IEEE ComSoc | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 14 June 2023; revised 8 October 2023; accepted 24 January 2024.
Date of publication 31 January 2024; date of current version 9 February 2024.

The associate editor coordinating the review of this article and approving it for publication was E. Bou-Harb.

Digital Object Identifier 10.1109/TMLCN.2024.3360915

Unleashing the Potential of Knowledge Distillation for IoT Traffic Classification

MAHMOUD ABBASI[1] (Member, IEEE), AMIN SHAHRAKI[2] (Senior Member, IEEE), JAVIER PRIETO[3] (Senior Member, IEEE), ANGÉLICA GONZÁLEZ ARRIETA[3], AND JUAN M. CORCHADO[3]

[1]BISITE Research Group, University of Salamanca, 37007 Salamanca, Spain
[2]Department of Informatics, University of Oslo, 0373 Oslo, Norway
[3]Department of Computer Science and Automation Control, University of Salamanca, 37007 Salamanca, Spain

CORRESPONDING AUTHOR: A. SHAHRAKI (am.shahraki@ieee.org)

This work was supported by the IoTalentum project within the framework of Marie Sklodowska-Curie Actions Innovative Training Networks-European Training Networks (ITN-ETN) (grant number 953442) which is funded by the European Union Horizon 2020 research and innovation program.

ABSTRACT The Internet of Things (IoT) has revolutionized our lives by generating large amounts of data, however, the data needs to be collected, processed, and analyzed in real-time. Network Traffic Classification (NTC) in IoT is a crucial step for optimizing network performance, enhancing security, and improving user experience. Different methods are introduced for NTC, but recently Machine Learning solutions have received high attention in this field, however, Traditional Machine Learning (ML) methods struggle with the complexity and heterogeneity of IoT traffic, as well as the limited resources of IoT devices. Deep learning shows promise but is computationally intensive for resource-constrained IoT devices. Knowledge distillation is a solution to help ML by compressing complex models into smaller ones suitable for IoT devices. In this paper, we examine the use of knowledge distillation for IoT traffic classification. Through experiments, we show that the student model achieves a balance between accuracy and efficiency. It exhibits similar accuracy to the larger teacher model while maintaining a smaller size. This makes it a suitable alternative for resource-constrained scenarios like mobile or IoT traffic classification. We find that the knowledge distillation technique effectively transfers knowledge from the teacher model to the student model, even with reduced training data. The results also demonstrate the robustness of the approach, as the student model performs well even with the removal of certain classes. Additionally, we highlight the trade-off between model capacity and computational cost, suggesting that increasing model size beyond a certain point may not be beneficial. The findings emphasize the value of soft labels in training student models with limited data resources.

INDEX TERMS Network traffic classification (NTC), IoT, machine learning, network management, knowledge distillation, IoT traffic classification.

I. INTRODUCTION

THE Internet of Things (IoT) has transformed the way we live, work, and interact with the world around us. From smart homes and wearable devices to industrial equipment and smart cities, IoT devices are generating vast amounts of data that must be collected, processed, and analyzed in real time [1]. One important task in IoT data analysis is Network Traffic Classification (NTC), which involves identifying the type of traffic that is being generated by the devices [2]. NTC is critical for ensuring the smooth functioning of IoT networks, enabling efficient resource allocation, and preventing security threats [3], [4]. This information can be used for various purposes, such as optimizing network performance, improving security, and enhancing user experience; However, IoT traffic classification is a challenging task due to the complexity and heterogeneity of IoT traffic, as well as the limited processing and memory resources of IoT devices. While it is acknowledged that the classification process often is not directly executed on these resource-constrained IoT devices [5], the broader IoT infrastructure, especially intermediary layers such as Edge computing, plays a pivotal role. Here, the demand for resource-efficient, real-time solutions

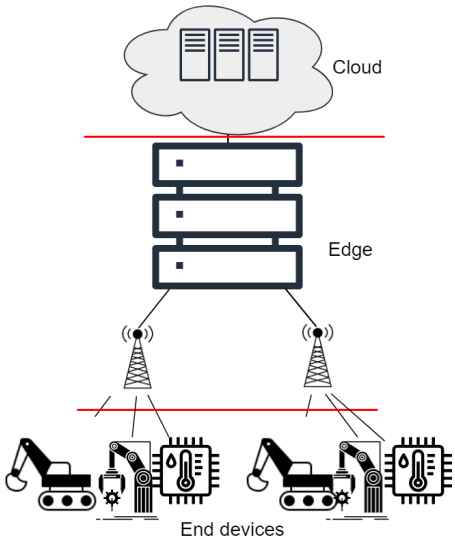


FIGURE 1. **The IoT ecosystem.**

becomes critical. Efficient models not only ensure quicker response times but also reduce data transmission overheads, offering better optimization of bandwidth and energy. This indirectly benefits IoT devices by reducing the computational load on intermediary layers.

Recently, Machine Learning (ML) solutions for NTC have received high attention from the community. Traditional ML methods, such as decision trees and support vector machines, have been used for traffic classification in the past [6]. However, these methods are often limited in their ability to handle the large volumes of data generated by IoT devices and may not be able to adapt to the rapidly changing traffic patterns [7].

Deep Learning (DL) as a part of ML has emerged as a promising approach for NTC in IoT networks, leveraging the power of neural networks to automatically learn representations from the data [8]. However, DL models are mainly computationally expensive and may not be suitable for deployment on resource-constrained IoT devices [9]. This has led to an interest in developing efficient and lightweight DL models that can be deployed on IoT devices [10]. As a solution, DL methods should be lightened up to be used in IoT devices resulting in various methods to help DL provide lighter models.

Figure 1 illustrates the IoT ecosystem, which includes IoT devices, Edge, and Cloud. It emphasizes that IoT encompasses the entire system, from data generation by IoT devices to processing and decision-making at the intermediary layers. In other words, while IoT devices generate data, Edge computing plays pivotal roles in data processing, analytics, and decision-making. This illustration underscores that IoT encompasses the entire system, from devices to intermediary layers, for efficient and real-time operations.

Knowledge distillation is a technique that has been used to address this issue by compressing large, complex models into smaller, simpler models [11]. In the expansive realm of IoT, where immediacy and adaptability are paramount, knowledge distillation can be transformative. By producing models that are both lightweight and accurate, it paves the way for real-time analytics and decision-making, directly on or closer to IoT devices. This not only alleviates the data transmission loads on networks but also expedites responses, making applications like autonomous vehicles, smart health monitoring, and industrial automation more robust and reactive [11].

Addressing the concern raised about data imbalance and pattern variations among different IoT devices, as highlighted in [1], is essential in the context of IoT traffic classification. Knowledge distillation offers a promising avenue for mitigating these challenges. By transferring knowledge from a larger, more complex model to a smaller one, we enable this small model to learn from the broader dataset represented by the large model. This transfer of knowledge helps the student model capture nuances in IoT traffic patterns, even those associated with data imbalance and differences among various IoT objects. The distilled model inherits insights from the teacher model's extensive training, potentially improving its ability to classify diverse IoT object patterns more effectively. This adaptive learning process is a key strength of knowledge distillation and can contribute to more robust and accurate IoT traffic classification, as demonstrated in our experiments.

The motivation for this study is to investigate the efficacy of knowledge distillation for IoT traffic classification. Specifically, we aim to compare the performance of a traditional ML model (big model) with that of a student model trained using knowledge distillation (distilled model). We hypothesize that the student model will be able to achieve comparable or even better performance than the big model while requiring fewer computational resources (the number of learnable parameters) and less storage space (the size of the model on disk). The results of this study could have implications for the development of more efficient and effective IoT traffic classification models.

By using knowledge distillation, it is possible to train a smaller and more efficient model for IoT traffic classification that can perform as well as or even better than larger and more complex models [12], [13]. This could lead to more efficient and effective IoT traffic classification systems, which help to improve network performance, enhance security, and provide a better user experience.

In this study, we seek to respond to this question that *how effective is knowledge distillation for IoT traffic classification?*

We examine our hypothesis that the student model trained using knowledge distillation will be able to achieve comparable performance to the traditional big DL model for IoT traffic classification while requiring fewer computational resources and less storage space.

To test this hypothesis, we conduct experiments using two datasets and compare the performance of the models through various metrics such as accuracy, recall, F1 score, size on the disk, number of parameters, and training time. The results of

the experiments strongly supported our hypothesis suggesting that knowledge distillation is an effective technique for compressing models for IoT traffic classification. This could have important implications for the development of more efficient and effective IoT traffic classification systems, which could help to improve network performance, enhance security, and provide a better user experience.

The remaining sections of the paper are structured as follows: Section II presents the background information on the topic and an overview of the related work. Section III discusses various aspects of IoT traffic classification. Detailed information about our method is presented in Section IV-A, and the results are thoroughly discussed in Section V.

II. BACKGROUND AND RELATED WORK

In this section, we begin by offering the reader essential context and background information regarding knowledge distillation. Subsequently, we present an overview of the latest knowledge distillation approaches employed in traffic classification, with a specific focus on IoT traffic classification.

A. KNOWLEDGE DISTILLATION

Knowledge distillation is a technique of ML that involves transferring knowledge from one model to another. The idea is to train a smaller, more lightweight model to mimic the behavior of a larger, more complex model by learning from the teacher's outputs [14], [15]. More specifically, the basic idea of knowledge distillation is to train the student model to predict the same outputs as the teacher model, but with fewer computational resources [16]. This is achieved by using the teacher model's outputs as *"soft targets"*, also known as soft labels, during training, rather than the *"hard labels"* used in standard supervised learning. Soft targets are probability distributions over the possible outputs, rather than the actual output values themselves. By using these soft targets, the student model is encouraged to learn a more general and robust representation of the data.

Knowledge distillation has been used successfully in a variety of applications, including image classification [17], object detection [18], and natural language processing [19]. It is particularly useful in scenarios where computational resources are limited, such as on mobile devices or embedded systems. Additionally, it has been shown that knowledge distillation can improve the performance of the student model, even when the teacher model is not perfect, by helping the student model learn from the teacher's mistakes [20].

The basic mathematics of knowledge distillation can be understood through the following steps:

- **Softmax function:** The softmax activation function is used to convert the output of a neural network (logits or scores) into probabilities. It is defined as follows:

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{1}$$

where z_i is the input to the i^{th} neuron, K is the total number of neurons, and e is the base of the natural logarithm. The softmax function ensures that the output probabilities sum up to one.

- **Temperature parameter:** In knowledge distillation, the teacher model's output is modified using a temperature parameter T. The modified output probabilities are given by:

$$q_i = \frac{e^{z_i/T}}{\sum_{j=1}^{K} e^{z_j/T}} \tag{2}$$

where z_i is the input to the i^{th} neuron of the teacher model and K is the total number of neurons. The temperature parameter T controls the "softness" of the probabilities. A higher temperature value resulting in a softer probability distribution, while a lower temperature value resulting in a sharper distribution.

