entropy

MDPI

Article

Utilizing TabNet Deep Learning for Elephant Flow Detection by Analyzing Information in First Packet Headers

Bartosz Kądziołka *, Piotr Jurkiewicz, Robert Wójcik and Jerzy Domżał

Institute of Telecommunications, AGH University of Krakow, 30-054 Krakow, Poland
* Correspondence: kadziolka@agh.edu.pl

Abstract: Rapid and precise detection of significant data streams within a network is crucial for efficient traffic management. This study leverages the TabNet deep learning architecture to identify large-scale flows, known as elephant flows, by analyzing the information in the 5-tuple fields of the initial packet header. The results demonstrate that employing a TabNet model can accurately identify elephant flows right at the start of the flow and makes it possible to reduce the number of flow table entries by up to 20 times while still effectively managing 80% of the network traffic through individual flow entries. The model was trained and tested on a comprehensive dataset from a campus network, demonstrating its robustness and potential applicability to varied network environments.

Keywords: flows; flow table; elephant; mice; traffic engineering; machine learning; TabNet; feature importance; input information

Citation: Kądziołka, B.; Jurkiewicz, P.; Wójcik, R.; Domżał, J. Utilizing TabNet Deep Learning for Elephant Flow Detection by Analyzing Information in First Packet Headers. *Entropy* **2024**, *26*, 537. https://doi.org/10.3390/e26070537

Academic Editors: Irad E. Ben-Gal and Amichai Painsky

Received: 15 May 2024
Revised: 18 June 2024
Accepted: 21 June 2024
Published: 22 June 2024

1. Introduction

The term *elephant flows* refers to the most substantial data transfers across the Internet, which, despite their limited numbers, usually carry the majority of the traffic. In contrast, the numerous *mouse flows* constitute a large portion of the flow count but account for only a minor fraction of total traffic. This imbalance surpasses the conventional 80/20 ratio defined by the Pareto principle. Recent studies, including references such as [1,2], have revealed that a mere 0.2–0.4% of all flows might be responsible for as much as 80% of the entirety of Internet traffic, showcasing an extreme concentration of data within a small fraction of flows.

The approach of flow-based traffic engineering has recently emerged as a powerful technique for addressing the challenges of escalating network demands while preserving the quality of service (QoS) [3–5]. This strategy involves assigning a unique forwarding entry to each flow in the switch's memory, with each entry detailing the subsequent hop along the flow's path. This arrangement allows for the use of varied paths for flows that share the same source and destination, thereby facilitating multipath routing. Moreover, paths for incoming flows can be chosen based on present or expected network congestion, enabling an adaptive routing that effectively avoids congested links. Furthermore, this method of flow-based adaptive routing is known to offer higher stability compared to conventional dynamic load-balancing techniques.

The primary issue in flow-based traffic engineering arises from the fact that the number of concurrent flows in a network often exceeds the capacity of the flow tables within switches [6]. Furthermore, in centralized software-defined networks (SDNs), there is a bottleneck concerning the controller's ability to handle new flow setups due to its throughput limitations. Beyond the issue of capacity, having fewer entries in the flow tables can lead to faster table lookups, thereby enhancing the packet switching speed. One viable approach to mitigate these challenges involves dedicating entries exclusively to the largest flows. Consequently, the majority of smaller flows could be directed along default, shortest-path routes. This strategy significantly reduces the number of flow table

entries while still effectively managing a substantial volume of traffic through specialized, flow-specific entries. It is crucial to predict these values as early as possible to quickly establish individual entries and route most of their packets through specific paths. Ideally, flows should be classified with their initial packet to prevent mid-connection rerouting, which can disrupt transport protocols' path state estimations. Additionally, we find that first packet classification is mostly an unexplored field, especially within the context of SDN traffic engineering. Therefore, we decided to focus solely on the first packet classification to fill this gap.

Our research evaluates the efficacy of the TabNet [7] deep tabular data learning architecture, with a particular focus on metrics vital for traffic engineering within SDNs. The key contributions of our work are as follows:

- **Traffic Coverage:** We examine the volume of traffic managed by flows classified as elephants following their identification, termed *traffic coverage*.
- **Flow Table Reduction:** We analyze the reduction in the necessity for individual flow entries in the tables, denoted as *flow table operation reduction*.
- **Entropy Analysis:** We provide an analysis of the average information entropy contained in each 5-tuple field in packet headers.
- **Feature Significance:** We identify which 5-tuple fields were the most significant for predictions.

