services, beyond the payload, other components are entirely irrelevant. Therefore, we remove headers and irrelevant information and convert them to bytes to reduce the input information for the model. The data is then normalized by dividing by

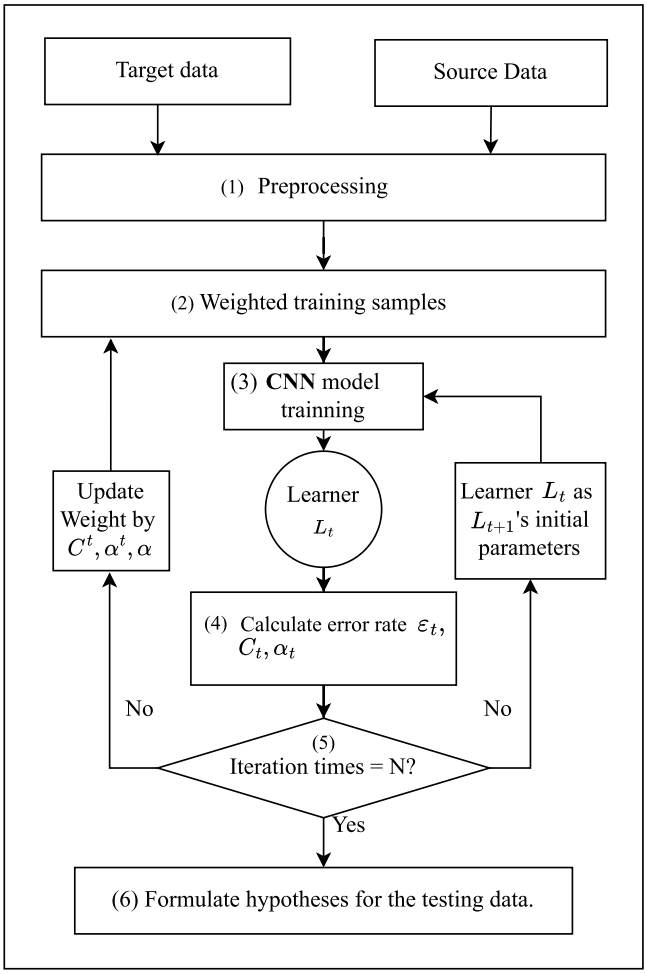

Figure 2: Multi-Class TrAdaBoost-CNN Algorithm Structure.

Table 2: Dataset specification.

Data	Services	Nums Flows	Type
T_S	Amazon	834	E-Commerce
	eBay	1836	E-Commerce
	Facebook video	1379	Video-on-Demand
	Voice Call	2198	Interactive Data
T_t	Thegioididong	509	E-Commerce
	Tiki	1205	E-Commerce
	Youtube	4645	Video-on-Demand
	Chat	2556	Interactive Data

255, the largest value of one byte. We consider only the first 1400 features of each packet and pad zeros to packets shorter than 1400 bytes. Finally, we aggregate 20 consecutive packets with the same flow-id to form a 1400x20 matrix as input to the model.

The CNN model consists of eight convolutional layers, each designed to extract features from the input data. The configurations of these layers are as follows:

Table 3: Details of Network Layers

Layer	Number of Filters	Filter Size	Padding
Layer 1	128	5x5	Same
Layer 2	64	5x5	Same
Layer 3	64	3x3	Same
Layer 4	32	3x3	Same
Layer 5	32	3x3	Same
Layer 6	16	3x3	Same
Layer 7	16	3x3	Same
Layer 8	8	3x3	Same

After every two convolutional layers, max-pooling is applied with a 2x2 pool size.

The final part of the CNN consists of two fully connected layers for classification. The first fully connected layer contains 256 neurons with ReLU activation. A dropout layer with a rate of 0.1 is applied to prevent overfitting. The last fully connected layer has three output neurons matching the number of service groups in our testing SDN domain.