- **Distillation loss:** The distillation loss is used to measure the difference between the output probabilities of the student model and the modified output probabilities of the teacher model. The distillation loss is defined as:

$$L_{dist} = \alpha T^2 \sum_{i=1}^{K} q_i log \frac{q_i}{p_i} \tag{3}$$

where α is a hyperparameter that controls the importance of the distillation loss, T is the temperature parameter, q_i is the modified output probability of the teacher model for the i^{th} neuron and p_i is the output probability of the student model for the i^{th} neuron.

- **Total loss:** The total loss is a weighted sum of the distillation loss and the standard cross-entropy loss between the output probabilities of the student model and the ground truth labels. The total loss is defined as:

$$L_{total} = (1-\alpha)L_{CE} + \alpha T^2 \sum_{i=1}^{K} q_i log \frac{q_i}{p_i} \tag{4}$$

where α is a hyperparameter that balances the importance of the two terms in the loss function, and L_{CE} is the cross-entropy loss between the output probabilities of the student model and the ground truth labels. By minimizing the total loss, the student model learns to mimic the behavior of the teacher model, resulting in improved performance.

B. KNOWLEDGE DISTILLATION FOR IoT TRAFFIC CLASSIFICATION

There are few studies that have investigated the use of knowledge distillation for compressing models for IoT traffic classification. For example, in [13] the authors proposed a framework for compressing deep neural networks for IoT traffic classification using two-step knowledge distillation. the study shows that their approach achieved comparable performance to the original deep neural network while requiring significantly fewer computational resources.

Another study has been done by Zhanf et al. in [21] proposed a method for the prediction of the space-time industrial IoT data based on knowledge distillation. The study shows that their approach achieved significant compression while maintaining high accuracy through compressing teacher network to multi-student networks.

A recent study by Zhao et al. [12] proposed an Intrusion Detection (ID) method based on semi-supervised Federated Learning (FL) and knowledge distillation. The authors proposed to use knowledge distillation to solve the problems related to FL for ID, such as model parameters transmission, non-independent and identically distributed (i.i.d) data, and high communication overhead. Similar work has been conducted in [22], in which the authors improved the performance of FL in edge settings through self-knowledge distillation. Indeed, the study proposed to personalize FL to train models to perform well for individual edge devices.

The study by Zhang et al. [23] provides a comprehensive survey of various methods and applications for compressing deep neural networks in the context of IoT. The authors discuss the motivation behind compressing deep models, including the need to reduce the memory and computational requirements of deep neural networks to enable their deployment on resource-constrained IoT devices. The study then presents a detailed review of various compression techniques, such as weight pruning, quantization, knowledge distillation, and model compression, along with their advantages and limitations. The authors also provide an overview of various applications of compact deep learning in IoT, such as image classification, object detection, and speech recognition.

The work has been done in [12] proposes a semi-supervised FL scheme for ID, which leverages knowledge distillation to improve the performance of the student model. The proposed scheme addresses the challenges of traditional supervised and unsupervised approaches to ID by combining the benefits of semi-supervised learning, FL, and knowledge distillation. The proposed scheme consists of a server and multiple clients, where each client has access to its own local data and a small portion of labeled data from other clients. The server initiates the training process by sending a pre-trained teacher model to each client, which is used to generate soft labels for the local data. The clients then use their labeled and unlabeled data to train their respective student models, which are subsequently sent back to the server for aggregation.

Last but not least, the study in [24] provides a review of the use of knowledge distillation in federated edge learning, which is a distributed learning approach that involves extending the FL paradigm to edge devices like smartphones and IoT devices. The review focuses on previous research on knowledge distillation in this context and highlights the potential benefits, challenges, and future directions of the proposed approaches.

In addition to the aforementioned studies, several recent works have further contributed to our understanding of network traffic classification methods, especially concerning malware. The work in [25] reflects the growing trend towards streamlined and efficient models. This work delineates the benefits of a broad learning architecture. In doing so, it joins the conversation on the imperative of creating lightweight, yet effective models for real-world IoT applications. This study in [26] stands out for its innovative approach utilizing domain adaptation and ladder networks. Focusing on enhancing security in the industrial IoT context, it underscores the importance of robust and adaptable methodologies in the ever-evolving realm of IoT. In addition, the authors in [27] investigate the realm of privacy preservation. This research introduces an attention mechanism embedded within neural networks. Such an approach not only champions efficiency but also underscores the criticality of maintaining privacy when classifying malware traffic.

These studies further cement the relevance and urgency of developing efficient methods for network traffic classification, particularly in the IoT context. Overall, these works, along with the ones we've previously discussed, show that knowledge distillation (model compression/lightweight model) is a promising technique for compressing models for IoT traffic classification. However, there is still a need for further research to explore the efficacy of knowledge distillation across different datasets and scenarios, specifically for NTC, and to show the performance of the knowledge distillation in this research area.

While the existing research on knowledge distillation for IoT traffic classification has shown promising results, there are still some research gaps that need to be addressed. Here are some of the main research gaps:

- ***Limited comparison with other compression techniques:*** while the research works have shown that knowledge distillation can effectively compress models for IoT traffic classification, there is a need for more comparisons with other compression techniques, such as quantization, pruning, and low-rank factorization [28]. This would help to determine the strengths and weaknesses of knowledge distillation in comparison to other techniques.
- ***Lack of standard datasets:*** Most of the existing studies on knowledge distillation for IoT traffic classification have used their own proprietary datasets, which makes it difficult to compare their results with other studies. There is a high need for standard datasets that can mimic the real world to compare the performance of different compression techniques.
- ***Limited analysis of the trade-off between compression and performance:*** While the studies have shown that knowledge distillation can effectively compress models for IoT traffic classification, there is a need for more analysis of the trade-off between compression and performance. Specifically, it is important to determine the optimal compression rate that can be achieved without cruelly sacrificing performance.
- ***Limited evaluation on edge devices:*** Most of the existing studies on knowledge distillation for IoT traffic

classification have focused on evaluating the performance of compressed models on high-performance servers; However, new technologies e.g., edge computing have other requirements. In this case, there is a need for more evaluation of less powerful devices, such as embedded systems and mobile devices, which have limited computational resources and storage space.

Addressing the aforementioned research gaps would help to further advance the use of knowledge distillation for compressing models for IoT traffic classification and provide a better understanding of its strengths and limitations.

III. IMPLICATIONS AND REQUIREMENTS OF IoT TRAFFIC CLASSIFICATION

Traffic classification is the systematic identification and categorization of data flows originating from network nodes, including IoT devices [29]. Essentially, the task of traffic classification can be expressed as a problem of multi-class classification.

Let X be the set of all network traffic flows generated by network devices, and let $Y = y_1, y_2, \ldots, y_m$ be the set of m pre-defined classes corresponding to different types of services, IoT devices, or applications. The goal of traffic classification is to learn a function $f : X \rightarrow Y$ that maps each traffic flow $x \in X$ to its correct class label $y \in Y$. This function is typically learned from a training dataset $D = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ of n labeled traffic flows, where each sample (x_i, y_i) consists of a traffic flow x_i and its corresponding class label y_i. Formally, the problem of network traffic classification can be stated as follows: given a training dataset D of labeled traffic, learn a function f that minimizes the expected classification error on a new, unseen test dataset D_{test}.

Network traffic classification stands as a critical pillar in the management of today's intricate digital landscapes. At its core, it is more than just categorizing data flows; it is about optimizing sprawling networks that grapple with a huge amount of data on a daily basis [30]. Through effective classification, network operators can discern and understand intricate traffic patterns, enabling them to prioritize specific data streams, allocate bandwidth with precision, and ensure the smooth flow of operations. Beyond optimization, the arena of digital security heavily relies on the robustness of traffic classification. As digital threats continue to advance, a well-classified network becomes the frontline defense, allowing operators to detect anomalies and potential threats promptly, especially vital for devices perpetually connected to the Internet. Furthermore, in the digital age, not all data packets hold the same weight. While some demand swift transmissions, such as in real-time gaming or video conferencing, others might have more flexibility. This is where classification comes into play, recognizing the diverse needs of each packet, and ensuring they are treated based on their specific latency, bandwidth, and reliability prerequisites. Finally, in sectors that are bound by rigorous data governance norms, the importance of classification escalates further. Whether it is a financial institution dealing with confidential transactional data or a healthcare entity managing patient records, a well-structured network traffic ensures unwavering data integrity, privacy, and adherence to regulatory standards.

A. IoT TRAFFIC CLASSIFICATION

While the foundational principles of traffic classification remain steadfast, IoT introduces a paradigm shift. Comprising billions of interconnected devices, ranging from smart thermostats to intricate industrial sensors, IoT not only magnifies the volume of network traffic but also introduces unparalleled diversity and complexity. Each device, with its unique operational metrics, contributes to an intricate web of network traffic, necessitating specialized approaches for effective management. More specifically, the data sources it relies upon are notably different. While general traffic classification often employs network logs or packet captures as primary data sources, IoT traffic classification is intrinsically dependent on data amassed directly from IoT devices [31]. These devices, in turn, exhibit a wide array of traffic types and patterns, diverging from the standard web, email, or multimedia traffic patterns commonly seen in broader network classifications.

This distinctive nature of IoT traffic is also reflected in the patterns it exhibits [32]. For instance, traffic stemming from sensors, actuators, or control systems remains peculiar to IoT devices, differentiating it from the commonplace traffic patterns of general network classifications.

Moreover, ML models employed for IoT traffic classification are specifically tailored to address its unique attributes. Given the high volume, low signal-to-noise ratio, and inherent variability of IoT traffic, specialized ML models are indispensable. In contrast, traditional network traffic classification might leverage a more generalized range of models, from deep neural networks to decision trees or support vector machines.

Furthermore, the scope of applications for IoT traffic classification extends beyond just data categorization. It plays a pivotal role in aspects like network optimization, security enhancement, ensuring Quality of Service (QoS), and maintaining regulatory compliance [33]. Meanwhile, other network traffic classifications, such as that for video streaming, might predominantly focus on bandwidth monitoring and performance assessment.

Lastly, the security landscape for IoT presents its own set of challenges. With a plethora of connected devices, the potential threats and vulnerabilities multiply. There is an ever-present risk of malicious entities aiming to exploit network vulnerabilities. In such a landscape, precise and accurate traffic classification becomes a frontline defense, aiding in the timely detection and mitigation of these looming threats.

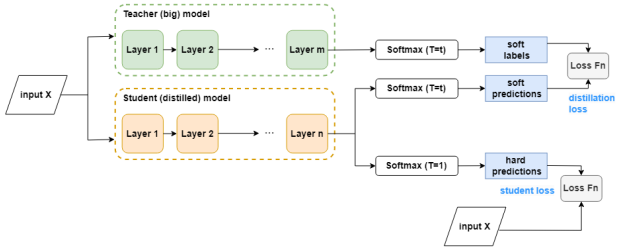


FIGURE 2. Framework for knowledge distillation.

IV. METHODS

A. KNOWLEDGE DISTILLATION FRAMEWORK

In this study, we use the knowledge distillation framework presented by Hinton et al. [11]. The goal is to transfer the knowledge from the larger model to the smaller model so that it can perform similarly to the larger model but with a smaller computational footprint. Here is a brief description of the knowledge distillation framework for IoT traffic classification (see figure 2).

