Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX.

Explainable AI for Lightweight Network Traffic Classification Using Depthwise Separable Convolutions

Mustafa Ghaleb*, Mosab Hamdan‡, Abdulaziz Y. Barnawi†, Muhammad Gambo†, Abubakar Danasabe†, Saheed Bello†, Aliyu Habib†

1* IRC for Intelligent Secure Systems, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
2† Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
3‡ School of Computing, National College of Ireland, Dublin 01, Ireland

CORRESPONDING AUTHORS: Abdulaziz Barnawi (e-mail: barnawi@ kfupm.edu.sa), Mosab Hamdan (e-mail: mosab.hamdan@ieee.org).

ABSTRACT With the rapid growth of internet usage and the increasing number of connected devices, there is a critical need for advanced Network Traffic Classification (NTC) solutions to ensure optimal performance and robust security. Traditional NTC methods, such as port-based analysis and deep packet inspection, struggle to cope with modern network complexities, particularly dynamic port allocation and encrypted traffic. Recently, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been employed to develop classification models to accomplish this task. Existing models for NTC often require significant computational resources due to their large number of parameters, leading to slower inference times and higher memory consumption. To overcome these limitations, we introduce a lightweight NTC model based on Depthwise Separable Convolutions and compare its performance against CNN, RNN, and state-of-the-art models. In terms of computational efficiency, our proposed lightweight CNN exhibits a markedly reduced computational footprint. It utilizes only 30,611 parameters and 0.627 MFLOPS, achieving inference times of 1.49 seconds on the CPU and 0.43 seconds on the GPU. This corresponds to roughly 4× fewer FLOPS than the RNN baseline and 16× fewer than the CNN baseline, while also offering an ultracompact design compared to state-of-the-art models. Such efficiency makes it exceptionally well-suited for real-time applications in resource-constrained environments. In addition, we have integrated eXplainable Artificial Intelligence techniques, specifically LIME and SHAP, to provide valuable insights into model predictions. LIME and SHAP help interpret the contribution of each feature in decision-making, enhancing the transparency and trust in the model's predictions, without compromising its lightweight nature. To support reproducibility and foster collaborative development, all associated code and resources have been made publicly available.

INDEX TERMS Network Traffic Classification, Deep Learning, Depthwise Separable Convolution, eXplainable Artificial Intelligence, LIME, SHAP

I. INTRODUCTION

With the rapid advancement of technology and the proliferation of internet-enabled devices, such as smartphones and devices connected to the Internet of Things (IoT), the volume of network traffic has increased dramatically [1]. The escalation in traffic, characterized by variations in confidentiality and value, poses substantial challenges for network management, especially concerning security and performance optimization. Network Traffic Classification (NTC) [2] has become a critical tool in this landscape, allowing network operators to prioritize traffic, enforce security measures, and ensure high Quality-of-Service (QoS) delivery [3], [4]. According to recent reports by Ericsson Mobility [5], global mobile network traffic is projected to reach hundreds of exabytes per month by 2028, underscoring the need for advanced traffic classification methods. In response to this growing demand, Machine Learning (ML) and Deep Learning (DL) have emerged as vital technologies for automating and enhancing NTC processes. As networks progressively increase in scale and complexity, the utilization of these advanced technologies will be essential for sustaining efficient, secure, and high-performing network infrastructures. However, while ML and DL approaches have been extensively explored to automate and enhance NTC, their real-world applicability, particularly in resource-constrained environments, remains limited [6], [7], [9], [10],

[15], [19]. Many existing models [22]–[25] rely on large datasets and resource-intensive architectures, making them unsuitable for deployment on edge devices due to their constrained processing capabilities, memory, and power resources. Consequently, conventional DL models are deemed impractical for networks necessitating rapid response times and minimal processing overhead [26]–[28]. Despite advancements in NTC, applying these models in real-world settings still requires addressing key challenges such as effectiveness, deployability, trustworthiness, robustness, and adaptability [20], [21]. Consequently, there is a growing need for lightweight NTC solutions that balance strong performance with efficient resource usage.

On the other hand, DL techniques have successfully tackled various challenges in modern network traffic by automating feature extraction and handling large datasets. However, their lack of interpretability remains a critical concern [19], [21], [59], [60]. NTC plays a pivotal role in network management and security, especially as mobile traffic continues to grow. The rise of smartphones and mobile applications has transformed the traffic landscape, leading to increased diversity and frequency of updates. Although DL techniques outperform traditional ML models in terms of accuracy, their ”black-box” nature raises concerns about the lack of transparency, particularly in critical scenarios such as network security. Without a clear understanding of how DL models make decisions, there is a risk of reduced trust, increased errors, and vulnerability to adversarial attacks [19]. To mitigate these concerns, eXplainable Artificial Intelligence (XAI) [19], [29], [30] has emerged as a promising solution, offering methods to enhance the interpretability of DL models. By linking classification outcomes to model structures, XAI provides insights that improve both trustworthiness and reliability. While XAI is still relatively new in the field of network traffic analysis, its importance is rapidly growing.

In order to overcome these challenges, we propose a novel lightweight NTC model based on Depthwise Separable Convolutions (DSC), designed to efficiently process large volumes of data while minimizing computational resource requirements. The lightweight model demonstrates significant improvements in efficiency, with fewer parameters (30,611) and a reduced inference time on the CPU and GPU of 1.49, 0.43 seconds, respectively, while achieving an impressive overall test accuracy of 99.96%. We evaluate our model using a dataset provided by Universidad Del Cauca, consisting of 3,577,296 IP flow instances collected over six days [56]. Additionally, to further validate our model’s robustness, we evaluate it using the UTMobileNetTraffic2021 dataset [62]. Moreover, we integrate explainable artificial intelligence (XAI) methodologies, specifically Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP), to provide comprehensive insights into the model’s predictive behavior. This approach enhances transparency and trustworthiness in network security environments.

The contributions of this study are summarized as follows:

- We propose a novel lightweight NTC model based on DSCs, designed to efficiently handle large datasets with reduced computational overhead.
- We assess the performance of our model by employing metrics including the number of parameters, floating-point operations per second (FLOPS), inference time on both CPU and GPU, accuracy, recall, precision, and F1-score. These evaluations are conducted in comparison with CNN, RNN baselines, and current state-of-the-art models.
- We incorporate XAI techniques, including LIME and SHAP, to interpret the model’s predictions and enhance transparency and trustworthiness in practical network security applications.

The rest of this paper is organized as follows: Section 2 provides a comprehensive background and an extensive review of pertinent literature. The lightweight NTC model is detailed in Section 3. Section 4 presents and conducts an in-depth analysis of the experimental outcomes. Lastly, Section 5 summarizes the study and describes prospective avenues for future research exploration.

II. BACKGROUND AND RELATED WORKS

This section covers key concepts and challenges of trustworthiness and interpretability in AI models for NTC. It also reviews recent advancements in DL-based NTC models, highlighting their impact on scalability, accuracy, and interpretability.

A. XAI for Trustworthiness and Transparency in NTC

The proliferation of extensive datasets across diverse domains has rendered AI an indispensable instrument for enhancing decision-making processes, reducing expenses, and managing risks effectively. However, the implementation of sophisticated AI algorithms, particularly DL models, has led to the creation of ”black-box” models, raising concerns about transparency and trustworthiness. The aforementioned challenges have compelled both governmental entities and the academic community to concentrate on XAI techniques as a means to address issues associated with transparency, causality, fairness, and safety [19], [31].

XAI aims to enhance the interpretability and trustworthiness of AI models by analyzing their behavior to assess robustness, evaluate resilience to data perturbations, and ensure compliance with legal and safety standards. One important aspect is calibration, where the confidence of a model’s probabilistic outputs should reflect the reliability of its predictions. Recent research has introduced evaluation metrics for model calibration [32]–[34] and learning techniques, such as Dirichlet-based methods, label smoothing, and focal loss to improve it [32], [35], [36].