2. Related Work

The strategy of selectively managing elephant flows dates back to 1999 when the idea of adaptively routing substantial data flows was introduced [8]. Initially, due to the hardware constraints of the era, the concept was largely theoretical and confined to academic discussions. However, the rise of SDNs has revitalized interest in this approach. In the contemporary networking landscape, a controller with comprehensive insight into network dynamics is well positioned to efficiently oversee large flows, leveraging the advanced capabilities of SDNs.

The Hedera traffic engineering system, unveiled in [9], was created to dynamically reroute flows through an embedded controller once they surpassed a predefined threshold, guiding these flows along paths selected in real time. It presupposes that edge devices are responsible for collecting comprehensive flow statistics via OpenFlow counters, with a focus on optimizing the performance of non-edge devices. DevoFlow, introduced in [10], emphasizes the management of elephant flows by implementing sampling techniques and utilizing threshold values for their identification. Nevertheless, the evaluation of DevoFlow's effectiveness is conducted based on the network's aggregated performance, not on flow table characteristics. A similar approach to DevoFlow is explored in [11], where elephant flows are identified at edge devices using an adapted Bloom filter. This method's underlying traffic model, assuming a disproportionate contribution of 20% of the flows to 80% of the traffic, diverges significantly from real-world data distributions, as recent research, such as [1,2], suggests a much more tail-skewed distribution.

The referenced studies primarily employ rudimentary techniques such as sampling, counters, and threshold settings for the detection of large flows. However, there has been a shift towards more sophisticated, machine-learning-based approaches in recent years. For instance, a decision tree model dedicated to identifying elephant flows was introduced and assessed in [12], with a particular emphasis on the *accuracy* of detection. In another study, ref. [13] by Poupart et al. explored the capabilities of three different machine learning (ML) strategies for estimating flow sizes and categorizing them as elephant flows. Their analysis was based on a comprehensive dataset of three million flows, covering both TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Their evaluation focused on two principal metrics: the success rate in correctly identifying large flows (*true positive rate*) and the success rate in accurately identifying smaller flows (*true negative rate*).

In [14], Liu et al. recommend the application of a random forest decision tree for pinpointing eight essential features crucial for developing a classification model. They

introduce a dual-layered architectural framework that involves an initial pre-classification phase at the edge devices within an SDN setup, followed by a more detailed classification at the network's central controller. This classification scheme distinguishes between four distinct types of flows: elephant, cheetah, tortoise, and porcupine, each representing different characteristics and behaviors within the network traffic. The research primarily evaluates the effectiveness of this system based on two metrics: the precision of the flow classification and the time delay associated with the classification procedure.

In their research, Hamdan et al. [15] introduce a two-level classification framework for network traffic, initially sorting flows at switches and finalizing classifications at the central controller. This method uses the count-min sketch algorithm at the switch level to separate mice from potential elephant flows, with a decision tree at the controller for final decisions. The system's algorithms are periodically refreshed with data from the controller, emphasizing classifier accuracy, and validated with real traffic models.

He et al. [16] and Qian et al. [17], in 2022, each proposed sketch-based solutions for flow table optimization. He et al. developed a single-level, lightweight scheme, while Qian et al. introduced TCAM-based storage for elephant flow labels to balance accuracy between elephant and mouse flow identification. Both studies utilized real Internet Service Provider (ISP) packet traces for evaluation, indicating the practical effectiveness of their approaches in traffic management.

In the study [18], da Silva et al. introduced a predictive model using the Locally Weighted Regression (LWR) algorithm to estimate the size and duration of new network flows by examining patterns from previous flows and their immediate correlations. Following up, in 2022, employing a hashing mechanism inspired by the Cuckoo Search meta-heuristic for enhanced flow management [19] was proposed by the same authors. Pekar et al. presented a novel threshold-agnostic heavy-hitter classification system [20], which utilizes template matching to identify elephant flows based on the packet size distribution observed in the initial packets, offering a nuanced method for flow classification without predetermined thresholds.