- **Data preprocessing:** The first step in the methodology is to preprocess the IoT traffic data. This stage involves cleaning the data to remove any irrelevant or corrupted data samples and converting the raw network traffic data into a format suitable for the training and testing of ML models. This step also involves feature extraction to capture relevant characteristics of the network traffic, such as packet size, protocol type, and source/destination IP addresses. Let X be the preprocessed dataset of network traffic samples with features $x_1, x_2, \ldots, x_n$ and their corresponding true labels $y_1, y_2, \ldots, y_n$, where n is the number of samples.
- **Pre-train the larger model:** Train a large and complex model (teacher) to perform the classification task on the training data. This larger model would have a higher accuracy than the smaller model we are going to train. To investigate different aspects of knowledge distillation for IoT traffic classification, we use multiple teacher and student architectures, as we describe in the sequel. For now, let f_T be the teacher model that is trained to predict the true labels of the IoT traffic data. The output of the teacher model for a given sample x_i is denoted as p_{iT}, which is a probability distribution over the possible classes or soft labels.
- **Generate soft labels:** We use the pre-trained larger model to generate ''soft labels'' for the training data. Soft labels are probability distributions over the classes, rather than ground truth labels. Soft labels provide more information than ground truth labels and allow the smaller model to learn from the uncertainty in the larger model's predictions.
- **Train the smaller model:** Train a smaller model (student) to learn from the soft labels generated by the larger model. This process involves minimizing the difference between the soft labels generated by the larger model and the soft predictions made by the smaller model. In this stage, the distillation loss is achieved by minimizing the difference between the teacher and the student models. In this study, we utilize the cross-entropy loss to determine the dissimilarity between the teacher and student models. Let f_S be the student model that is trained to mimic the predictions of the teacher model. The output of the student model for a given sample x_i is denoted as p_{iS}, which is also the probability distribution over the possible classes.
- **Fine-tune the smaller model:** Fine-tune the smaller model on the training data using the standard ''hard labels'', also known as ''hard predictions''. This process involves minimizing the cross-entropy loss between the predicted labels and the true labels.
- **Performance evaluation:** After the student model is trained, we evaluate its performance on the test set. We also compare the performance of the student model with that of the teacher model and the smaller model trained from scratch to assess the effectiveness of the knowledge distillation.

Algorithm 1 provides an algorithmic description of the knowledge distillation framework for IoT traffic classification.

B. DATASET DESCRIPTION

In our experiments, we utilized two widely used, real-world network traffic datasets that are frequently employed in research on network traffic classification. These datasets include the following:

- The University of Cambridge dataset: This dataset was created by the University of Cambridge, Computer Laboratory and contains ten datasets [34]. Generally, the University of Cambridge dataset is a widely used and well-established dataset for traffic classification tasks and has been used in several previous studies on this topic (e.g., [35]). Each dataset is unique, with its own distinct start time, end time, duration, number of traffic flows, and type of traffic. Note that for the purposes of our experiment and specific requirements, we have utilized various modified versions of the original dataset. For example, we may remove a specific traffic label or change the size of the training dataset. The dataset covers a wide range of traffic types, such as, such as P2P, ATTACK, and DATABASE.
  To provide a more diverse sample of mixing throughout the day, the start time for each sample was selected randomly in this dataset. Each dataset describes a period captured within the day. The dataset also contains features such as median inter-arrival time, maximum packet inter-arrival time, and variance of control bytes packet which can be used to classify the traffic.
  We split the dataset into training and testing sets, with 70% of the data used for training and 30% used for

Algorithm 1 :Knowledge distillation IoT traffic classification

1: **Input:** Preprocessed dataset X with features $x_1, x_2, \ldots, x_n$ and true labels $y_1, y_2, \ldots, y_n$; Hyperparameters: temperature T, α, learning rate γ, momentum, and number of epochs.
2: **Output:** Student model f_S.
3: ***Step 1:*** *Define the teacher model f_T:*
- Initialize f_T, a pre-trained neural network model
- Train f_T on the dataset X by minimizing the cross-entropy loss with the stochastic gradient descent (SGD) optimizer
- Obtain the probability distribution p_T for each sample x_i in X using the model f_T

4: ***Step 2:*** *Define the student model f_S:*
- Initialize the student model f_S with a smaller architecture than f_T
- Train f_S on the dataset X by minimizing the knowledge distillation loss function with SGD optimizer
- Obtain the probability distribution p_S for each sample x_i in X using f_S

5: ***Step 3:*** *Train the student model using the knowledge distillation technique:*
- ***For each epoch*** t ***in*** $[1, N]$***:***
- ***For each sample*** x_i ***in*** X***:***
- Compute the logits (scores) s_T and s_S for f_T and f_S, respectively
- Compute the probability distribution q_T and q_S using the softmax function and with temperature T: $q_T = \text{softmax}(s_T/T)$, $q_S = \text{softmax}(s_S/T)$
- Compute the cross-entropy loss function $L_{CE}(p_T, y)$ and the knowledge distillation loss L_{dist}
- Compute the total loss function L_{total}
- Update the weights of f_S using the gradient of L_{total} with respect to the weights and the learning rate γ

6: ***Step 4:*** *Evaluate the student model*
- Evaluate the performance of f_S on a held-out test dataset using various performance metrics such as accuracy, recall, and F1 score

7: ***Step 5:*** *Fine-tune the student model*
- If the performance of f_S is not satisfactory, fine-tune f_S by adjusting its hyperparameters or training it for longer

8: ***Step 6:*** *Compare the performance of the student and teacher models*
- Compare the performance of f_S with that of f_T to assess the effectiveness of the knowledge distillation approach in reducing model complexity without sacrificing performance

testing. We also performed some data preprocessing, including normalization and feature selection, to prepare the data for the ML models.

- CIC-IDS2017 dataset: The CIC-IDS2017 dataset is a network traffic dataset that was captured in a real-world IoT network environment. It contains both benign and malicious traffic, and it includes traffic from several different IoT devices such as cameras, routers, and smart thermostats. The dataset was created by the Canadian Institute for Cybersecurity (CIC) and is widely used for evaluating IDSs for IoT networks [36]. The dataset includes both raw network traffic (pcap files) and pre-processed features (CSV files), which can be used for ML-based classification. The pre-processed features include statistical features such as mean, standard deviation, and maximum value, as well as domain-specific features such as HTTP headers and SSL certificates.
The CIC-IDS2017 dataset contains a total of 2.8 million network flows, of which 1.9 million are benign and 0.9 million are malicious instances. The dataset also includes different types of attacks, such as botnet, DDoS, and port scan attacks.
In our experiments, we use a modified version of this dataset. Each record in the dataset is labeled with a label from a list of nine different labels, such as BENIGN, DDoS, PortScan, and SSH-Patator, among others.

C. EXPERIMENTAL DESIGN AND SETUP

In this study, we conduct a series of comprehensive experiments to evaluate the effectiveness of knowledge distillation in compressing models for IoT traffic classification under varying conditions. We compared the performance of our knowledge distillation model with that of traditional learning models used for this task. For each experiment, we also report the number of epochs and data used for training purposes. We trained and evaluated three distinct models using the datasets:

- Teacher model: The model was constructed using a multi-layer feedforward neural network, which consists of multiple linear layers followed by a softmax output layer for multiclass classification tasks and a sigmoid output layer for binary classification tasks. We trained the model using the SGD optimizer with a learning rate of $\gamma = 0.01$ and with a momentum of 0.9. The model was trained to predict the 9 categories of traffic in the Cambridge dataset and 9 types of traffic in the CIC-IDS2017 dataset. Each experiment is repeated five times, and the mean and the standard deviation are reported.
- Student model: This model has the same architecture as the teacher but with fewer hidden layers and neurons. We train the model using the same hyperparameters as the teacher model.
- A neural network model compressed using knowledge distillation: We use the big model as the teacher model and train a student model using knowledge distillation. To implement knowledge distillation, the student model is trained using both the input data and the soft labels

generated by the teacher model. In terms of the hyperparameters used for knowledge distillation, we follow the widely adopted approach ([11], [37]), using a temperature of $T = 4$, and the α value of 0.9.

As mentioned, we conducted a series of comprehensive experiments to evaluate the effectiveness of knowledge distillation. The architectures of the teacher and student models play a crucial role in the effectiveness of knowledge distillation for IoT traffic classification. The architecture of the teacher and student models can vary depending on the specific experiments and the datasets used. In this study, we employ different architectures for both the teacher and student models, depending on the specific purpose of each experiment.

We evaluate the models based on their accuracy, recall, and F1 score on the test set. We also measure the computational resources required to train and evaluate each model, including the training time, the number of parameters, and the size.

We conducted the experiments on a machine with an AMD Ryzen 5000 Series processor, 16 GB of RAM, and an NVIDIA GeForce RTX 3050 Ti graphics card. We use Python 3.7 and PyTorch to develop the ML models.

By comparing the performance of the different models, we aim to determine whether knowledge distillation can effectively compress models for IoT traffic classification while maintaining high accuracy and reducing computational resources.

V. RESULTS AND ANALYSIS

A. ACCURACY COMPARISON

We compare the performance of three different models for IoT traffic classification: a big model, a small model, and a student model compressed using knowledge distillation. The big neural network consists of four linear layers. The first layer has "$input - dim$" number of inputs and 1024 output neurons, which means it receives the input traffic dataset with "$input - dim$" features. We considered 235 features for the Cambridge dataset, whereas the CIC-IDS2017 dataset was analyzed with only 73 features. The second layer receives input from the previous layer with 1024 neurons. This layer has 1200 output neurons. The third layer has 700 output neurons. It receives input from the previous layer with 1200 neurons. Finally, the fourth and last layer has "$output - dim$" number of output neurons. It receives input from the previous layer with 700 neurons. As previously mentioned, the size of the "$output - dim$" for both datasets is nine.

For the small and student model we consider a simple two-layer feedforward neural network, with an input layer, a hidden layer, and an output layer. The first layer is a Linear layer with "$input - dim$" input neurons and 800 output neurons. The second layer is a linear layer with 800 input neurons and "$output - dim$" output neurons.

The classification results of the Cambridge dataset are presented in Table 1. Additionally, Figure 3 depicts the confusion matrix for the evaluation results on the same dataset. We train the teacher model for 350 epochs, and the small and student models are trained for 350 epochs each in this experiment.

The teacher model achieved the highest accuracy, F1, and recall, with an accuracy of 98.51 and an F1 score of 0.981. However, it also had the highest number of parameters with 2,318,673 parameters, and the size of the trained teacher model on the disk with 9,060 KB.

The small model achieved a lower accuracy, F1 score, and Recall than the teacher model, with an accuracy of 97.13, and an F1 score of 0.971. However, it also had fewer parameters and a faster training time than the teacher model, with 196,009 parameters and a training time of 86.99 S. The size of the small trained model on the disk is 768 KB, which is approximately 11.7 times smaller than the teacher model. Finally, the ML model compressed using knowledge distillation achieved an accuracy of 98.31 and an F1 score of 0.982, which is approximately similar to the teacher model but higher than the smaller model. The student model had considerably fewer parameters compared to the teacher model, containing only 196,009 parameters and a smaller model size of 768 KB. Nevertheless, the training time for the student models is longer than for their counterparts.