Table 4: Additional Network Layers

Layer	Number of Neurons	AF
Fully Connected Layer	256	ReLU
Dropout	0.1	-
Fully Connected Layer	3	Softmax

4.2 Evaluation Results and Analysis

This section presents the findings of the conducted experiments, which are assessed from three distinct perspectives. **(1) Optimal Selection of Input Parameters:** An examination is made of the trade-off between model complexity and accuracy as the size of the input matrix is systematically varied. **(2) Selection of the Optimal CNN Model:** The fine-tuning scenarios of the CNN model are implemented on our dataset. **(3) Comparative Assessment of Algorithms:** An evaluation is performed to gauge the quality of classifiers when training data is significantly limited. For this latter assessment, the testing dataset is partitioned into a training subset comprising 1% of the data and a testing subset containing the remaining 99%.

1) Optimal Selection of Input Parameters: Before conducting algorithm comparisons, it is imperative to determine the optimal parameter settings for the input data. Table 5 presents an evaluation of the CNN model's performance as the number of packets and the number of bytes per packet are incrementally increased. The metric employed for assessing quality is accuracy.

In Table 5, the most proficient model corresponds to a dataset configuration with N_b (Number of Bytes) set at 1400 and N_p (Number of Packets) at 20, yielding an accuracy of 81%. However, it is noteworthy that this configuration exhibits a relatively high Average Flow Classification Time (AFCT) of 0.119 seconds. Subsequently, as observed in Table 3, variations in N_p with values of 5, 10, and 20 packets are examined. The accuracy demonstrates a significant

Table 5: Evaluate the CNN model with different numbers of packets and bytes

Number of bytes	5 Packets		10 Packets		20 Packets	
	Accuracy	AFCT	Accuracy	AFCT	Accuracy	AFCT
128 Bytes	0.57	0.0406	0.64	0.0518	0.67	0.0573
256 Bytes	0.57	0.0505	0.69	0.0596	0.76	0.0649
512 Bytes	0.59	0.0666	0.71	0.0817	0.77	0.0893
1400 Bytes	0.66	0.0942	0.73	0.1070	0.78	0.1190

Table 6: Performance Metrics for Different Network Layers and Epochs

Layers	100 epochs		200 epochs		300 epochs		400 epochs		500 epochs	
	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
4 Layers	0.72	0.68	0.72	0.68	0.71	0.66	0.72	0.68	0.69	0.65
6 Layers	0.73	0.69	0.76	0.75	0.70	0.65	0.75	0.72	0.72	0.70
8 Layers	0.63	0.54	0.72	0.70	0.69	0.64	0.76	0.75	0.75	0.75

Table 7: Overal Results and F1-Score on each class

Algorithm	F1-Score	Precision	Accuracy	Recall	E-commerce	Video-on-Demand	Interactive Data
MC TrAdaBoost-CNN	0.92	0.93	0.92	0.92	1.00	0.92	0.86
CNN ($D_S + D_T$)	0.78	0.82	0.78	0.78	0.96	0.76	0.71
CNN (D_T)	0.76	0.79	0.76	0.76	0.95	0.76	0.63