Interpretability in XAI methods is categorized into four key areas: explaining black-box models, creating white-box models, promoting fairness, and analyzing model prediction sensitivity [19]. These methods can be either model-specific or model-agnostic and operate on different scales—local (explaining specific instances) or global (explaining the entire model). Various approaches, such as Occlusion Analysis [37], Shapley-based methods like SHAP [38], local surrogate models like LIME [39], Integrated Gradients [40], and Layerwise Relevance Propagation (LRP) [41], aim to leverages the layered structure of the neural network, operating iteratively to provide post-hoc explanations for ML/DL model outcomes

B. Related Works

DL-based NTC has gained considerable attention in recent years, primarily due to its capability to manage the growing complexity and heterogeneity of network traffic patterns. It enables effecient feature extraction and achieving improved classification accuracy. This section reviews the state of the art in network traffic classification methods, starting with a discussion of foundational methods followed by discsussion of deep learning methods. We then present more rececent techniques and specific architectures in NTS. The review concludes with a discussion of recent advancements in XAI for traffic classification.

Wang et al. [6] studied the application of DL for network traffic classification. They introduced a framework using Artificial Neural Networks (ANN) and Stacked Auto-Encoders (SAE) to classify traffic into encrypted and non-encrypted protocols. The model highlighted the importance of the first few bytes of traffic data, and automated feature learning replaced manual feature engineering. While it showed high classification accuracy on a private dataset, its computational complexity made it unsuitable for real-time or resource-constrained environments. Dange et al. [24] proposed a hybrid architecture combining convolutional and recurrent neural networks with an autoencoder, enabling autonomous extraction of spatial and temporal features from raw traffic data. This eliminated manual feature engineering and enhanced feature expressiveness but increased computational complexity, limiting suitability for resource-constrained environments. Similarly, Ren et al. [25] introduced Tree-RNN, which partitions large classification tasks using a tree structure and multiple specialized classifiers. Tree-RNN achieved higher accuracy and faster training but faced scalability challenges due to its complexity. In contrast, our lightweight DSC-based CNN reduces computational complexity while maintaining high accuracy. Unlike these models, it is optimized for resource-limited environments like IoT devices, offering efficiency without sacrificing performance.

Several approaches in NTC have used DL models, particularly CNNs, to exploit spatial and temporal features in network traffic data. Huang et al. [13] proposed a multi-task CNN architecture for classifying malicious and benign traffic using the CTU-13 and ISCX datasets, achieving high accuracy. Tong et al. [14] combined CNN1D and supervised learning to classify Google's QUIC protocol-based traffic, demonstrating effectiveness on a private dataset. Bu et al. [16] suggested a deep parallel Network-In-Network (NIN) model, while Liu et al. [17] introduced Flow Sequence Network (FS-Net), a framework with an encoder-decoder structure. Both outperformed traditional CNN models on the ISCX dataset and a private dataset collected by [18].

Moreover, Anwar et al. [42] proposed an attention-based CNN using the QUIC dataset to predict bandwidth requirements and network flow duration. Their analysis showed that the attention mechanism enhances accuracy by focusing on relevant features while reducing reliance on irrelevant ones. However, attention-based CNNs have drawbacks, including higher computational costs, slower inference, and greater training complexity, making them unsuitable for real-time or resource-constrained environments. On the other hand, Hussain et al. [43] described a distributed computing paradigm for NTC, that distributes the ML model across nodes, each handling a subset of the training dataset. Using the UNB ISCX VPN and non-VPN datasets [44], the study used data parallelism with the Single Program Multiple Data (SPMD) model. This approach is effective for IoT systems, with training in the cloud and minimal inference on edge devices using pre-trained DNN models. Algorithms like LSTM, ConvGRU, CNN, ConvLSTM, and XGBoost were evaluated, with CNN achieving a 22.78% speedup as nodes increased from two to four. However, despite reduced training time, recall, precision, and F1 score declined as nodes increased.

Furthermore, Wang et al. [7] used a one-dimensional CNN (CNN1D) to classify both VPN and non-VPN traffic. They argued that CNN1D is well-suited for sequential data, such as network traffic, compared to CNN2D and traditional models like C4.5. Using the ISCX dataset, their experimental results showed that CNN1D outperformed both CNN2D and C4.5, especially when using 784 bytes of data. However, CNN1D is less suitable for real-time applications in resource-constrained environments. Lotfollahi et al. [8] introduced a deep packet framework that integrates CNN1D with a stacked autoencoder (SAE), achieving high accuracy on the ISCX dataset for application identification and service group classification. Aceto et al. [9], [10] compared CNN1D for mobile traffic classification, where it outperformed other models on a private dataset. The same authors [11], [12] further improved traffic classification with the MIMETIC framework, a multimodal DL solution integrating CNNs and GRUs, yielding strong results on Android and iOS datasets.

Lopez et al. [15] proposed a method that integrates spatial and temporal dimensions of network traffic using CNNs for spatial correlations and LSTMs for temporal correlations. This approach extracts six features from the first 20 packets of traffic: payload bytes, source port, destination port, TCP window size, packet direction, and interarrival time. Evaluated on the RedIRIS dataset, the CNN-LSTM model achieved the highest accuracy among tested models. However, the integration of CNN with LSTM increases computational costs, limiting its suitability for resource-constrained environments.

Wang et al. [59] proposed Ulfar, a multi-scale feature attention method using CNNs for network traffic classification, focusing on format-related bytes at fixed offsets in packets. Ulfar's multi-scale n-gram feature extraction and attention mechanisms enable adaptive feature fusion for robust classification. Evaluated on RealTrace-I [57] and RealTrace-II [58] datasets, Ulfar outperformed state-of-the-art methods in classification accuracy, demonstrating its effectiveness in handling diverse traffic patterns. However, its complexity introduces challenges for real-time applications. Furthermore, Wei et al. [60] introduced ABL-TC, a lightweight NTC model based on an attention-based LSTM. This model processes raw traffic data by focusing on essential features and discarding redundant ones. ABL-TC achieved over 99.6% average recall, precision, and F1 score on real-world datasets, with reduced computational and memory requirements, making it suitable for resource-constrained environments. Compared to ABL-TC, our model is even lighter in terms of parameters and FLOPS.

Nascita et al. [19] applied XAI techniques to enhance the interpretability and trustworthiness of multimodal deep learning architectures for mobile traffic classification. Their Mimetic-Enhanced framework utilized multiple data modalities, focusing on payload and packet-sequence analysis. Using SHAP-based methods, the study offered global insights into model behavior, improving performance and transparency. Evaluations on a public dataset showed significant accuracy gains and increased trustworthiness, aided by model calibration techniques such as focal loss and label smoothing, ensuring reliable predictions even with encrypted traffic.

The approaches discussed above have significantly advanced network traffic classification (NTC). However, the growing complexity of modern network environments, especially in IoT edge networks, exposes critical vulnerabilities. Traditional deep learning models, though effective, are resource-intensive and computationally expensive, making them unsuitable for real-time use on constrained IoT devices. In addition, their lack of transparency limits adoption in industries requiring explainable and trustworthy AI solutions. These challenges requires the adoption of a lightweight, efficient, and interpretable solution that can meet the unique demands of IoT edge networks while maintaining high accuracy and performance.

To bridge this gap, our proposed work introduces a novel framework for lightweight NTC using DSCs combined with XAI techniques such as LIME and SHAP. This approach reduces computational overhead without sacrificing accuracy, making it ideal for real-time deployment in resource-constrained IoT edge environments. Furthermore, the integration of XAI enhances the transparency and interpretability of the model, allowing network administrators and security professionals to better understand the decision-making process, fostering trust, and ensuring compliance with industry standards. By addressing these challenges, our methodology offers a comprehensive solution that balances efficiency, accuracy, and explainability, positioning it as a practical and innovative contribution to NTC for IoT edge networks.

III. PROPOSED LIGHTWEIGHT NETWORK TRAFFIC CLASSIFCATION MODEL

This study introduces a lightweight CNN architecture specifically designed for NTC. While CNNs are widely known for their powerful feature extraction capabilities in applications like object detection and image classification [45], their computational demand can pose challenges in resource-constrained environments, such as real-time traffic classification. To address this, we leverage separable convolution, introduced by Sifre and Mallat [46], to minimize the model's parameters while preserving high performance.

Separable convolution splits the convolution operation into two stages: a depthwise convolution and a 1×1 pointwise convolution. This approach has been successfully applied in models such as Xception-net [48], Mobilenets [47], EfficientNet [49], and ShuffleNet [50], which achieve an optimal trade-off between computational efficiency and accuracy. By utilizing this strategy, we achieve significant reductions in computational cost without sacrificing the model's ability to extract critical features from network traffic data.