We repeated the experiment on the CIC-IDS2017 dataset using the same settings. As we expected, we achieved almost the same results as the first dataset. More specifically, the results indicate that the teacher model is the best-performing model on the CIC-IDS2017 dataset, followed by the student and the small models, respectively; However, the trade-off between the performance and the model size/computation time should be carefully considered when selecting a model for a specific use case. Table 2 presents the classification results of the CIC-IDS2017 dataset. In addition, the evaluation results on the same dataset are illustrated by the confusion matrix shown in Figure 4.

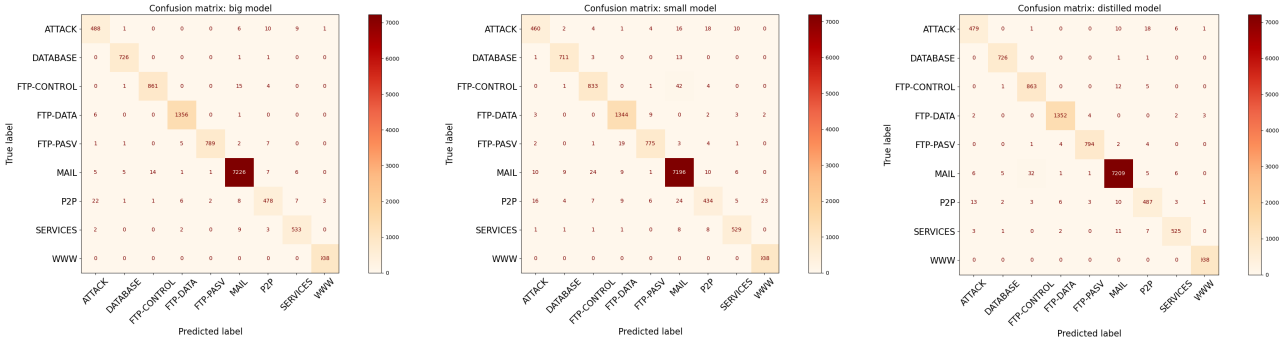
From these observations and the confusion matrices depicted in the figures, we can conclude that the student model is a good compromise between accuracy and efficiency, as it has a similar accuracy to the teacher model while having a smaller size like the small model. Therefore, it can be a suitable alternative for scenarios where computational resources are limited, such as mobile or IoT traffic classification. However, the longer training time may be a drawback in certain applications.

B. THE EFFECTIVENESS OF KNOWLEDGE DISTILLATION WITH EXCLUDED CLASSES

In this experiment, we are exploring the effectiveness of knowledge distillation in the context of removed classes. Specifically, we train a student model to predict a subset of the classes that the teacher model was trained on. To achieve this, we use a transfer dataset where we exclude all examples of different traffic labels. In other words, the student model is trained on a subset of the original dataset. The goal of this experiment is to evaluate how well the student model can learn from the teacher model despite the reduced amount of training data, and whether the knowledge distillation

TABLE 1. Performance of models on the cambridge dataset (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	98.51±0.071	0.985±0.0007	0.985±0.0007	997.52	2,318,673	9060 KB
Small	97.13±0.103	0.971±0.001	0.971±0.001	86.99	196,009	768 KB
Student	98.31±0.14	0.9833±0.002	0.9829±0.002	1227.7	196,009	768 KB

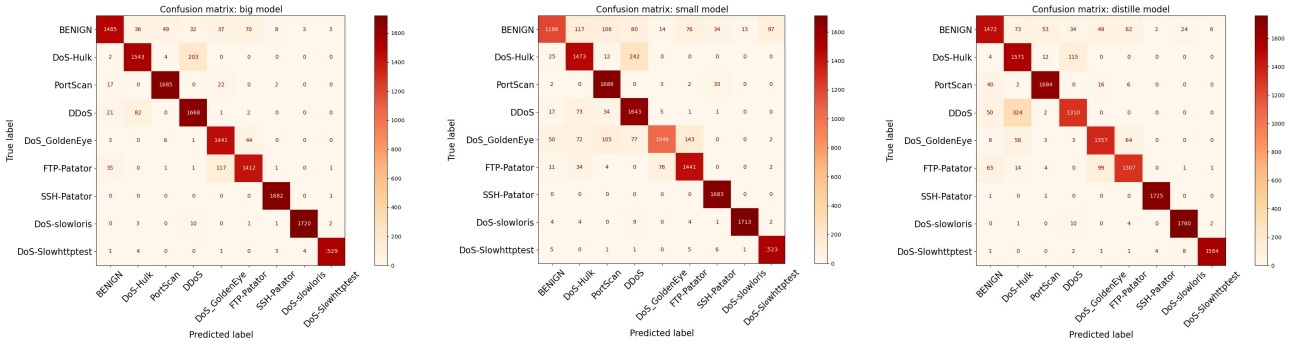


(a) Teacher model (b) Small model (c) Student model

FIGURE 3. Confusion matrix for the evaluation results on the Cambridge dataset.

TABLE 2. Performance of models on the CIC-IDS2017 dataset (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	93.57±0.920	0.935±0.0092	0.935±0.0092	656.63	2,152,785	8,412 KB
Small	88.02±0.103	0.880±0.010	0.880±0.010	28.39	66,409	261 KB
Student	92.45±0.13	0.935±0.009	0.935±0.009	709.03	66,409	261 KB



(a) Teacher model (b) Small model (c) Student model

FIGURE 4. Confusion matrix for the evaluation results on the CIC-IDS2017 dataset.

technique can effectively transfer the knowledge from the teacher model to the student model even when some classes are missing from the training data. In this experiment, we use the same student and teacher architectures as in the previous experiment.

- **Two classes:** In the first experiment, the teacher model is first trained on the Cambridge dataset with all traffic labels, and then the student model is trained to predict a subset of those labels by using a transfer dataset where two classes have been removed. The removed classes are "ATTACK" and "FTP-DATA". The small model is trained on the same transfer dataset as the student model. Table 3 shows the performance of three different models on the Cambridge dataset for traffic classification when two classes are removed from the transfer dataset. The teacher model achieved an accuracy of 97.51% with an F1 score and recall of 0.975. The model took 356.10 seconds for training and has 2,318,673 parameters. Its size is 9,060 KB on the disk. This model was also trained on the full Cambridge dataset with all traffic labels. The most intriguing aspect of this experiment is the

performance of the student model, which achieves the highest accuracy of 97.61%, slightly better accuracy, and recall than the teacher model with two removed classes. According to the confusion matrix (Fig. 5), the student model correctly predicted 397 ''ATTACK'' and 454 ''FTP-DATA'' labels. Moreover, it is clear that the small model performs poorly and provides inadequate predictions, especially for the two removed labels. However, the small model has the smallest training time compared to the other models.

The classes 'DDoS' and 'SSH-Patator' were excluded from the CIC-IDS2017 dataset in the second experiment. Table 4 presents the classification performance of three distinct models applied to the CIC-IDS2017 dataset. Based on the table, the student model has the highest accuracy of 95.06%, followed by the teacher model with an accuracy of 94.86%. The small model has the lowest accuracy of 72.77% $\pm$ 0.32%. Both the teacher and student models have almost similar classification performance. The teacher model took the longest time to train, at 1,555 seconds, while the small model took the shortest time, at 43.7 seconds. The student model took 1,110 seconds to train. Finally, the teacher model has the highest number of parameters, at 2,153,864, while the small and student models have 68,012 parameters. The size of all models is relatively small, with the largest being only 8,412 KB.

The results suggest that knowledge distillation can still be an effective technique for creating smaller models with comparable performance to larger, more complex models, despite being trained on a transfer dataset with two classes removed.

It is also worth noting that the training time of the teacher model is much longer than the student and small models in both datasets. Therefore, one advantage of knowledge distillation is that it can produce smaller models that require less training time while still maintaining comparable performance to larger models. Training time is an important factor to consider in real-world applications, especially in time-sensitive environments such as IoT networks where quick response times are necessary. Longer training times may not be feasible in such scenarios, and therefore, more efficient techniques for knowledge distillation may be required as the training time of the student models is still longer than the training time for the small model in both cases.

- **Three and four classes:** To obtain a more thorough comprehension of the efficacy of knowledge distillation in a scenario where certain traffic classes are excluded, we conduct additional experiments by removing multiple classes from the transfer dataset. Specifically, we repeat the experiment by removing three and four classes from the transfer dataset during the training of both the student and small models. Tables 5, 6, 7, and 8 provide detailed performance metrics, while Figures 7, 8, 9, and 10 offer confusion metrics of the results.

In all tables except Table 4, the teacher model (i.e., the baseline model without any removed classes) has the highest accuracy and F1 scores. This is expected, as the model was trained on the full dataset and can therefore classify all classes of the data accurately. When three classes are removed, both the small and student models show a decrease in performance compared to the teacher model. The decrease was much more severe in the small models than in the student models. The small models have the lowest accuracy, F1 score, and recall, while the student models have better performance but still lag behind the teacher models. This indicates that removing even a small subset of classes can have a detrimental impact on the performance of the models.

The results of these experiments further confirm the efficacy of the proposed approach for traffic classification. Despite the reduced amount of training data, the student model was able to achieve high accuracy, F1 score, and recall, indicating that the knowledge distillation technique was effective in transferring the knowledge from the teacher model. The small model, on the other hand, was not able to achieve comparable performance. Furthermore, we can observe that as the number of removed classes increases, the performance of the student model begins to degrade slightly. This is to be expected, as the model has less data to learn from and fewer examples to draw upon. However, even when four classes were removed, the student model was still able to achieve impressive results, indicating the robustness of the proposed approach.

C. ANALYZING THE TEACHER CAPACITY

To analyze the impact of the teacher model capacity on the performance of the knowledge distillation approach in the context of IoT traffic classification, we conducted a series of experiments. In these experiments, we train multiple teacher models with varying capacities using the same training dataset. To control the other factors that may affect performance, such as the size and complexity of the student model and the amount of training data, we use the same student model and training dataset across all the experiments. Additionally, we use the same hyperparameters for all the experiments.

To train teacher models of varying capacities, we adjust the architecture of the models. More specifically, we modify the number of layers (depth). In the first attempt, we consider a teacher model with three linear layers. The first layer takes an input size of ''$input - dim$'' and outputs a tensor of size 1024. The second layer takes this tensor of size 1024 and outputs a tensor of size 700. Finally, the third layer takes this tensor of size 700 and outputs a tensor of size ''$output - dim$''. This architecture provides a limited capacity teacher model that can serve as a baseline for comparing with the teacher models of different capacities. We use the same architecture for the student model as the previous experiment.