The CrossBal system, detailed in [21], is a hybrid load-balancing solution that employs Deep Reinforcement Learning (DRL) to specifically address elephant flows through a three-level detection mechanism, including threshold-based filtering, followed by rerouting for efficient load distribution. In a related study, Wassie et al. [22] introduced a deep learning approach utilizing deep autoencoders, gradient boosting, and autoML predictive algorithms like eXtreme gradient boosting (XGBoost) [23] and the gradient boosting machine (GBM) [24], aimed at enhancing flow management.

All the mentioned studies focus on classifying flows after observing several initial packets. However, our goal is to identify a flow as quickly as possible, ideally based on the information carried in the first packet. Durner et al. [25] achieved flow classification using just the first packet's 5-tuple data and its size. Hardegen et al. [26] proposed using multiclass prediction instead of binary classification (elephant/mouse) with a deep neural network to predict flow characteristics from the first packet's 5-tuple. This approach follows a similar methodology to their earlier work [27] on predicting a flow's bit rate from the first packet's 5-tuple.

Regarding the most recent works, in 2023 Gomez et al. [28] evaluated several machine learning algorithms for classifying flows from the first packet. Similar to other studies, it focused on classification accuracy and not on flow table impact or traffic coverage. Xie et al.'s 2024 paper [29] proposed a two-stage decision tree system for elephant flow classification. The first stage is utilizing information contained in first packet headers. The system was developed in P4, but tested only in an emulator, lacking real-device validation.

Recent studies have also applied neural networks for network flow classification with a focus on QoS rather than traffic engineering. Alkhalidi et al. [30] introduced a one-dimensional convolutional neural network for classifying flows into various classes using packet header information. A notable innovation is the automatic selection of specific packet header bits, reducing feature count, processing time, and energy consumption while

maintaining satisfactory accuracy. Yaseen et al. [31] employed a similar approach to classify traffic and assign the Differentiated Services Code Point (DSCP) field, implementing their system within an SDN controller and testing it in the Mininet emulator for emergency traffic prioritization scenarios.

As can be seen, the referenced studies focus on flow classification accuracy or true positive/true negative rates. However, they neglect the practical implications of these algorithms for traffic engineering goals. For instance, misclassifying the largest flow in a network can substantially impact traffic coverage, far more than the misclassification of smaller flows. The metrics employed in these studies do not account for such nuances. Specifically, there has been a lack of focus on metrics essential for traffic engineering, such as the reduction in flow table entries or the volume of traffic managed after classification. These aspects are critical for assessing the load on switches and controllers, as well as for understanding the broader effects on traffic engineering strategies and overall system performance. Moreover, an analysis of the significance and entropy of the information contained in the first packet's 5-tuple—specifically, identifying which fields are crucial for detecting an elephant flow at its inception—has not yet been addressed.

3. Methodology

Predicting the size of a flow based on its initial packet is achievable with a type of machine learning known as regression. Regression, a principal method of supervised learning, requires labeled input data to train the model for predictive tasks. The TabNet model utilized in this research is available in the GitHub repository: https://github.com/dreamquark-ai/tabnet (accessed on 20 June 2024).

3.1. Training Environment

Training was conducted on a high-performance machine equipped with the following specifications:

- **Memory:** 128 GB RAM
- **Graphics Processing Unit (GPU):** NVIDIA GeForce RTX 4090 with 24 GB of VRAM
- **Central Processing Unit (CPU):** Intel Core i9-13900KF with 24 cores

These resources were more than sufficient to conduct the training and validation processes. In fact, the computational resources were not consumed beyond 20%, even when using the most demanding hyperparameter combinations. Training times for a single epoch ranged from as short as 10 s to as long as 2 min, depending on the complexity of the hyperparameter configurations. This ample capacity ensured efficient handling of the large dataset and complex computations involved in training the TabNet model, facilitating timely convergence and optimal performance.

Inference, which involves using the trained model to make predictions on a validation dataset, was also highly efficient. Inference latency varied from 10 to 23 s, depending on the model complexity and the size of the validation dataset (maximum 1,303,496; minimum 130,349). This capability is crucial for practical deployment in high-speed network environments where timely decision-making is essential.

3.2. Dataset

The effectiveness of an ML algorithm is significantly influenced by the dataset it is trained on. In our study, we base our evaluation on data that includes length and size distributions of flows, collected from a large campus network over 30 days [1]. For processing these data, we employed the package described in [32].