In our design, the DSC results in a reduction of trainable parameters and an improvement in inference speed, which is critical for low-latency traffic classification tasks. For instance, in a one-dimensional convolution, the input feature maps of size $D_{\mathrm{F}} \times 1 \times M$ (where D_F denotes the length of the input, and M indicates the number of input channels) are transformed into output feature maps of size $D_{\mathrm{F}} \times 1 \times N$, where N represents the number of output channels. The computational cost of this operation is:

$$D_{\mathrm{F}} \cdot N \cdot M \cdot D_{\mathrm{K}} \tag{1}$$

In contrast, the DSC first applies a depthwise convolution with a cost of:

$$D_{\mathrm{F}} \cdot M \cdot D_{\mathrm{K}} \tag{2}$$

Then, a pointwise convolution follows, which combines the feature maps with a cost of:

$$D_F \times N \times M \tag{3}$$

Thus, the total reduction in computational cost can be expressed as:

$$\frac{1}{N} + \frac{1}{D_K} \tag{4}$$

Given that N typically ranges from 32 to 1024, and D_K for one-dimensional convolution lies between 9 and 27, the reduction in computational cost can reach up to 85%–95%,

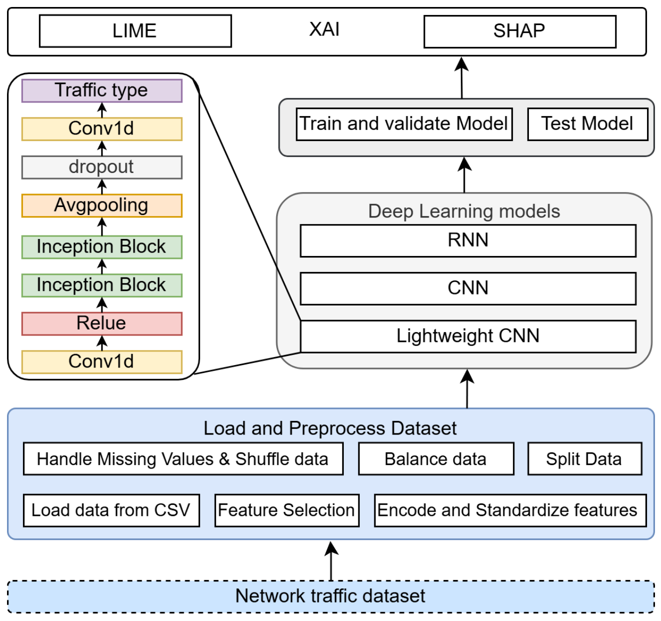


FIGURE 1: Methodology for NTC Using a Separable Convolution Network with XAI

making it highly suitable for real-time NTC tasks [47], [48], [51].

A. Architecture of the Proposed Lightweight CNN

The methodology illustrated in the Figure 1 outlines the process for classifying network traffic using a Separable Convolution Network with XAI. The process begins with loading and preprocessing the dataset, where raw network traffic data is cleaned, encoded, and standardized. Furthermore, handling missing values and shuffling the data processes are performed. The preprocessed data is then fed into various DL models, including RNN, CNN, and a lightweight CNN model, which is selected for its balance between accuracy and computational efficiency.

The lightweight CNN architecture (Separable Convolution Network) consists of a Conv1d layer followed by ReLU activation. Two Inception blocks, as elaborated in the subsequent subsection, are employed to capture multi-scale features, helping the model better understand the network traffic patterns. Subsequent to feature extraction, the model employs average pooling and dropout techniques to reduce dimensionality and mitigate the risk of overfitting. The final output is passed through a classifier that predicts the traffic type.

Following the model design, the methodology proceeds with training, validating, and testing the model on the preprocessed dataset. To enhance the interpretability of the results, Explainable XAI techniques, such as LIME and SHAP, are employed to clarify the predictions of the model. LIME provides local explanations by identifying key features in individual predictions, while SHAP offers a global understanding of feature importance, making the model's decisions transparent and easier to interpret for network analysts. Our design decouples prediction from explanation: the core model, optimized with DSCs, ensures low-latency predictions, while XAI methods (LIME and SHAP) are invoked on-demand, adding no overhead to the critical prediction path.

B. Inception Block

The output from the preceding layer is fed into an Inception Block designed for multi-scale feature extraction. In this updated design, three parallel DSCs are applied with kernel sizes of 11, 19, and 27, respectively, each using a stride of 1 and appropriate padding (5, 9, and 13) to preserve spatial dimensions. Instead of batch normalization, we applied group normalization—with 8 groups—to normalize the activations across multiple channels simultaneously [54], [55], followed by a ReLU activation. The outputs from these three branches are summed element-wise. In addition, if reduction is enabled, a parallel 1×1 convolution is applied to the input using a stride equal to the pooling size (specified as a parameter), followed by group normalization, and its output is added to the max-pooled sum of the three branches (with the max-pooling performed using the given pooling size and padding 1). Moreover, all channel dimensions in the block are dynamically adjusted by scaling the base channel counts with a width multiplier. For example, with a width multiplier of 0.75, the base channel counts of 32, 64, and 128 are reduced to 24, 48, and 96 respectively. This reduction results in a lighter model with significantly fewer parameters and lower FLOPs, thereby improving computational efficiency without sacrificing performance. Such efficiency gains are particularly beneficial for real-time applications in resource-constrained environments. The modification of the Inception Block to include group normalization is based on prior work [52].

C. Group Normalization Technique

Batch Normalization (BatchNorm) [54] and Group Normalization (GroupNorm) [55] are two normalization techniques commonly used in DL to improve the stability and convergence of neural networks.

BatchNorm is a normalization method that normalizes the activations of a layer for each mini-batch of data during training. It subtracts the mean and divides it by the variance of the activations within a mini-batch. This helps to ensure that the activations have zero mean and unit variance, which can improve the stability of the network and speed up convergence. BatchNorm also has the added benefit of serving as a regularizer, helping to reduce overfitting.

GroupNorm, on the other hand, normalizes the activations within a layer by dividing the channels into groups and normalizing within each group. Unlike BatchNorm, GroupNorm does not normalize the activations across all channels in a layer, which can result in a more computationally efficient normalization method. GroupNorm has been found to work well in practice for small batch sizes, where BatchNorm

Algorithm 1: Depthwise Separable Convolution Network with XAI for NTC

```
Input: Dataset $\mathcal{D}$, Number of classes $C$, Pool size $P$, Epochs $N$, Learning rate $\eta$
Output: Trained model, Evaluation Metrics, FLOPs, Inference Times, LIME and SHAP explanations
1  Data Preprocessing
2  Define depthwise separable Conv1 with width multiplier: (input size, 32 out);
3  Define Inception blocks $B1$ and $B2$:
4    - $B1$: input = adjust$(32, w)$, output = adjust$(64, w)$ using GroupNorm with 8 groups and ReLU
5    - $B2$: input = adjust$(64, w)$, output = adjust$(128, w)$ using GroupNorm with 8 groups and ReLU;
6  Add adaptive pooling, dropout, and define classifier (input = adjust$(128, w)$, output = $C$) with log softmax activation;
7  Build the model by stacking Conv1, $B1$, $B2$, pooling, dropout, and classifier;
8  for $epoch = 1$ to $N$ do
9      for each batch $(x, y)$ from training set do
10         Perform forward pass: $output \leftarrow model(x)$;
11         Compute loss: $\mathcal{L} \leftarrow$ CrossEntropyLoss$(output, y)$;
12         Backpropagate and update model weights using optimizer;
13     Compute training and validation accuracy;
14     if Validation accuracy improves then
15         Save current model as checkpoint;
16 for each batch $(x, y)$ from test set do
17     Perform forward pass and compute predictions;
18     Update confusion matrix;
19 Compute overall test accuracy;
20 Generate LIME and SHAP for model interpretability;
21 Compute number of FLOPs and inference time;
22 Report overall test accuracy, confusion matrix, FLOPs, and XAI visualizations (LIME and SHAP);
```

may struggle with poor mean and variance estimates. In the modified inception block, they have used GroupNorm technique.