Table 9 shows the performance of different models on the Cambridge dataset using a three-layer teacher model. The teacher model, which serves as a baseline for comparison,

TABLE 3. Performance of models on the cambridge dataset with two removed classes (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	97.51±0.16	0.975±0.001	0.975±0.001	356.10	2,318,673	9,060 KB
Small	76.90±0.40	0.769±0.004	0.769±0.004	21.49	196,009	768 KB
Student	97.61±2.9	0.974±0.02	0.977±0.02	285.5	196,009	768 KB

TABLE 4. Performance of models on the CIC-IDS2017 Dataset with two removed classes (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	94.86±1.18	0.948±0.011	0.948±0.011	1,555	2,153,864	8,412 KB
Small	72.77±0.32	0.727±0.0032	0.727±0.0032	43.7	68,012	268 KB
Student	95.06±0.95	0.950±0.0095	0.950±0.0095	1,110	68,012	268 KB

TABLE 5. Performance of models on the cambridge dataset with three removed classes (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	97.55±0.17	0.975±0.001	0.975±0.001	340.87	2,318,673	9060 KB
Small	65.9±0.66	0.655±0.001	0.655±0.001	18.09	196,009	768 KB
Student	97.3±0.21	0.973±0.002	0.973±0.002	242.9	196,009	768 KB

TABLE 6. Performance of models on the CIC-IDS2017 dataset with three removed classes (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	93.82±1.38	0.938±0.013	0.938±0.013	1,184	2,153,864	8,417
Small	63.48±0.52	0.634±0.005	0.634±0.005	35	68,012	268 KB
Student	93.29±1.38	0.932±0.013	0.932±0.013	895	68,012	268 KB

TABLE 7. Performance of models on the cambridge dataset with four removed classes (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	97.91±0.16	0.979±0.001	0.979±0.001	469.00	2,318,673	9060 KB
Small	55.94±0.66	0.559±0.006	0.559±0.006	8.31	196,009	768 KB
Student	96.22±0.87	0.979±0.001	0.979±0.001	116.25	196,009	768 KB

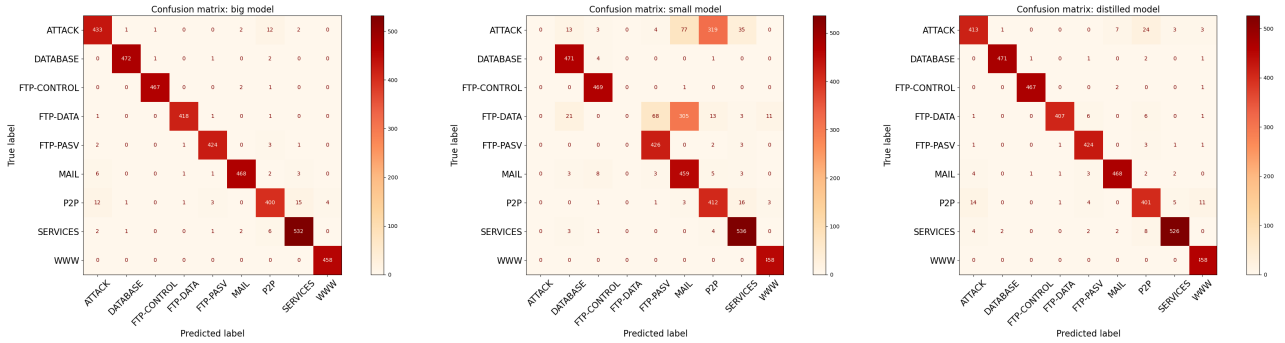
TABLE 8. Performance of models on the CIC-IDS2017 Dataset with four removed classes (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	94.95±0.76	0.949±0.007	0.949±0.007	1,564	2,153,864	8,417 KB
Small	52.72±0.57	0.527±0.005	0.527±0.005	30	68,012	268 KB
Student	93.99±1.24	0.939±0.012	0.939±0.012	702	68,012	268 KB

achieved an accuracy of 98.79/ an F1 score of 0.98 on the test dataset. It took 708.22 seconds to train, having 965,473 parameters, and a size of 3774 KB. Generally, the results suggest that the small model has a lower accuracy compared to the student model and the teacher model.

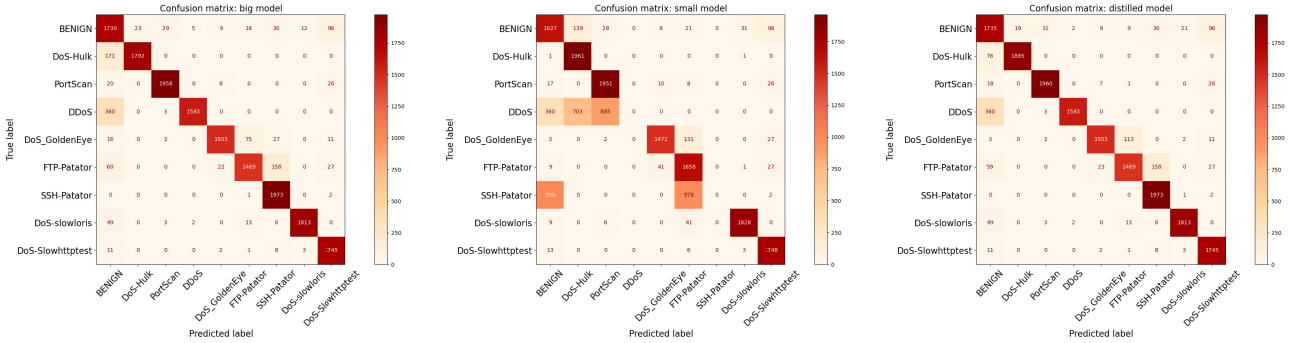
To better assess the influence of teacher capacity, we carried out additional experiments wherein we trained teacher models with four, five, six, and seven layers. In doing so, we incrementally added one, two, three, and four hidden layers with a size of 1024*1024, respectively. We repeated the same experimental setup as before, using the same student model, training dataset, and hyperparameters across all experiments. Moreover, upon conducting a more in-depth analysis of the CIC-IDS2017 dataset, we discovered that the results were strikingly similar to those of the previous dataset, highlighting a consistent pattern in our findings(see Fig. 12). Looking at the results in Fig. 11 and Fig. 12, we can see that increasing the number of layers in the teacher model did not necessarily result in a significant improvement in performance. While the seven-layer teacher model had slightly better performance than the three-layer teacher model in terms of accuracy, F1 score, and recall, the difference was not significant.

One interesting finding is that the training time and model



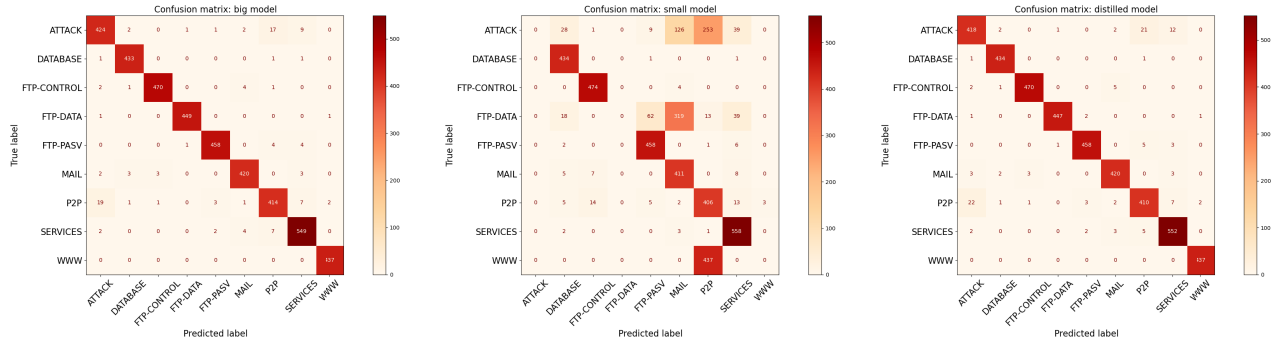
(a) Teacher model (b) Small model (c) Student model

FIGURE 5. Confusion matrix for the evaluation results on the Cambridge dataset, two removed classes.



(a) Teacher model (b) Small model (c) Student model

FIGURE 6. Confusion matrix for the evaluation results on the CIC-IDS2017 dataset, two removed classes.



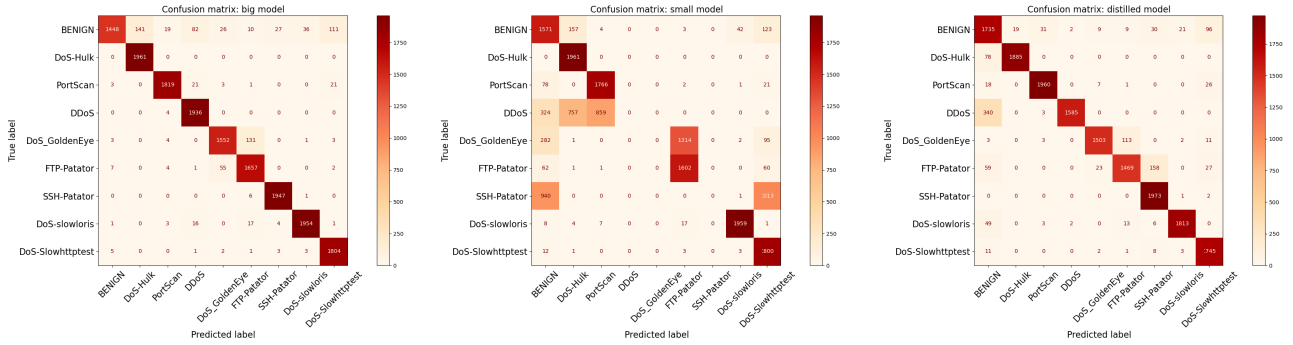
(a) Teacher model (b) Small model (c) Student model

FIGURE 7. Confusion matrix for the evaluation results on the Cambridge dataset, three removed classes.

size increased significantly as the number of layers in the teacher model increased. The seven-layer teacher model had the highest training time and model size, which indicates that increasing the model capacity beyond a certain point may not be worth the additional computational cost. Additionally, the results suggest that the knowledge distillation approach was effective in transferring knowledge from the teacher model to the student model, as the student model achieved a high level of performance despite having a much smaller capacity than the teacher models.