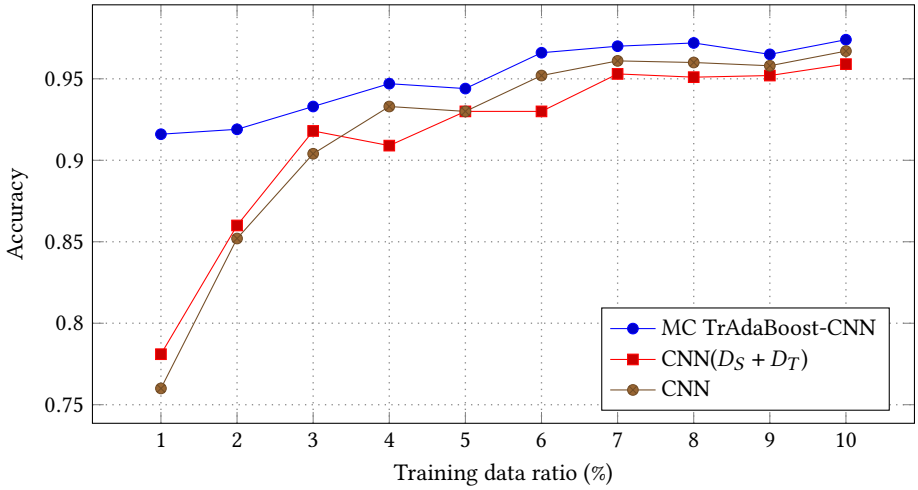

Figure 3: Effect of Data Training Ratio on Accuracy

increase across these experiments, while the AFCT remains reasonably manageable. Consequently, the choice for the input size parameter is established as $N_p = 20$. Notably, for $N_p = 20$, it is observed that the accuracy levels of 256, 512, and 1400 do not differ significantly; however, their processing times are considerably higher. Therefore, a judicious trade-off is made in favor of input parameters with $N_b = 256$ and $N_p = 20$ due to their reasonably good performance and accuracy.

2) Selection of the Optimal CNN Model: Through experiments involving variations in the number of layers, as delineated in Table 6, the two most competitive models with statistically indistinguishable results are identified. One model features 8 layers and 400 epochs, while the other comprises 6 layers and 200 epochs, both achieving an accuracy of 0.76 and an F1-Score of 0.75. However, the 6-layer model with 200 epochs exhibits faster convergence. Consequently, in subsequent experiments, the 6-layer model is employed for the MCTrAdaboost-CNN algorithm.

3) Comparative Assessment of Algorithms: To verify the effectiveness of the proposed algorithm, a comprehensive series of experiments was meticulously conducted, comparing its performance to several other algorithms.

Benchmarks:

- $CNN(D_T)$ is model, which is trained on data from the target domain (D_T).
- $CNN_{D_S+D_T}$ represents a CNN model trained using data from both the source domain (D_S) and the target domain (D_T).

The results of these experiments are presented in Table 7. Notably, when employing the MC TrAdaboost CNN algorithm, the F1-Score exhibits a 16% improvement compared to utilizing data solely from the target domain, and a 14% enhancement over data synthesis from both domains.

Furthermore, the scope of experimentation was extended by varying the ratio of training and testing data to underscore the efficacy of the proposed algorithm, particularly under different data scarcity conditions in SDN classification.

Table 8 illustrates the algorithm's accuracy under varying training data ratios. It is evident that in scenarios of extreme data scarcity, algorithms incorporating additional data from D_S ($MCTrAdaBoost-CNN$ and $CNN(D_S + D_T)$) tend to outperform the traditional $CNN(D_T)$. However, with an ample data supply, the $CNN(D_S + D_T)$ algorithm displays inferior accuracy when compared to the traditional CNN algorithm. In contrast, the $MC-TrAdaBoostCNN$ algorithm consistently demonstrates commendable accuracy across diverse data scenarios. Notably, as data availability decreases, the $MCTrAdaboost-CNN$ algorithm exhibits superior classification accuracy compared to other algorithms.

5 CONCLUSION

In this paper, we have introduced a novel methodology called MC-TrAdaboost CNN for cross-domain classification of encrypted network services in Knowledge-defined networking, especially when dealing with limited data in emerging domains. Our proposed algorithm consistently outperforms the traditional CNN model, achieving accuracy improvements of up to 16% in situations with extremely scarce data.

Despite the superior accuracy achieved, our proposed algorithm has shown some significant drawbacks. For example, if the Data between the source and target domains is too different, the transfer learning process may lead to lower results than without transfer learning. In the future, we may develop an algorithm to select domains before the transfer process to mitigate this issue.

6 ACKNOWLEDGEMENT

Cong-Son Duong was funded by the Master, PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), code VINIF.2023.ThS.113.