D. Algorithm: Separable Convolution Network with XAI

The detailed steps of the Separable Convolution Network with XAI are summarized in Algorithm 1. The approach begins with data preprocessing, which includes converting IP addresses into numeric values and removing unnecessary columns such as flow IDs and timestamps. The dataset is filtered to retain the top 15 protocol classes, and each class is downsampled to ensure a balanced representation of 15,000 instances per class.

Once the data is prepared, it is split into training, validation, and testing sets. The numerical features are standardized using the StandardScaler, and the target classes (protocol names) are label-encoded to prepare the data for classification.

At its core, the model employs a DSC-based CNN network enhanced with a width multiplier that scales channel dimensions dynamically. The initial convolutional layer (`Conv1`) outputs adjusted 32 channels, which feed into two Inception blocks. Each Inception block applies DSCs using varying kernel sizes (11, 19, and 27) to capture multi-scale features, and incorporates Group Normalization (with 8 groups) and ReLU activation for stability. This is followed by adaptive average pooling, dropout, and a final classifier with log softmax activation. The model is trained using CrossEntropy loss and the Adam optimizer over 27 epochs, with checkpointing based on validation accuracy improvements.

Upon completing the training phase, the model is evaluated using previously unseen data, and a confusion matrix is constructed to depict the classification performance across various protocol categories. Both LIME and SHAP methodologies are employed to elucidate the model's decision-making processes. LIME furnishes local explanations for specific predictions by altering input features, whereas SHAP calculates global feature importance scores, thereby providing insight into each feature's contribution to the model's predictions. Finally, the model's computational efficiency is assessed through FLOP count and inference times on both CPU and GPU, confirming its suitability for real-time applications in resource-constrained environments.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

This section provides a comprehensive discussion of the results. Additionally, we utilize XAI techniques (SHAP [38] and LIME [39]) to provide interpretability of the models' decisions. The performance of the proposed method is thoroughly analyzed, followed by a comparative evaluation against relevant existing methods. Experiments were conducted on standardized hardware to ensure consistency and reliability[1].

A. Datasets and Preprocessing

In this subsection, we present the datasets used in this study, along with the preprocessing techniques applied to ensure a balanced and well-structured dataset for model training and evaluation.

[1]Experiments conducted on a Linux-based cluster (SLURM-managed) with Intel® Core™ i9-14900K CPUs, 188 GiB RAM, and NVIDIA GeForce RTX 4090 GPUs (24 GB VRAM, CUDA 12.4). Jobs limited to 16 CPU cores, 160 GiB RAM, and 1 GPU each.

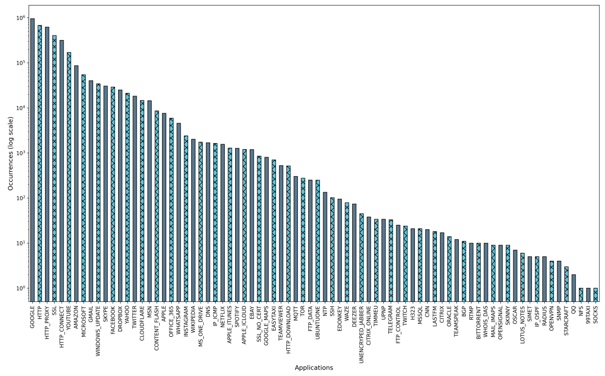


FIGURE 2: Class Distribution Before Preprocessing in Unicauca-Dataset

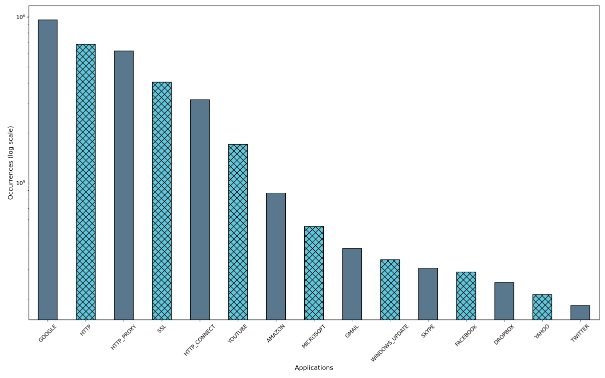


FIGURE 3: Top 15 Most Occurring Classes in Unicauca-Dataset

1) Unicauca-Dataset

The Unicauca-dataset discussed herein is derived from preprocessing operations applied to its former versions, as described in [56], and is accessible for download in [61]. The dataset utilized in this research was obtained from Universidad Del Cauca, Popayán, Colombia, and comprises 3,577,296 instances of network traffic data collected across six days, encompassing both morning and afternoon sessions. This dataset encompasses features such as source and destination IP addresses, IP flows, ports, interarrival times, and Layer 7 protocols. Initially, the dataset contained 86 features, but after removing 13 features with null values, 73 unique features remained. The distribution of all the classes is shown in Figure 2, illustrating the diversity and imbalance in the dataset. From these classes, the 15 most frequently occurring ones were selected, as shown in Figure 3.

To address the high diversity and imbalance in class occurrences, preprocessing techniques were applied, including downsampling, to ensure a balanced dataset with 15,000 instances per class. This balanced dataset focuses on 15 distinct traffic classes, enabling effective model training without over-representing certain classes. The dataset was partitioned into 90% for training and 10% for testing. Furthermore, the training dataset was subdivided such that 90% was allocated for actual training and 10% was reserved for validation purposes.

2) UTMobileNetTraffic2021 Dataset

The UTMobileNetTraffic2021 dataset [62], developed at the University of Texas at Austin, is focused on the analysis of contemporary mobile traffic through the application of ML models to classify behaviors specific to applications. Encompassing over 29 hours of encrypted traffic derived from 14 widely used mobile applications, the dataset provides comprehensive insights into both application-level and activity-level dynamics, while intentionally excluding direct attack scenarios. This meticulous construction makes it an invaluable resource for researchers aiming to evaluate and enhance machine learning techniques in realistic mobile network environments.

In our data preprocessing pipeline, we begin by loading the dataset and addressing missing values in the numeric columns by imputing them with the respective column means. The target variable ('app') is subsequently label-encoded to convert it into a numerical format, while only the numeric features are retained and standardized using the StandardScaler to ensure consistency. Notably, the initial class distribution exhibited significant imbalance, with the minimum count at 201 instances (e.g., for "google-drive") and the maximum at 2029 instances (e.g., for "pandora"). To rectify this imbalance and achieve a uniform representation across classes, SMOTE is employed, thereby providing a robust and balanced foundation for effective model training and evaluation. Furthermore, to ensure reproducibility, we have made the datasets and all preprocessing steps available in a Python notebook[2].

B. Performance Metrics

In traffic classification, True Positive (TP) is the correct identification of a traffic class, False Positive (FP) is an incorrect identification of a non-positive class as positive, True Negative (TN) is the correct rejection of a traffic class, and False Negative (FN) is failing to identify a positive class correctly.

We evaluated our model using standard metrics: accuracy, precision, recall, and F1-score. These metrics were derived from the confusion matrix, enabling a detailed and insightful analysis of classification performance across various classes.

C. Results and Analysis

In this subsection, we assess the performance of five models—the CNN and RNN baselines, Ulfar's multi-scale feature attention model [59], the ABL-TC LSTM-based design [60], and the proposed lightweight DSC-based CNN. Our evaluation focuses on essential performance metrics, namely, accuracy, recall, precision, and F1-score, which

[2]https://github.com/Mustafa2009/Lightweight_CNN

collectively offer a thorough perspective on the classification effectiveness and robustness of each model.

The results in Table 3 clearly show that the proposed lightweight CNN model consistently outperforms CNN, RNN baselines, and ABL-TC in all key evaluation metrics. The proposed lightweight CNN model achieves an outstanding accuracy of 99.96%, which is comparable to Ulfar's 99.97% and notably outperforms other models. In contrast, the traditional CNN baseline records an accuracy of 99.90%, while the RNN lags behind with 98.53%. Although the numerical differences in accuracy might seem minimal, even slight improvements can have a significant impact in real-world network environments, where every fraction of a percent in classification performance is crucial.