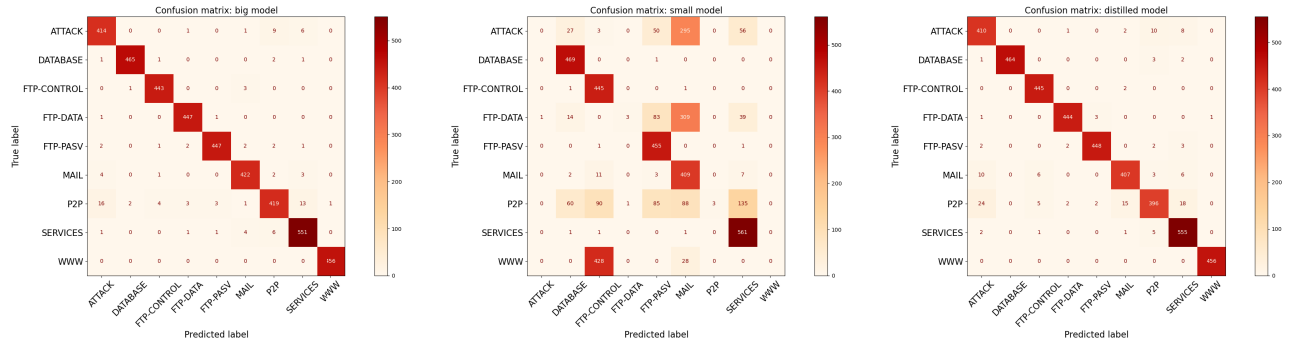
D. MODEL GENERALIZATION USING FAR LESS TRAINING DATA

Our main argument in favor of using soft targets instead of hard labels is that soft labels can contain a wealth of useful information that cannot be captured by a single hard label. This means that the student model is able to learn and capture the essential features of the IoT traffic data with fewer data samples, which is desirable as collecting and labeling traffic data can be a time-consuming and expensive process [2]. To verify our argument, we conducted an experiment where



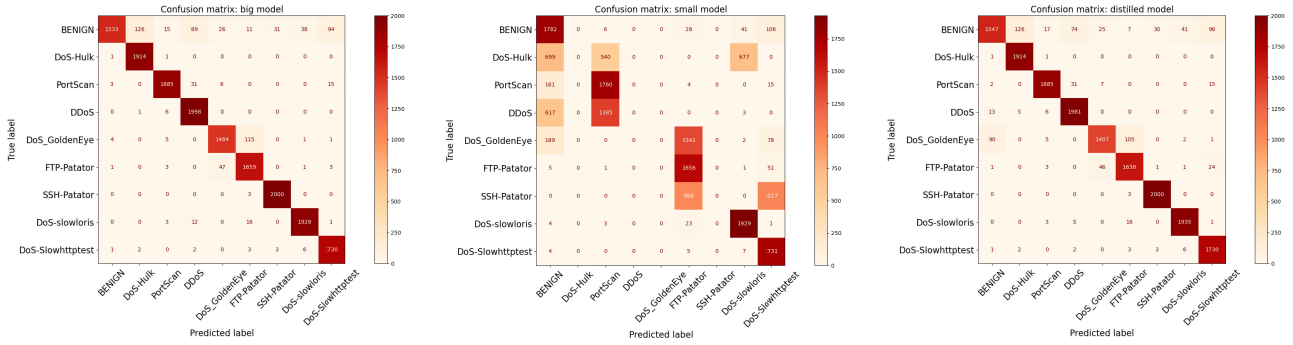
(a) Teacher model (b) Small model (c) Student model

FIGURE 8. Confusion matrix for the evaluation results on the CIC-IDS2017 dataset, three removed classes.



(a) Teacher model (b) Small model (c) Student model

FIGURE 9. Confusion matrix for the evaluation results on the Cambridge dataset, four removed classes.



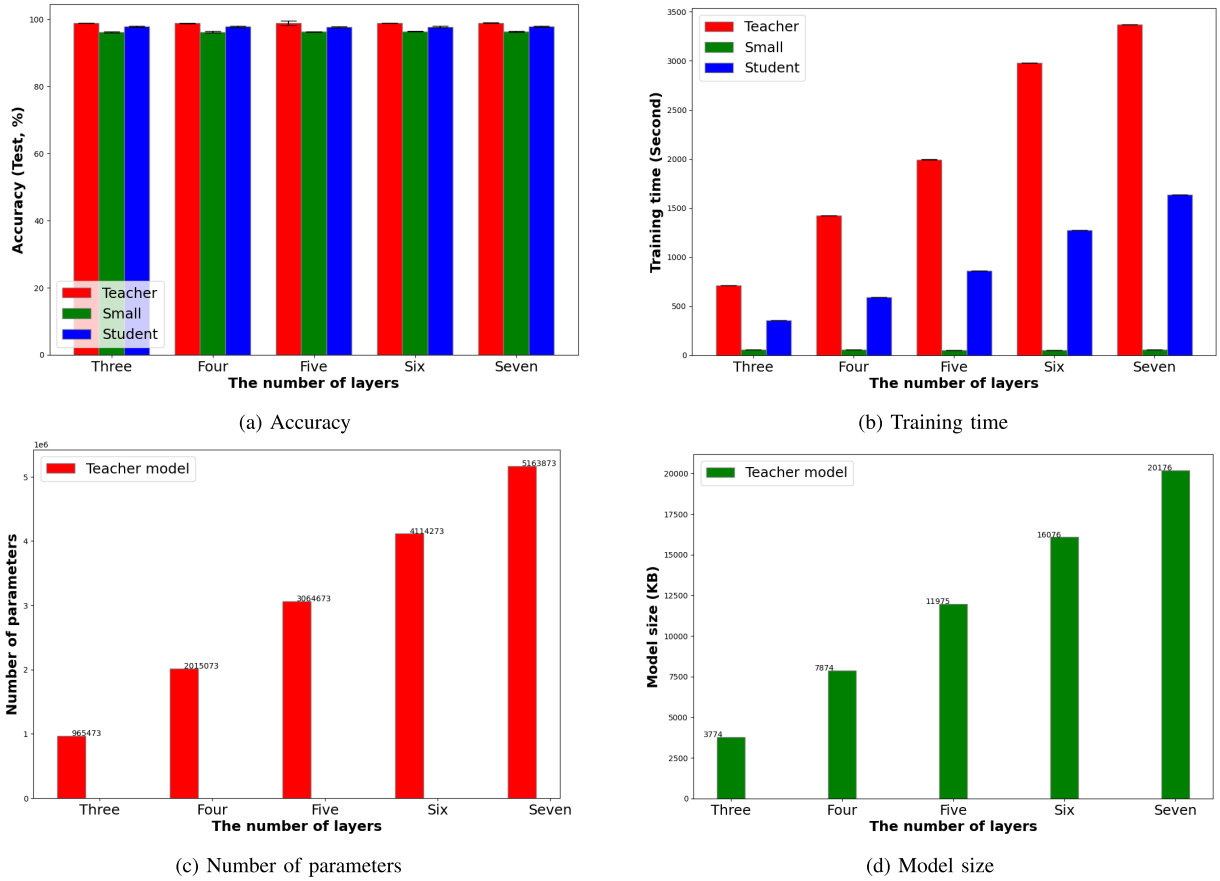
(a) Teacher model (b) Small model (c) Student model

FIGURE 10. Confusion matrix for the evaluation results on the CIC-IDS2017 dataset, four removed classes.

TABLE 9. Performance of models on the cambridge dataset with a three-layerteacher model (Test, %).

Model	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher	98.79±0.06	0.987±0.0006	0.987±0.0006	708.22	965,473	3,774 KB
Small	96.08±0.13	0.96±0.0006	0.96±0.0006	55.26	196,009	768 KB
Student	97.73±0.87	0.977±0.002	0.977±0.002	353.76	196,009	768 KB

we trained the student model using significantly less data than in previous trials. Specifically, we trained our model on a mere 3% of the available data for the Cambridge and CIC-IDS2017 datasets, comprising 949 and 615 data instances, respectively, using hard labels. Remarkably, our model achieved an accuracy of 94.2% and 87.98% for the



(a) Accuracy

(b) Training time

(c) Number of parameters

(d) Model size

FIGURE 11. Comparison of classification accuracy, training time, number of parameters, and model size on the Cambridge dataset with varying number of layers.

two datasets, demonstrating the efficacy of our approach even with limited data. The results presented in the tables 10 and 11 show that even with only 3% of the data, the student model trained with soft labels achieved a high level of accuracy, indicating that the model was able to effectively capture the essential features of the IoT traffic data with limited data.

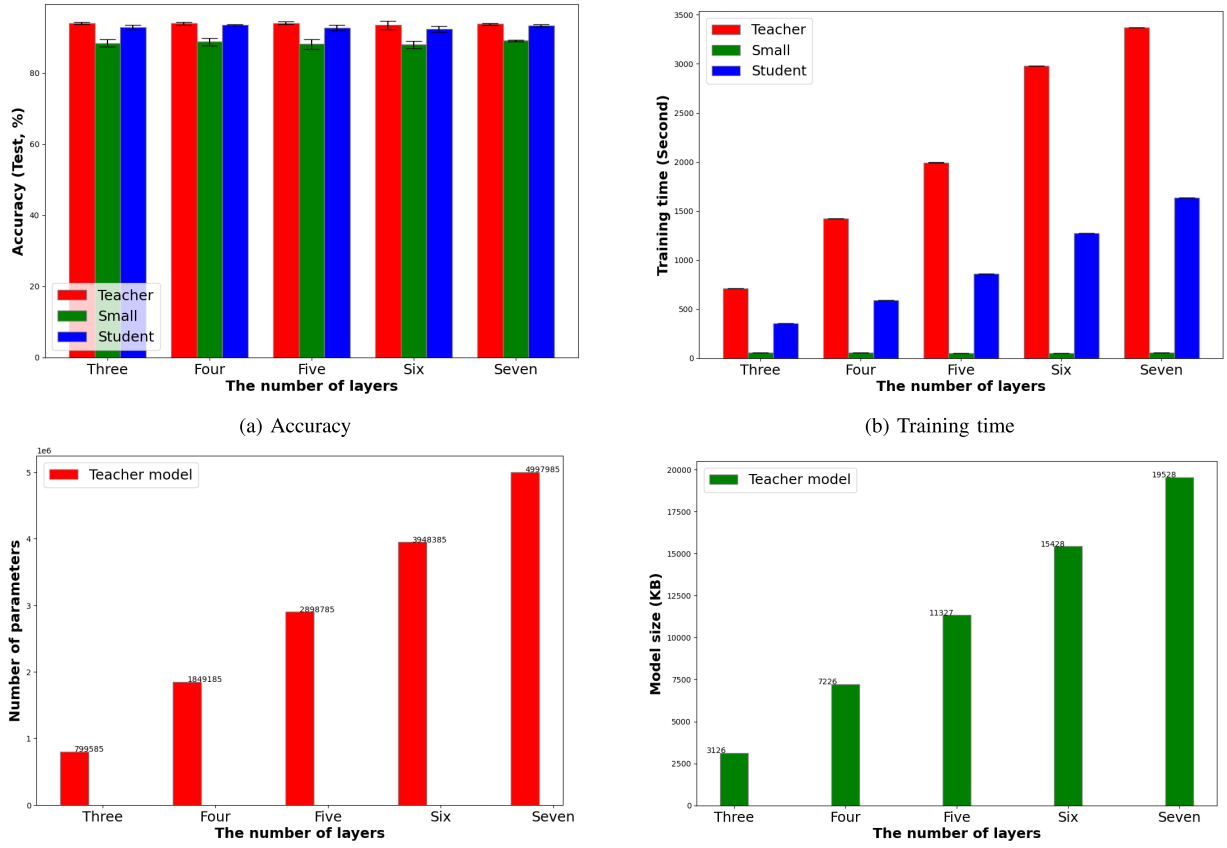
We also trained the student model using soft labels on 6% (1,899 and 1,230 data instances) and 15% (4,794 and 3,073 data instances) of the training data, achieving higher accuracies of 95.67%/, 91.27%, 96.946%, and 91.51%, respectively. The results suggest that training the model with soft labels allows the model to generalize well with less data, as it captures more information than a single hard label. Additionally, the tables provide information on the training times and the number of parameters for each model, with the student models requiring significantly less time and fewer parameters than the teacher models.