The dataset in question comprises over 4 billion flows, with its complete flow records amounting to approximately 278 GB in binary format. Given this immense size, we used an anonymized subset of the data for training and evaluating our models, as published in [33]. This subset represents one hour of traffic, encompassing 6,517,484 flows and 547 GB of data transmission. This specific time frame was chosen to ensure it was free of anomalies and that the theoretical reduction rate curve of a perfect elephant classifier for this hour

closely mirrors that of the complete 30-day dataset. In the published open-source dataset, IP addresses were anonymized using the prefix-preserving Crypto-PAn algorithm [34]. As demonstrated in [33], this anonymization process does not affect the performance of the ML models.

3.3. Input Features

The input data, sourced from the flow 5-tuple, encompasses the source IP address, destination IP address, transport layer source port, transport layer destination port, and transport layer protocol, cumulatively contributing to 104 bits. Our investigation focuses on two distinct representations of this input data:

- **Bits**: Each header field is segmented into separate bits, yielding 104 unique features, which are denoted as binary values (0 or 1).
- **Octets**: Headers that exceed 8 bits in length, such as IP addresses or ports, are segmented into distinct octets. This approach produces 13 features, with each feature represented as an 8-bit integer.

3.4. Balancing the Dataset

Achieving a balanced training dataset was key to the effectiveness of the model. In our initial training dataset of 5,213,988 flows, mouse flows greatly outnumbered elephant flows, necessitating measures to balance this disparity for optimal accuracy. The results discussed in this paper stem from the model trained on a balanced dataset, achieved through various ratios, following these steps:

1. Define the *ratio*, e.g., 10%.
2. Calculate the *balanced dataset size* as the size of the initial training dataset multiplied by the ratio (5,213,988 $\times$ 10% = 521,398 flows).
3. Organize the initial training dataset in descending order, with the largest flows positioned at the start.
4. Extract the top half of the *balanced dataset size* number of flows from the start of this sorted list.
5. Randomly select the remaining half of the *balanced dataset size* number of flows from the rest of the initial dataset.

3.5. Training

This phase encompasses the selection of hyperparameters, normalization of labels, and the model training process. The workflow of the training phase is depicted in Figure 1.

The model underwent training on a shuffled, balanced training dataset before its performance was assessed using the validation dataset. Training and validation were carried out with several combinations of hyperparameters. The hyperparameters that were varied are listed in Table 1, while those that remained unchanged throughout all training sessions are detailed in Table 2. We also present the table with the parameters that varied and are coupled directly to the TabNet model in Table 3.

Table 1. Variable hyperparameters.

Batch Size	Learning Rate	Loss Function	Balancing Ratio
2560	1×10^{-3}	Mean Absolute Error (MAE)	10%
5120	3×10^{-3}	Mean Square Error (MSE)	20%
10,240	6×10^{-3}		100% (no balancing)

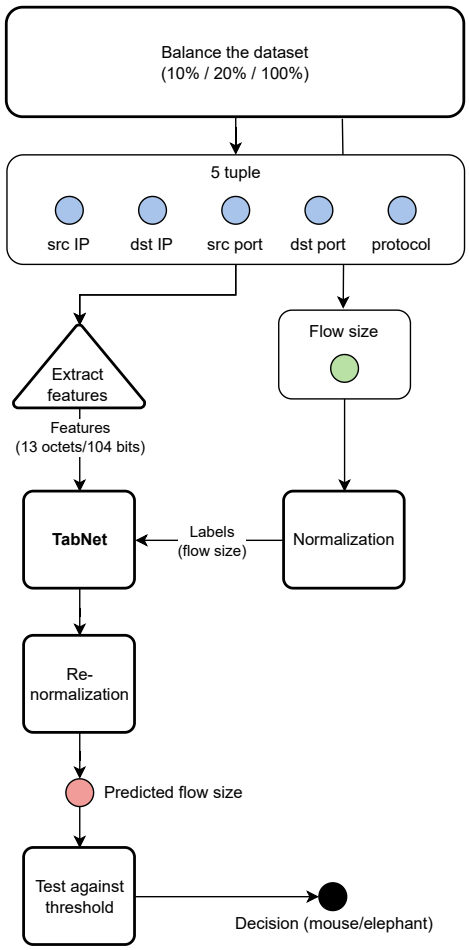

Figure 1. Training and evaluation of the TabNet model.