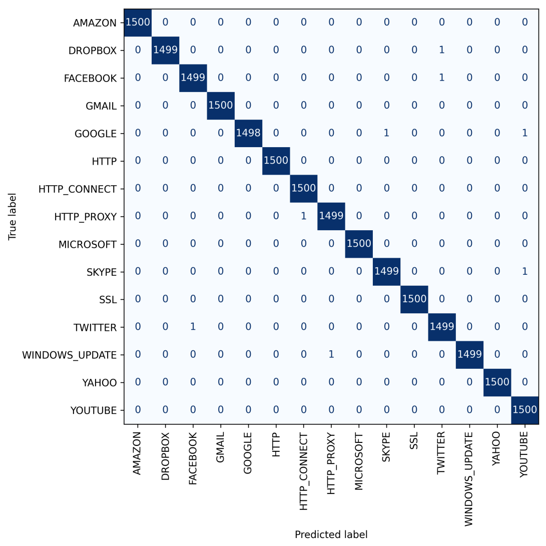
Moreover, when evaluating across all models, the proposed lightweight CNN model stands out by achieving a balanced performance with recall, precision, and F1-scores of 99.96%. In comparison, the traditional CNN baseline achieves an accuracy of 99.90% (with precision at 99.87%, recall at 99.89%, and an F1-score of 99.88%), while the RNN model lags behind with 98.53% across these metrics. Notably, the Ulfar model attains the highest performance with all metrics at 99.97%, and the ABL-TC model follows closely with 99.93%. Although these numerical differences may seem marginal, they are critical in high-stakes network traffic classification where even a 0.03–0.44% improvement can significantly reduce misclassifications.

The confusion matrix, shown in Figure 4, provides a detailed breakdown of the classification accuracy across all traffic classes. Each diagonal entry represents correctly classified instances, while off-diagonal entries indicate misclassifications. Notably, the proposed lightweight CNN model exhibits near-perfect performance with minimal errors: for instance, the '$GOOGLE$' class shows only 2 misclassifications out of 1,500 samples, while '$GMAIL$', '$HTTP\_CONNECT$', '$SKYBE$' and '$WINDOWS\_UPDATE$' each have just a single error. These results underscore the model's strong discriminative power and robustness, confirming that even at the per-class level, the classification performance remains exceptional. Overall, the combination of high global accuracy and near-perfect class-wise performance demonstrates that our model is well-suited for real-world, resource-constrained network traffic classification tasks.

Figure 5 presents the loss plot curves over 27 training epochs. The persistent decrease in training loss indicates effective learning by the model. The validation loss closely parallels the training loss, which implies a good generalization of the model without overfitting. Consequently, the training and validation accuracy plots, as illustrated in Figure 6, demonstrate a consistent improvement over time.

D. Performance Comparison with state-of-the-art Models

To further highlight the efficiency of the proposed lightweight CNN model based on DSCs, we compared its performance with the state-of-the-art models using Unicauca-dataset.



True label \ Predicted label	AMAZON	DROPBOX	FACEBOOK	GMAIL	GOOGLE	HTTP	HTTP_CONNECT	HTTP_PROXY	MICROSOFT	SKYPE	SSL	TWITTER	WINDOWS_UPDATE	YAHOO	YOUTUBE
AMAZON	1500	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DROPBOX	0	1499	0	0	0	0	0	0	0	0	0	1	0	0	0
FACEBOOK	0	0	1499	0	0	0	0	0	0	0	0	1	0	0	0
GMAIL	0	0	0	1500	0	0	0	0	0	0	0	0	0	0	0
GOOGLE	0	0	0	0	1498	0	0	0	0	1	0	0	0	0	1
HTTP	0	0	0	0	0	1500	0	0	0	0	0	0	0	0	0
HTTP_CONNECT	0	0	0	0	0	0	1500	0	0	0	0	0	0	0	0
HTTP_PROXY	0	0	0	0	0	0	1	1499	0	0	0	0	0	0	0
MICROSOFT	0	0	0	0	0	0	0	0	1500	0	0	0	0	0	0
SKYPE	0	0	0	0	0	0	0	0	0	1499	0	0	0	0	1
SSL	0	0	0	0	0	0	0	0	0	0	1500	0	0	0	0
TWITTER	0	0	1	0	0	0	0	0	0	0	0	1499	0	0	0
WINDOWS_UPDATE	0	0	0	0	0	0	0	1	0	0	0	0	1499	0	0
YAHOO	0	0	0	0	0	0	0	0	0	0	0	0	0	1500	0
YOUTUBE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1500

FIGURE 4: Confusion Matrix of the Proposed Lightweight CNN Model using Unicauca-Dataset

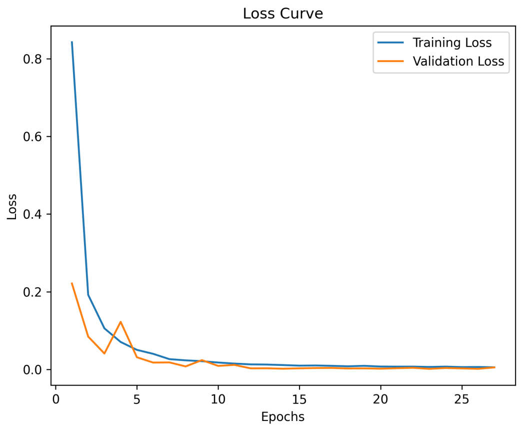


FIGURE 5: Training and Validation Loss of the Lightweight CNN Model using Unicauca-Dataset

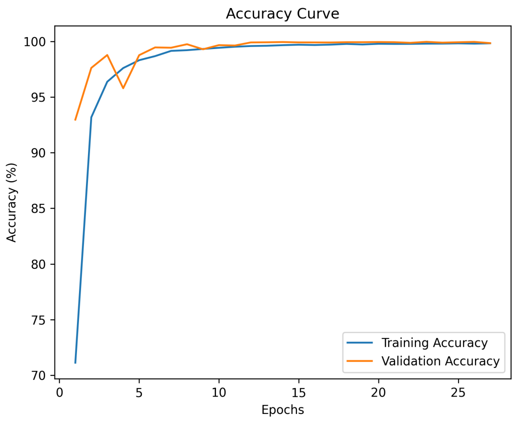


FIGURE 6: Training and Validation Accuracy of the Lightweight CNN Model using Unicauca-dataset

TABLE 1: Performance Evaluation of CNN, RNN, Ulfar, ABL-TC and Lightweight CNN Models using Unicauca-Dataset

Model	Accuracy	Precision	Recall	F1-Score
CNN	99.90%	99.87%	99.89%	99.88%
RNN	98.53%	98.55%	98.53%	98.53%
Ulfar	99.97%	99.97%	99.97%	99.97%
ABL-TC	99.93%	99.93%	99.93%	99.93%
Proposed Lightweight CNN	99.96%	99.96%	99.96%	99.96%

TABLE 2: Overall Performance Comparison with other Models using Unicauca-Dataset

Model	Parameters	FLOPS	Inference Time GPU(s)	Inference Time CPU(s)	Accuracy (%)
RNN	4.1M	2.55 MFLOPS	0.75	24.99	98.53
CNN	2.8M	9.94 MFLOPS	0.25	1.33	99.47
Ulfar	473K	0.223 MFLOPS	0.25	20.86	99.97
ABL-TC	86K	1.4 MFLOPS	0.165	0.722	99.93
Proposed Lightweight CNN	30.6K	0.627 MFLOPS	0.43	1.49	99.96

Table 2 offers a comprehensive comparison of the models in terms of the number of parameters, FLOPS, inference time on CPU and GPU, and accuracy. The proposed lightweight CNN model is particularly noteworthy, utilizing only 30.6K parameters and requiring 0.627 MFLOPS. In contrast, the traditional CNN baseline employs 2.8M parameters with 9.94 MFLOPS, the RNN model uses 4.1M parameters with 2.55 MFLOPS, the Ulfar model is built with 473K parameters and 0.223 MFLOPS, and the ABL-TC model uses 86K parameters with 1.4 MFLOPS. This dramatic reduction in model size and computational load in our proposed model is primarily due to the efficient use of DSCs, which decouple spatial and channel-wise operations to eliminate redundant computations. The Ulfar model, with its 473K parameters, achieves an impressively low FLOPS count of 0.223 MFLOPS. This is accomplished by incorporating a multi-scale feature attention mechanism alongside residual blocks to optimize computational pathways. In contrast, our proposed lightweight CNN model reduces the overall parameter count to just 30.6K by leveraging DSCs. However, this design choice introduces additional pointwise convolutions and branch concatenations, which contribute to a slightly higher FLOPS count of 0.627 MFLOPS. This trade-off is acceptable given that our model still delivers competitive accuracy and is ideally suited for deployment in resource-constrained environments, where the significant reduction in parameters greatly benefits memory usage and real-time processing.