Overall, the experiment's results lend support to the claim that soft labels are a valuable approach for training student models with scarce data resources. This is because soft labels contain more information than a single hard label, which enables the model to effectively capture the critical features of the data using fewer samples.

E. EFFECT OF STUDENT MODEL ARCHITECTURE

The student model architecture (e.g., number of layers, types of activation functions, number of neurons in each layer, etc.) is a crucial component in knowledge distillation, as it determines the complexity and capacity of the model that will be used to learn from the teacher model. In this section, we will investigate the effect of different student model architectures on the performance of IoT traffic classification using knowledge distillation. We will compare the performance of different architectures and analyze the results to identify which architecture is most effective for this task. This investigation can help us to better understand the trade-off between model complexity and performance in the context of knowledge distillation for IoT traffic classification. Indeed, by investigating various student model architectures, we can identify architectures that are lightweight and computationally efficient while maintaining acceptable classification performance as t is crucial for deploying IoT traffic classification models on resource-constrained devices.

In this experiment, we consider varying the number of layers in student models while keeping the number of neurons fixed at 400 in each layer. We consider student models with two-layer, three-layer, four-layer, and five-layer configurations.



(a) Accuracy

(b) Training time

(c) Number of parameters

(d) Model size

FIGURE 12. Comparison of classification accuracy, training time, number of parameters, and model size on the CIC-IDS2017 dataset with varying number of layers.

TABLE 10. Soft labels enable the student model to effectively generalize withonly a limited amount of training data. these soft labels are acquired throughtraining on the complete set of training data (Cambridge dataset).

System and training set	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher (100 % of training set)	98.631±0.11	0.986±0.001	0.986±0.001	389.12	463,009	1,813 KB
Student (3 % of training set)	94.249±0.18	0.942±0.001	0.942±0.001	7.88	122,509	481 KB
Student (6 % of training set)	95.67±0.5	0.956±0.001	0.956±0.001	12.28	122,509	481 KB
Student (15 % of training set)	96.946±0.22	0.964±0.002	0.964±0.002	32.2	122,509	481 KB

TABLE 11. Soft labels enable the student model to effectively generalize withonly a limited amount of training data. these soft labels are acquired throughtraining on the complete set of training data (CIC-IDS2017 dataset).

System and training set	Accc	F1	Recall	Training time(S)	# of Parameters	Size
Teacher (100 % of training set)	93.57±0.920	0.935±0.0092	0.935±0.0092	656.63	2,152,785	8,412 KB
Student (3 % of training set)	87.98±1.83	0.879±0.018	0.879±0.018	34.55	66,409	262 KB
Student (6 % of training set)	91.27±0.60	0.912±0.006	0.912±0.006	60.92	66409	262 KB
Student (15 % of training set)	91.51±0.78	0.915±0.007	0.915±0.007	110	66409	262 KB

The tables 12 and 13 show the results for the Cambridge and CIC-IDS2017 datasets, respectively. Based on the results, the accuracy of the student model increases as the number of layers in the architecture increases. The highest accuracy is achieved with the five-layer student model in both datasets; However, the differences in the accuracy between the architectures are relatively small, ranging from 98.3% and 92.6% for the two-layer models to 98.55% and 93.92% for the five-layer models, respectively. Moreover, the number of parameters in the student model increases significantly

with the number of layers. The two-layer model has the fewest parameters (98,009 and 33,209), while the five-layer model has the most parameters (579,209 and 354,009). More parameters generally allow the model to capture more complex patterns in the data, but they also increase the risk of overfitting if not properly regularized. Similar to the number of parameters, the model size increases as the number of layers in the student model increases. The two-layer model has the smallest size (385 KB/132 KB), while the five-layer model has the largest size (2267 KB/1386 KB). Model size can be a consideration, especially in resource-constrained environments such as IoT devices, where smaller models may be preferred.

Based on the results, it can be concluded that increasing the number of layers in the student model improves accuracy and reduces error, but at the cost of increased training time, parameters, and model size. The choice of the student model architecture should consider the trade-off between model performance and resource requirements in the specific context of IoT traffic classification.

F. COMPARISON WITH THE MACHINE LEARNING MODELS

In addition to our evaluation of deep learning models, we conducted a comprehensive comparison of several machine learning models to further enrich our understanding of knowledge distillation and IoT traffic classification performance. This comparison encompassed a diverse set of machine learning algorithms, each with its unique characteristics and suitability for different use cases. The models under consideration include Random Forest, Gaussian Naive Bayes (NB), Bernoulli NB, K-Nearest Neighbors (KNN), and XGBoost.

Machine learning models offer an alternative approach to IoT traffic classification, relying on established algorithms and principles. They are particularly valuable when resource constraints and computational efficiency are critical considerations. The inclusion of these models in our analysis provides a holistic view of the landscape, offering insights into both the distillation of knowledge in neural networks and traditional machine learning paradigms.

In the following, we present the results and analysis of these machine learning models, highlighting their performance across accuracy, F1 score, and recall. This multifaceted evaluation allows us to assess the predictive power of these models, enabling more informed decisions in selecting the most appropriate model for specific IoT traffic classification scenarios.

As it can be seen from Figure. 13a, knowledge distillation showcased exceptional performance, achieving an accuracy, F1 score, and recall of approximately 98.3% on the Cambridge dataset. Random Forest demonstrated competitive accuracy at around 94.9%, albeit with lower F1 scores and recalls compared to knowledge distillation. Gaussian NB, and Bernoulli NB yielded moderate accuracy at approximately 54.3% and 72.9%, respectively, with lower F1 scores and recalls, making them more applicable to simpler IoT contexts. KNN delivered robust performance with high accuracy (around 97.0%) and competitive F1 scores and recalls, establishing it as a versatile choice across diverse IoT applications. XGBoost achieved moderate accuracy (around 84.2%) with competitive F1 scores and recalls, striking a balance between precision and computational efficiency. In conclusion, knowledge distillation is highly effective for IoT traffic classification, while traditional machine learning models like KNN and XGBoost offer robust performance in various IoT scenarios, allowing method selection to align with specific application requirements and numerical performance metrics.

Moreover, to provide a comprehensive comparison, we evaluated the performance of the machine learning models on the CIC-IDS2017 dataset (see Figure. 13b). On the CIC-IDS2017 dataset, knowledge distillation maintained its effectiveness, achieving an impressive accuracy of 92.7%. This result reaffirms the value of knowledge distillation as a robust approach for IoT traffic classification across multiple datasets. Notably, the student model, compressed through knowledge distillation, demonstrated remarkable accuracy and F1 scores, showcasing its potential as a lightweight and efficient classifier. When comparing the machine learning models, Random Forest displayed competitive accuracy, hovering around 56.1%. However, it exhibited relatively lower F1 scores and recalls in contrast to knowledge distillation. Gaussian NB and Bernoulli NB achieved moderate accuracy levels of approximately 29.0% and 55.0%, respectively. KNN emerged as a robust performer with a high accuracy rate of around 93.0%, coupled with competitive F1 scores and recalls, rendering it a versatile choice across a wide range of IoT applications. XGBoost delivered moderate accuracy at approximately 50.5%, while maintaining competitive F1 scores and recalls. This equilibrium in performance establishes XGBoost as an appealing option, particularly in situations where both precision and computational efficiency hold paramount importance.

G. COMPARISON WITH MODEL PRUNING AS ALTERNATIVE

In this subsection, we delve into a critical aspect of our research–evaluating the effectiveness of two distinct compression techniques: knowledge distillation and model pruning. As the demand for efficient and resource-conscious IoT traffic classification models continues to grow, the selection of an optimal compression method becomes paramount. Model pruning, a traditional technique, aims to directly trim the architecture of a neural network by eliminating redundant or less informative components, thereby reducing its size and computational demands [38]. In this comparative analysis, we rigorously assess the performance of these compression techniques in the context of IoT traffic classification.

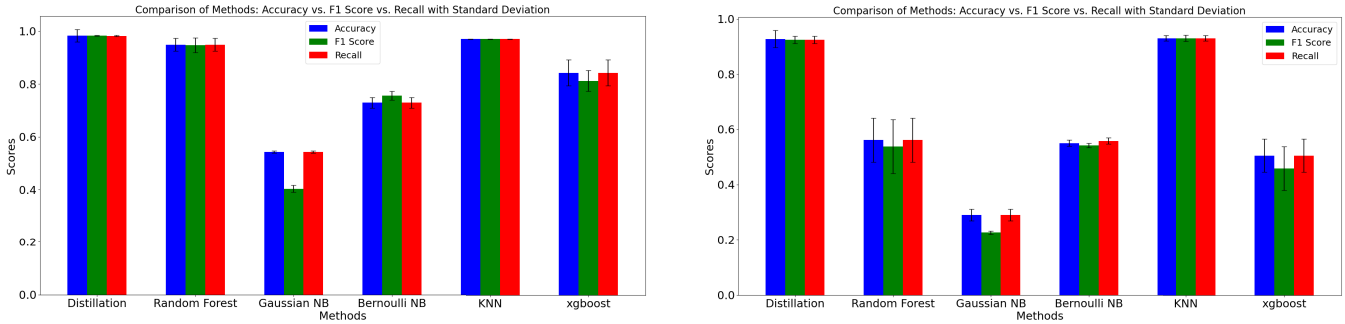
The choice to compare model pruning alongside knowledge distillation is rooted in its direct relevance to the unique challenges of IoT traffic classification. Model pruning, as a method for reducing the size and computational complex-

TABLE 12. Performance of the student model on the cambridge dataset with a varying number of layers.

	Two-layer	Three-layer	Four-layer	Five-layer
ACC	98.3	98.46	98.49	98.55
Error (Std.)	0.181	0.109	0.125	0.067
Training time (second)	1288	1306	1350	1425
# of parameters	98,009	258,409	418,809	579,209
Model size (KB)	385	1012	1639	2267

TABLE 13. Performance of the student model on the CIC-IDS2017 dataset with a varying number of layers.

	Two-layer	Three-layer	Four-layer	Five-layer
ACC	92.61	93.44	93.8	93.92
Error (Std.)	0.82	1.08	0.21	0.29
Training time (second)	755	779	931	942
# of parameters	33,209	193,609	514,409	354,009
Model size (KB)	132	754	2013	1386



(a) Cambridge (b) CIC-IDS2017

FIGURE 13. The performance comparison of knowledge distillation with traditional machine learning methods.

TABLE 14. Comparison of model performance metrics for knowledge distillation and pruning technique (Cambridge dataset).