REFERENCES

[1] Sudad Abdulazzaq and Mehmet Demirci. A deep learning based system for traffic engineering in software defined networks. *International Journal of Intelligent Systems and Applications in Engineering*, 8(4):206–213, Dec. 2020.

[2] Daniel J. Arndt and A. Nur Zincir-Heywood. A comparison of three machine learning techniques for encrypted network traffic analysis. In *2011 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 107–114, 2011.

[3] Ahmad Azab, Mahmoud Khasawneh, Saed Alrabaee, Kim-Kwang Raymond Choo, and Maysa Sarsour. Network traffic classification: Techniques, datasets, and challenges. *Digital Communications and Networks*, 2022.

[4] Roni Bar-Yanai, Michael Langberg, David Peleg, and Liam Roditty. Realtime classification for encrypted traffic. In *The Sea*, 2010.

[5] Laurent Bernaille and Renata Teixeira. Early recognition of encrypted applications. In *Proceedings of the 8th International Conference on Passive and Active Network Measurement*, PAM'07, page 165–175, Berlin, Heidelberg, 2007. Springer-Verlag.

[6] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 193–200, New York, NY, USA, 2007. Association for Computing Machinery.

[7] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[8] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[9] Hanxian He, Kourosh Khoshelham, and Clive S. Fraser. A multiclass tradaboost transfer learning algorithm for the classification of mobile lidar data. *Isprs Journal of Photogrammetry and Remote Sensing*, 166:118–127, 2020.

[10] Nam-Thang Hoang, Hai-Nam Nguyen, Hai-Anh Tran, and Sami Souihi. A novel adaptive east–west interface for a heterogeneous and distributed sdn network. *Electronics*, 11(7), 2022.

[11] Yuichi Kumano, Shingo Ata, Nobuyuki Nakamura, Yoshihiro Nakahira, and Ikuo Oka. Towards real-time processing for application identification of encrypted traffic. In *2014 International Conference on Computing, Networking and Communications (ICNC)*, pages 136–140, 2014.

[12] Xinjie Lin, Gang Xiong, Gaopeng Gou, Zhen Li, Junzheng Shi, and Jing Yu. ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In *Proceedings of the ACM Web Conference 2022*. ACM, apr 2022.

[13] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, 5:18042–18050, 2017.

[14] Thuy T.T. Nguyen and Grenville Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials*, 10(4):56–76, 2008.

[15] Yohei Okada, Shingo Ata, Nobuyuki Nakamura, Yoshihiro Nakahira, and Ikuo Oka. Application identification from encrypted traffic based on characteristic changes by encryption. In *2011 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pages 1–6, 2011.

[16] Arpita Prajapati, Achyut Sakadasariya, and Jitisha Patel. Software defined network: Future of networking. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, pages 1351–1354, 2018.

[17] Shahbaz Rezaei and Xin Liu. Deep learning for encrypted traffic classification: An overview. *IEEE Communications Magazine*, 57(5):76–81, 2019.

[18] Shahbaz Rezaei and Xin Liu. How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets, 2020.

[19] Fengjun Shang, Saisai Li, and Jinlong He. Improved application of transfer learning in network traffic classification. *Journal of Physics: Conference Series*, 1682(1):012011, nov 2020.

[20] Guang-Lu Sun, Yibo Xue, Yingfei Dong, Dongsheng Wang, and Chenglong Li. An novel hybrid method for effectively classifying encrypted traffic. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5, 2010.

[21] Guanglu Sun, Lili Liang, Teng Chen, Feng Xiao, and Fei Lang. Network traffic classification based on transfer learning. *Computers & Electrical Engineering*, 69:920–927, 2018.

[22] Van Tong, Hai Anh Tran, Sami Souihi, and Abdelhamid Mellouk. A novel quic traffic classifier based on convolutional neural networks. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[23] Pan Wang, Xuejiao Chen, Feng Ye, and Zhixin Sun. A survey of techniques for mobile service encrypted traffic classification using deep learning. *IEEE Access*, 7:54024–54033, 2019.

[24] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 43–48, 2017.