Regarding inference time, the GPU inference times are competitive across models. While the proposed model's GPU time (0.43 s) is slightly higher than that of the CNN baseline (0.25 s) and the ABL-TC model (0.165 s), its CPU inference time of 1.49 s remains efficient compared to the significantly slower RNN (24.99 s) and Ulfar's model (20.86 s). Notably, the Ulfar model demonstrates superior GPU performance due to its advanced multi-scale feature attention mechanism, yet its intricate architecture incurs a substantial penalty in CPU inference time, making it less favorable in CPU-bound scenarios. The minor overhead observed in the proposed model's GPU performance can be attributed to the current implementation of separable convolutions, which, although computationally efficient (lower FLOPS), may not be as optimized in all GPU libraries. Nonetheless, the drastic reduction in parameters and FLOPS makes our model highly attractive for implementation in environments with limited resources, such as IoT devices.

TABLE 3: Performance Evaluation of CNN, RNN, and Lightweight CNN Models using UTMobileNetTraffic2021 Dataset

Model	Accuracy	Precision	Recall	F1-Score
CNN	55.36%	60.57%	55.37%	56.10%
RNN	67.79%	74.81%	67.81%	69.75%
Lightweight CNN	70.57%	76.01%	70.57%	72.06%

Table 3 shows that the proposed lightweight CNN model outperforms traditional CNN and RNN models across key performance metrics, including accuracy, precision, recall, and F1-score using UTMobileNetTraffic2021 dataset. Specifically, the lightweight CNN achieves an accuracy of 70.57%, with precision, recall, and F1-scores of 76.01%, 70.57%, and 72.06%, respectively. This enhancement can be attributed to the model's utilization of DSCs, which markedly decrease the number of parameters and computational complexity, thereby augmenting efficiency without degrading performance. In contrast, the traditional CNN and RNN models suffer from higher computational costs, limiting their applicability in resource-constrained environments. It is important to highlight that the overall lower accuracy observed on the UTMobileNetTraffic2021 dataset, compared to the main dataset, can be attributed to its considerably smaller sample size. The reduced number of training instances limits the diversity of patterns that the models can learn, thus affecting their generalization capability. This challenge persists even after balancing the class distribution through SMOTE.

In summary, the proposed lightweight CNN model strikes an excellent balance between computational efficiency and high classification accuracy. Its reduced model size and lower computational demands, achieved through DSCs, enable rapid and reliable network traffic classification, making it a compelling choice for real-world applications in Network Traffic Classification.

E. Explainability Results with LIME and SHAP using Unicauca-Dataset

To ensure transparency and trust in the model's decision-making process, we employed XAI techniques—specifically, LIME [39] and SHAP [38]. These tools enable us to understand how different features contribute to the model's predictions, thereby enhancing the interpretability of the lightweight CNN model. This transparency helps decision-makers trust and validate the classification outcomes.

The LIME explanation for the lightweight CNN traffic classification model, as illustrated in Figure 7, reveals that *L7Protocol*, *RST.Flag.Count* and *ECE.Flag.Count* have the most significant impact on the model's prediction for the specific "Dropbox" class, as shown in Figure 7. In this instance, *L7Protocol* has the strongest negative contribution, indicating that a higher value for this feature reduces the likelihood of the prediction being classified as Dropbox. Similarly, *RST.Flag.Count* and *ECE.Flag.Count* show negative impacts on the classification. On the other hand, certain features, such as *Active.Max*, *Idle.Mean*, and *Bwd.IAT.Max*, exhibit positive contributions, which help increase the confidence in the Dropbox classification. LIME's ability to provide localized, instance-specific explanations makes it useful for understanding the model's behavior in individual cases. However, this instance-specific focus means that feature importance may vary between different traffic types and classification instances.

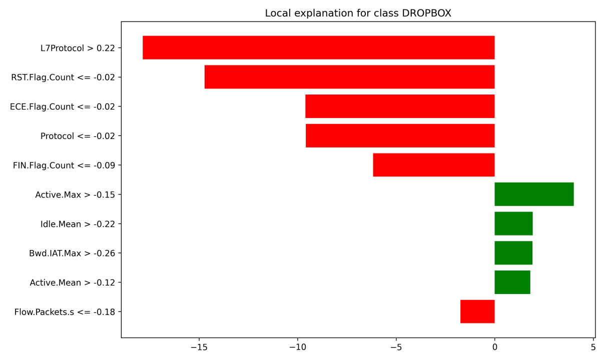


FIGURE 7: LIME Explanation for Lightweight CNN

The SHAP summary plot, as shown in Figure 8, presents a global interpretation across all classifications, identifying *L7Protocol*, *Idle.Max*, *Idle.Mean*, and *Idle.Min* as the most influential features overall, as shown in Figure 8. SHAP uses cooperative game theory to compute the average contribution of each feature, providing a more stable and consistent feature ranking. For example, *L7Protocol* consistently appears as the most significant feature across multiple instances, making it critical for model performance across the dataset. This global perspective allows SHAP to offer more generalizable insights into feature importance.

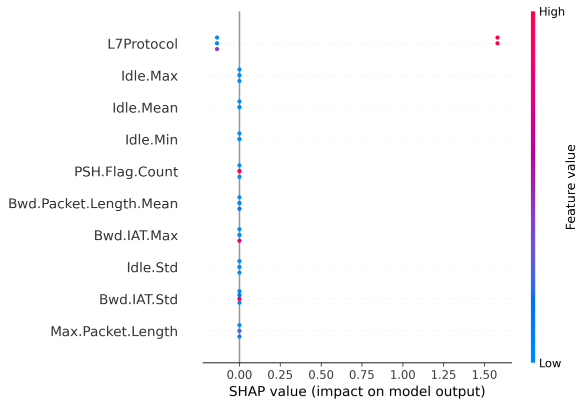


FIGURE 8: SHAP Summary Plot for Lightweight CNN

LIME and SHAP offer complementary views into the model's decision-making process. LIME is focused on local, instance-specific interpretations by perturbing input features and observing their effects on individual predictions. This leads to feature rankings that can vary significantly depending on the particular instance being analyzed, as seen in the importance of features like *ECE.Flag.Count* and *RST.Flag.Count* in the "Dropbox" classification. LIME is particularly useful for analyzing how the model behaves in specific cases, allowing for a detailed, localized understanding.

In contrast, SHAP provides a global perspective by evaluating all possible feature combinations, resulting in a consistent and stable ranking of feature importance across the entire dataset. As demonstrated in the SHAP plot, features such as *L7Protocol*, *Idle.Max*, and *Idle.Mean* consistently contribute to the model's predictions across various instances. SHAP's ability to present a global view makes it ideal for understanding which features are most important for the overall model, offering more robust insights into the model's general behavior. Thus, while LIME excels at explaining individual predictions, SHAP provides a broader, more stable understanding of feature importance across the entire dataset. Using both methods together can offer a comprehensive view, balancing the need for local interpretability and global consistency.

F. Discussion and Limitations

The experimental results indicate that the proposed lightweight NTC model based on DSCs offers an optimal solution for NTC, particularly in environments with limited computational resources. This high level of accuracy, when combined with its drastically reduced model size, underscores its suitability for real-time applications.

- **Computational Efficiency:** The proposed model exhibits remarkable efficiency by utilizing only 30.6K parameters—approximately 91.5 times fewer than the CNN baseline's 2.8M parameters. It also requires 0.627 MFLOPS, which is nearly 15.9 times fewer than the CNN baseline and about 4 times fewer than the RNN model. Although its FLOPS are 2.8 times higher than those of the Ulfar model, this difference stems from architectural choices; the Ulfar model employs multi-scale feature attention with optimized residual blocks to minimize redundant operations, whereas our design uses additional pointwise convolutions and branch concatenations inherent to DSCs.