Technique	Pruning Setting	Accc	F1	Recall	Training time (S)	# of Parameters
Knowledge distillation	N/A	98.31±0.14	0.9833±0.002	0.9829±0.002	1227.7	196,009
Model pruning	(model.linear, amount=0.2) (model.linear3, amount=0.3) (model.linear4, amount=0.3)	88.56±0.58	0.885±0.005	0.885±0.005	1108.4	678,001
Model pruning	(model.linear, amount=0.2) (model.linear3, amount=0.3) (model.linear4, amount=0.3) (model.linear6, amount=0.2)	87.21±0.15	0.872±0.0015	0.872±0.0015	1079.2	908,945
Model pruning	(model.linear, amount=0.4) (model.linear3, amount=0.5) (model.linear4, amount=0.5) (model.linear6, amount=0.3)	86.45±0.46	0.864±0.004	0.864±0.004	1095.5	1,139,889

ity of machine learning models, aligns inherently with the requirements of these resource-constrained settings. By identifying and eliminating redundant or less influential model parameters, model pruning offers an effective means to create more compact and efficient models. This characteristic of model pruning is particularly pertinent to IoT applications, where model size reduction is crucial for efficient deployment and operation on resource-constrained devices. Moreover,

TABLE 15. Comparison of model performance metrics for knowledge distillation and pruning technique (CIC-IDS2017 dataset).

Technique	Pruning Setting	Accc	F1	Recall	Training time(S)	# of Parameters
Knowledge distillation	N/A	92.45±0.13	0.935±0.009	0.935±0.009	709.03	66,409
Model pruning	(model.linear, amount=0.2) (model.linear3, amount=0.3) (model.linear4, amount=0.3)	77.31±0.50	0.773±0.005	0.773±0.005	662.4	644,823
Model pruning	(model.linear, amount=0.2) (model.linear3, amount=0.3) (model.linear4, amount=0.3) (model.linear6, amount=0.2)	77.82±0.38	0.778±0.003	0.778±0.003	691.2	644,823
Model pruning	(model.linear, amount=0.4) (model.linear3, amount=0.5) (model.linear4, amount=0.5) (model.linear6, amount=0.3)	76.84±0.33	0.768±0.003	0.768±0.003	707.5	1,073,534

model pruning's focus on striking a balance between model size reduction and classification accuracy is well-suited to the needs of IoT traffic classification.

Tables 14 and 15 provide the comparison of model performance metrics between knowledge distillation and model pruning techniques using the Cambridge dataset and CIC-IDS2017 dataset, respectively. The model pruning settings involve systematically reducing the size and complexity of neural network architectures by selectively removing a percentage of model parameters from specific layers. In the presented comparison, three different pruning settings were employed, each varying in the choice of layers and the extent of pruning, expressed as the ''amount'' parameter. These settings explore the impact of pruning on the model's performance, with higher ''amount'' values indicating more aggressive pruning. The objective is to provide a comprehensive evaluation of the effectiveness of model pruning as a compression technique.

Considering the Cambridge dataset (Table 14), the accuracy achieved by knowledge distillation is remarkably high at 98.31%. This technique also demonstrates excellent F1 score and recall. Model pruning, even with a relatively aggressive pruning setting (removing 20-50% of parameters), results in a significant drop in accuracy compared to knowledge distillation. Moreover, model pruning, even with different pruning settings, requires lower training times compared to knowledge distillation. The training times range from 1,079.2 seconds to 1,108.4 seconds for different pruning settings. Furthermore, as the model pruning becomes more aggressive (higher ''amount'' values), the number of parameters increases, and it remains substantially larger than the knowledge distillation model.

Similar to the Cambridge dataset, knowledge distillation achieves high accuracy at 92.45% on the CIC-IDS2017 dataset. It also exhibits good F1 score and recall, demonstrating its effectiveness in this dataset (Table 15). Model pruning in this dataset also leads to a notable decrease in accuracy compared to knowledge distillation. The decrease in accuracy is more pronounced here compared to the Cambridge dataset. This suggests that model pruning may not be as effective for this dataset, potentially due to the dataset's complexity.

In Table 15, knowledge distillation exhibits a relatively longer training time, with an average of approximately 709 seconds. However, it is important to note that this training time is not much longer than that observed for the model pruning. Model pruning results in a larger model with 1,073,534 parameters, which is substantially larger than the knowledge distillation.

VI. CONCLUSION

In this study, we investigated the effectiveness of knowledge distillation for IoT traffic classification. Through a comprehensive experimental analysis, we explored the effectiveness of this technique in improving the classification performance of IoT traffic data. Our findings provide valuable insights into the benefits and limitations of knowledge distillation in this domain. Our study provides empirical evidence that knowledge distillation is a powerful technique for improving IoT traffic classification. The results highlight the effectiveness of transferring knowledge from a teacher model to a student model, enabling the student model to achieve superior performance, even in scenarios with limited training data or removed classes. These findings contribute to the advancement of knowledge distillation techniques and their application in IoT traffic classification and other related domains. Future research can further explore optimization strategies and investigate the scalability of knowledge distillation for larger-scale IoT environments.

REFERENCES

[1] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, ''The rise of traffic classification in IoT networks: A survey,'' *J. Netw. Comput. Appl.*, vol. 154, Mar. 2020, Art. no. 102538.

[2] M. Abbasi, A. Shahraki, M. Jalil Piran, and A. Taherkordi, ''Deep reinforcement learning for QoS provisioning at the MAC layer: A survey,'' *Eng. Appl. Artif. Intell.*, vol. 102, Jun. 2021, Art. no. 104234. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197621000816

[3] P. Khandait, N. Hubballi, and B. Mazumdar, "IoTHunter: IoT network traffic classification using device specific keywords," *IET Netw.*, vol. 10, no. 2, pp. 59–75, Mar. 2021.

[4] M. Abbasi, M. Plaza-Hernández, J. Prieto, and J. M. Corchado, "Security in the Internet of Things application layer: Requirements, threats, and solutions," *IEEE Access*, vol. 10, pp. 97197–97216, 2022.

[5] H. Jmila, G. Blanc, M. R. Shahid, and M. Lazrag, "A survey of smart home IoT device classification using machine learning-based network traffic analysis," *IEEE Access*, vol. 10, pp. 97117–97141, 2022.

[6] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "A review on machine learning-based approaches for Internet traffic classification," *Ann. Telecommun.*, vol. 75, nos. 11–12, pp. 673–710, Dec. 2020.

[7] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.

[8] J. Guan, J. Cai, H. Bai, and I. You, "Deep transfer learning-based network traffic classification for scarce dataset in 5G IoT systems," *Int. J. Mach. Learn. Cybern.*, vol. 12, pp. 3351–3365, Nov. 2021.

[9] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artif. Intell. Rev.*, vol. 53, pp. 5113–5155, Feb. 2020.

[10] M. Shafiq, Z. Tian, A. K. Bashir, A. Jolfaei, and X. Yu, "Data mining and machine learning methods for sustainable smart cities traffic classification: A survey," *Sustain. Cities Soc.*, vol. 60, Sep. 2020, Art. no. 102177.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[12] R. Zhao, L. Yang, Y. Wang, Z. Xue, G. Gui, and T. Ohtsuki, "A semi-supervised federated learning scheme via knowledge distillation for intrusion detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2022, pp. 2688–2693.

[13] M. Lu, B. Zhou, Z. Bu, and Y. Zhao, "Lightweight models for traffic classification: A two-step distillation approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2022, pp. 2108–2113.

[14] X. Cheng, Z. Rao, Y. Chen, and Q. Zhang, "Explaining knowledge distillation by quantifying the knowledge," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12925–12935.

[15] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, 2020, pp. 5191–5198.

[16] N. Passalis and A. Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 268–284.

[17] Y. Liu, L. Sheng, J. Shao, J. Yan, S. Xiang, and C. Pan, "Multi-label image classification via knowledge distillation from weakly-supervised detection," in *Proc. ACM Int. Conf. Multimedia*, 2018, pp. 700–708.

[18] W. Wang, W. Hong, F. Wang, and J. Yu, "GAN-knowledge distillation for one-stage object detection," *IEEE Access*, vol. 8, pp. 60719–60727, 2020.

[19] X. Liu, P. He, W. Chen, and J. Gao, "Improving multi-task deep neural networks via knowledge distillation for natural language understanding," 2019, *arXiv:1904.09482*.

[20] P. Chen, S. Liu, H. Zhao, and J. Jia, "Distilling knowledge via knowledge review," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 5008–5017.

[21] Y. Zhang, Y. Xing, Y. Liu, and T. Zhang, "Distillation knowledge-based space–time data prediction on industrial IoT edge devices," *Ad Hoc Netw.*, vol. 137, Dec. 2022, Art. no. 102984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870522001561

[22] H. Jin et al., "Personalized edge intelligence via federated self-knowledge distillation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 2, pp. 567–580, Feb. 2023.

[23] K. Zhang et al., "Compacting deep neural networks for Internet of Things: Methods and applications," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 11935–11959, Aug. 2021.

[24] Z. Wu et al., "Survey of knowledge distillation in federated edge learning," 2023, *arXiv:2301.05849*.

[25] Y. Zhang, G. Gui, and S. Mao, "A lightweight malware traffic classification method based on a broad learning architecture," *IEEE Internet Things J.*, vol. 10, no. 23, pp. 21131–21132, Dec. 2023.

[26] J. Ning et al., "Malware traffic classification using domain adaptation and ladder network for secure Industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17058–17069, Sep. 2022.

[27] O. Barut, Y. Luo, P. Li, and T. Zhang, "R1DIT: Privacy-preserving malware traffic classification with attention-based neural networks," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 2, pp. 2071–2085, Jun. 2023.

[28] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," 2018, *arXiv:1802.05668*.

[29] R. Zhao et al., "A novel traffic classifier with attention mechanism for Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 19, no. 11, pp. 10799–10810, Nov. 2023.

[30] M. S. Sheikh and Y. Peng, "Procedures, criteria, and machine learning techniques for network traffic classification: A survey," *IEEE Access*, vol. 10, pp. 61135–61158, 2022.

[31] R. Kumar, M. Swarnkar, G. Singal, and N. Kumar, "IoT network traffic classification using machine learning algorithms: An experimental analysis," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 989–1008, Jan. 2022.

[32] A. Hameed, J. Violos, and A. Leivadeas, "A deep learning approach for IoT traffic multi-classification in a smart-city scenario," *IEEE Access*, vol. 10, pp. 21193–21210, 2022.

[33] M. Abbasi, A. Taherkordi, and A. Shahraki, "FLITC: A novel federated learning-based method for IoT traffic classification," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Jun. 2022, pp. 206–212.

[34] A. W. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst.*, Jun. 2005, pp. 50–60.

[35] W. Liu, C. Zhu, Z. Ding, H. Zhang, and Q. Liu, "Multiclass imbalanced and concept drift network traffic classification framework based on online active learning," *Eng. Appl. Artif. Intell.*, vol. 117, Jan. 2023, Art. no. 105607.

[36] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, vol. 1, 2018, pp. 108–116.

[37] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," 2016, *arXiv:1612.03928*.

[38] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.