- **Model Accuracy and Robustness:** Despite its lightweight design, the proposed model achieves an impressive accuracy of 99.96%, which is marginally lower than the Ulfar model (99.97%) but significantly higher than the CNN baseline (99.90%), ABL-TC (99.93%), and RNN (98.53%). This robust performance confirms the model's ability to maintain high classification accuracy while substantially reducing computational demands.
- **Explainability:** The integration of XAI techniques, such as LIME and SHAP, enhances the model's transparency by providing both local and global insights into its decision-making process. Importantly, these XAI modules are decoupled from the prediction process and are invoked on-demand, ensuring that the core model's real-time performance remains unaffected.
- **Scalability:** The drastic reduction in parameters and computational load enables rapid inference and makes the proposed model highly scalable for deployment in resource-constrained environments, including IoT and edge computing networks.

In addition to these points, we have conducted an analysis of the model's classification performance across different types of network traffic. Our results indicate that the proposed lightweight CNN model consistently maintains high performance across various application protocols. This analysis not only reinforces the model's robustness but also provides valuable insights into its advantages and potential limitations when applied to diverse real-world scenarios.

Although the proposed lightweight CNN model demonstrates an excellent balance between accuracy and efficiency, its evaluation has been confined to two datasets. In summary, the proposed lightweight CNN model based on DSCs achieves state-of-the-art performance in network traffic classification by combining high accuracy, computational efficiency, and enhanced explainability. These attributes position it as a highly promising solution for real-world applications in environments with stringent resource constraints.

V. CONCLUSION AND FUTURE WORKS

In this study, we proposed and evaluated a lightweight NTC DSC-based CNN, which significantly enhances computational efficiency without compromising accuracy. Real traffic datasets such as Unicauca and UTMobileNetTraffic2021 are used to measure improvement in the number of parameters, FLOPS, inference time on CPU and GPU, accuracy, recall, F1-score, and precision. Our proposed model achieved an impressive overall accuracy of 99. 96%, closely competing with Ulfar model 99.97% and outperforming the traditional CNN baseline (99.90%), ABL-TC (99.93%), and RNN (98.53%). Notably, our model utilizes only 30.6K parameters—approximately $91x$ fewer than the CNN baseline's 2.8M parameters—and requires 0.627 MFLOPS, which is about $16x$ fewer than the CNN baseline and roughly $4x$ fewer than the RNN model. Although the Ulfar model achieves a lower FLOPS count (0.223 MFLOPS) with 473K parameters, its larger model size emphasizes the efficiency of our depthwise separable approach. The proposed lightweight CNN model demonstrates competitive overall performance with an efficient CPU inference time (1.49 s) and high classification accuracy. Furthermore, the incorporation of XAI methodologies, including LIME and SHAP, provides valuable insights into feature importance and enhances the transparency of the model's decision-making process. These attributes, combined with the model's scalability and low computational overhead, make it particularly well-suited for deployment in resource-constrained environments like IoT and edge computing networks.

As a future direction, although the proposed lightweight model offers significant advantages, there are some opportunities for further enhancement:

- **Generalization Across Diverse Datasets:** Extend evaluations to datasets with varied network traffic patterns to improve the model's generalizability in different real-world scenarios.
- **Edge Deployment and Federated Learning:** Explore deploying the model in edge computing environments using federated learning approaches, ensuring real-time classification while preserving data privacy.
- **Further Architectural Optimizations:** Investigate advanced hyperparameter optimization techniques, such as neural architecture search, to further refine the model and potentially reduce inference times without compromising accuracy.

By addressing these avenues, we aim to further enhance the applicability and performance of lightweight NTC models, making them even more effective for deployment in IoT edge networks and smart cities.

ACKNOWLEDGMENT

”The authors acknowledge the support from the Interdisciplinary Research Center for Intelligent Secure Systems and the Department of Computer Engineering at King Fahd University of Petroleum & Minerals (KFUPM)”.

REFERENCES

[1] M. R. Choudhury, N. Muraleedharan, P. Acharjee, and A. T. George, ”Network Traffic Classification Using Supervised Learning Algorithms,” in *ICCECE 2023 - International Conference on Computer,

Electrical and Communication Engineering*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICCECE51049.2023.10084931.

[2] Y. S. Kuruba Manjunath, S. Zhao, H. Abou-Zeid, A. Bin Sediq, R. Atawia, and X. P. Zhang, "Segmented Learning for Class-of-Service Network Traffic Classification," in *2022 IEEE Global Communications Conference, GLOBECOM 2022 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 6115–6120. doi: 10.1109/GLOBECOM48099.2022.10000867.

[3] M. A. Anwar, M. Agrawal, N. Saroha, and A. Bhat, "Attention to Traffic: Network Traffic Classification using Attention-Based CNNs," in *2023 14th International Conference on Computing Communication and Networking Technologies, ICCCNT 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICCCNT56998.2023.10306767.

[4] O. Aouedi, K. Piamrat, and B. Parrein, "Ensemble-Based Deep Learning Model for Network Traffic Classification," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4124–4135, Dec. 2022. doi: 10.1109/TNSM.2022.3193748.

[5] Ericsson Mobility Report, https://www.ericsson.com/49dd9d/assets/local/reports-papers/mobility-report/documents/2023/ericsson-mobility-report-june-2023.pdf, (accessed 4 September 2024).

[6] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, vol. 24, no. 11, pp. 1–10, 2015.

[7] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conference Intelligence and Security Informatics, ISI*, 2017, pp. 43–48.

[8] M. Lotfollahi, R. Shirali Hossein Zade, M. Jafari Siavoshani, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, pp. 1999–2012, 2020.

[9] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 2, pp. 445-458, 2019.

[10] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning," in *2018 Network Traffic Measurement and Analysis Conference, TMA*, pp. 1–8, 2018.

[11] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," *Comput. Network.*, vol. 165, Article 106944, 2019.

[12] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Toward effective mobile encrypted traffic classification through deep learning," *Neurocomputing*, vol. 409, pp. 306–315, 2020.

[13] H. Huang, H. Deng, J. Chen, L. Han, and W. Wang, "Automatic multi-task learning system for abnormal network traffic detection," *Int. J. Eng. Technol. Learn.*, vol. 13, no. 4, pp. 4–20, 2018.

[14] V. Tong, H. A. Tran, S. Souihi, and A. Mellouk, "A novel QUIC traffic classifier based on convolutional neural networks," in *2018 IEEE Global Communications Conference, GLOBECOM*, pp. 1–6, 2018.

[15] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18042-18050, 2017.

[16] Z. Bu, B. Zhou, P. Cheng, K. Zhang, and Z.-H. Ling, "Encrypted network traffic classification using deep and parallel network-in-network models," *IEEE Access*, vol. 8, pp. 132950–132959, 2020.

[17] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1171–1179, 2019.

[18] C. Liu, Z. Cao, G. Xiong, G. Gou, S.-M. Yiu, and L. He, "MAMPF: Encrypted traffic classification based on multi-attribute Markov probability fingerprints," in *2018 IEEE/ACM 26th International Symposium on Quality of Service, IWQoS*, pp. 1–10, 2018.

[19] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, A. Pescapé, "XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225-4246, 2021.

[20] Baek, U.-J., Park, J.-T., Jang, Y.-S., Kim, J.-S., Choi, Y.-S., and Kim, M.-S.: A filter-and-refine approach to lightweight application traffic classification. *ICT Express* (2024).

[21] Aceto, G., Ciuonzo, D., Montieri, A., Persico, V., and Pescape, A.: AI-powered Internet Traffic Classification: Past, Present, and Future. *IEEE Communications Magazine* (2023).

[22] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, A. Pescapé, "MIRAGE: Mobile-app Traffic Capture and Ground-truth Creation," in *Proc. IEEE Int. Conference on Computing Communication Security, ICCS*, 2019, pp. 1–8.

[23] B. Mohammed, M. Hamdan, J. S. Bassi, H. A. Jamil, S. Khan, A. Elhigazi, D. B. Rawat, I. B. Ismail, M. N. Marsono, "Edge computing intelligence using robust feature selection for network traffic classification in internet-of-things," *IEEE Access*, vol. 8, pp. 224059-224070, 2020.

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

[25] Ren, X., Gu, H., Wei, W.: Tree-RNN: Tree structural recurrent neural network for network traffic classification. Expert Systems with Applications 167, 114363 (2021).

[26] N. Kukreja et al., "Training on the Edge: The why and the how," in *Proc. - 2019 IEEE 33rd Int. Parallel Distrib. Process. Symp. Work. IPDPSW 2019*, pp. 899-903, 2019.

[27] S. K. Prashanthi, S. A. Kesanapalli, and Y. Simmhan, "Characterizing the Performance of Accelerated Jetson Edge Devices for Training Deep Learning Models," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 3, Dec. 2022.

[28] A. Ariffin, F. Zaki, and N. B. Anuar, "Leveraging Federated Learning and XAI for Privacy-Aware and Lightweight Edge Training in Network Traffic Classification," in *2023 IEEE International Conference on Computing (ICOCO)*, pp. 47-52, IEEE, 2023.

[29] V. Dignum, "Responsible artificial intelligence: Designing AI for human values," *ITU J. ICT Discoveries Special Issue*, vol. 1, pp. 1-8, Sep. 2017.

[30] R. Inam, A. Terra, A. Mujumdar, E. Fersman, and A. V. Feljan, "Explainable AI—How humans can trust AI," Apr. 2021, [online] Available: https://www.ericsson.com/en/reports-and-papers/white-papers/explainable-aihow-humans-can-trust-ai.

[31] H. Hagras, "Toward human-understandable, explainable AI," *IEEE Comput.*, vol. 51, no. 9, pp. 28–36, Sep. 2018.

[32] M. Kull et al., "Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with Dirichlet calibration," in *Proc. 32th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, p. 38.

[33] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, 2005, pp. 625–632.

[34] D. Widmann, F. Lindsten, and D. Zachariah, "Calibration tests in multiclass classification: A unifying framework," in *Proc. 33th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 12236–12246.

[35] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. Torr, and P. K. Dokania, "Calibrating deep neural networks using focal loss," in *Proc. 34th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 1–9.

[36] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" in *Proc. 32th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 4696–4705.

[37] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 818–833.

[38] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 30th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 4765–4774.

[39] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2016, pp. 1135–1144.

[40] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. 34th PMLR Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3319–3328.

[41] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proc. 34th PMLR Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3145–3153.

[42] M. A. Anwar, M. Agrawal, N. Saroha, and A. Bhat, ”Attention to Traffic: Network Traffic Classification using Attention-Based CNNs,” in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1-6, IEEE, 2023.

[43] M. M. K. Abid Hussain, M. Jaseemuddin, and R. Kashef, ”Network Traffic Classification Using Distributed ML-Based Data Parallelization Approach,” in *2023 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Sep. 2023, pp. 539–546, doi: 10.1109/CCECE58730.2023.10288743.

[44] “VPN 2016 — Datasets — Research — Canadian Institute for Cybersecurity — UNB.” Accessed: Feb. 17, 2024. [Online]. Available: https://www.unb.ca/cic/datasets/vpn.html

[45] Fukushima, K.: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybern. 36, 193–202 (1980).

[46] Sifre, L., Mallat, S.: Rigid-Motion Scattering for Texture Classification. (2014).

[47] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (2017).

[48] Chollet, F.: Xception: Deep Learning with Depthwise Separable Convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1800-1807 (2017).

[49] Sze, V., Chen, Y.-H., Yang, T.-J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proc. IEEE 105(12), 2295–2329 (2017).

[50] Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6848–6856 (2018).

[51] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1314–1324 (2018).

[52] Felemban, M., Ghaleb, M., Saaim, K., AlSaleh, S., Almulhem, A.: File Fragment Type Classification Using Light-Weight Convolutional Neural Networks. IEEE Access, 12, pp. 157179–157191 (2024).

[53] Mittal, G., Korus, P., Memon, N.: Fifty: Large-scale file fragment type identification using convolutional neural networks. IEEE Transactions on Information Forensics and Security 16, 28–41 (2020).

[54] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015).

[55] Wu, Y., He, K.: Group normalization. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19 (2018).

[56] Rojas, J.S., Gallón, Á.R., Corrales, J.C.: Personalized service degradation policies on OTT applications based on the consumption behavior of users. In: Computational Science and Its Applications—ICCSA, Cham, Switzerland: Springer, pp. 543-557 (2018).

[57] Ren, J., Dubois, D.J., Choffnes, D., Mandalari, A.M., Kolcun, R., Haddadi, H.: Information exposure from consumer IoT devices: A multidimensional, network-informed measurement approach. In: Proceedings of the Internet Measurement Conference, pp. 267-279 (2019).

[58] Sivanathan, A., Gharakheili, H.H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., Sivaraman, V.: Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. IEEE Transactions on Mobile Computing, 18(8), pp. 1745-1759 (2019). doi: 10.1109/TMC.2018.2866249.

[59] Wang, Y., Yun, X., Zhang, Y., Zhao, C., Liu, X.: A multi-scale feature attention approach to network traffic classification and its model explanation. IEEE Transactions on Network and Service Management, 19(2), pp. 875-889 (2022).

[60] Wei, W., Gu, H., Deng, W., Xiao, Z., Ren, X.: ABL-TC: A lightweight design for network traffic classification empowered by deep learning. Neurocomputing, 489, pp. 333-344 (2022).

[61] IP Network Traffic Flows Labeled With 75 Apps, Aug. 2024, [online] Available: https://kaggle.com/jsrojas/ip-network-traffic-flows-labeled-with-87-apps.

[62] Heng, Y., Chandrasekhar, V., Andrews, J.G.: UTMobileNetTraffic2021: A labeled public network traffic dataset. IEEE Networking Letters, 3(3), pp. 156-160 (2021).

Mustafa Ghaleb is a Postdoctoral Research Fellow at the Interdisciplinary Research Center for Intelligent Secure Systems (IRC-ISS) at KFUPM in Saudi Arabia. He obtained his M.Sc. and Ph.D. degrees in Computer Science from KFUPM. Ghaleb's research interests include computer networks, cybersecurity, Internet of Things (IoT), distributed computing, NLP, trust modeling, and deep learning applications in various domains.

Mosab Hamdan (Senior Member, IEEE) is currently an Assistant Professor at the School of Computing, National College of Ireland, Dublin. He has previously worked with several academic and research institutions across Malaysia, Brazil, Saudi Arabia, and Ireland. His research focuses on computer networks, cybersecurity, IoT, smart cities, intelligent transportation systems, and future network technologies.

Abdulaziz Barnawi (Member, IEEE) is an Assistant Professor at Department of Computer Engineering and a Research Scholar at the Interdisciplinary Research Center for Intelligent Secure Systems (IRC-ISS), both at KFUPM. His research interests include the Internet of Things (IoT), wireless sensor networks, network management, cybersecurity, and the Internet of Drones. He has been both principal investigator and collaborator on several funded projects and has authored or co-authored numerous peer-reviewed articles.

Muhammad Gambo, is currently pursuing a Ph.D. in Computer Engineering at KFUPM (2024–present). He holds an M.Sc. in Information Security from the University of Technology Malaysia (2022) and a B.Eng. in Computer Engineering from Ahmadu Bello University, Zaria. His research interests include dynamic access control, trust management, privacy preservation, and the application of AI/ML in cybersecurity.

Saheed Bello, is a PhD student in the department of computer engineering at KFUPM. He completed his master's degree in computer engineering at King Abdul-Aziz University, Jeddah, Saudi Arabia. His research interests revolve around the development and application of deep learning models in solving critical human problems in the area of Biomedical Systems, Computer Vision and Cybersecurity.

Aliyu Habib, is currently pursuing his PhD. in computer science and engineering at Hamad Bin Khalifa University (HBKU), Qatar. He got his master's degree in computer engineering from KFUPM, Dammam, Saudi Arabia, in 2024. His research interests include hardware security, logic locking, and complementary metal-oxide semiconductor integrated circuit design.

Abubakar Danasabe, (Student Member, IEEE) is currently pursuing his M.Sc. degree in Computer Networks and Cybersecurity at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. His research interests include network security and machine learning.