The batch size was varied to observe its effect on model convergence and generalization. The selected range of 2560 to 10,240 was chosen based on several considerations:

- **Computational Efficiency:** Batch sizes in the range of 2560 to 10,240 were selected to balance between memory usage and computational efficiency. Very small batch sizes might lead to inefficient Graphics Processing Unit (GPU) utilization, while very large batch sizes could exceed the memory limits of the hardware, leading to slower training times due to increased paging or the need to reduce model complexity.
- **Empirical Performance:** Preliminary experiments indicated that this range of batch sizes yielded good performance across various metrics. A batch size of 2560 provided a good trade-off between frequent weight updates and manageable noise in gradient estimates. Increasing the batch size to 5120 and 10,240 allowed for more stable training with slightly slower convergence, which was beneficial in achieving better generalization on the validation set.
- **Model and Data Characteristics:** The nature of the dataset and the model architecture also influenced the choice of batch size. Given the large dataset (5,213,988 flows) and the complexity of the TabNet architecture, batch sizes within this range were found to be effective in leveraging the computational capabilities of modern GPUs while ensuring efficient training dynamics.

Different learning rates were tested to find the optimal balance between convergence speed and stability. A lower learning rate (1×10^{-3}) allows for finer weight adjustments, potentially reducing the risk of overshooting minima. Higher learning rates (6×10^{-3}) can accelerate convergence but may require careful tuning to avoid instability. In general, a larger batch size can lead to more stable training by decreasing the likelihood of overfitting the model. In tandem with increasing the batch size, we also scaled the learning rate. This

approach enabled the model to more quickly locate local minima and maxima without necessitating a proportional adjustment in the number of epochs. The use of different loss functions, MAE and MSE, allows us to assess their impact on regression performance. MAE is less sensitive to outliers compared to MSE, which penalizes larger errors more heavily. Specific balancing dataset ratios were chosen to ensure a sufficient number of elephant flows were included without overwhelming computational resources. A 10% ratio provides a conservative balance, while a 20% ratio allows for a more comprehensive inclusion of elephant flows. The 100% ratio indicates no rebalancing was performed, serving as a control to compare against the balanced scenarios. These steps ensure that minority classes are adequately represented, improving the model's ability to generalize across different traffic types while keeping the computational expense manageable.

Table 2. Constant hyperparameters.

Epochs	Optimizer
200	Adam

The model was trained for up to 200 epochs to ensure sufficient learning time for convergence. This duration was selected based on preliminary experiments indicating that 200 epochs allow the model to adequately learn from the data without overfitting. However, training did not always take the full 200 epochs thanks to the early stopping feature, which halted training when no significant improvement in performance was observed over a set number of epochs. The Adam optimizer was chosen for its adaptive learning rate capabilities, which can improve convergence speed and stability.

Table 3. TabNet parameters.

Width of the Decision Prediction Layer	Width of the Attention Embedding for Each Mask	Number of Steps in the Architecture
8	8	3
16	16	6
32	32	9

Varying the width of this layer (8, 16, 32) allows us to investigate the impact of model capacity on performance. A wider layer can capture more complex patterns but may also increase the risk of overfitting. Similar to the decision layer, varying the width of the attention embedding (8, 16, 32) helps us understand how the model's attention mechanism scales with complexity. Wider embeddings can capture more detailed feature interactions. The number of steps (3, 6, 9) determines how many sequential decision and attention layers the data pass through. More steps can improve model performance by allowing more complex transformations but at the cost of increased computational requirements.

Training and validation were conducted with various normalization techniques. We explored two distinct approaches to **label normalization**, designated as NONE, and MINMAX:

- **NONE** refers to the absence of label normalization. Models are trained and assessed using the unaltered labels, which vary from 64 bytes (minimum flow size) to 3,218,210,994 bytes (maximum flow size).
- **MINMAX** is a transformation where its minimum value becomes 0, its maximum value becomes 1, and all other values are scaled proportionally to fall within the range of 0 to 1. The procedure is detailed in Equation (1).

Let labels $= \{l_1, l_2, \ldots, l_n\}$, then for each label l_i in labels, the normalized value $T(l_i)$ is defined by:

$$T(l_i) = \frac{l_i - l_{min}}{l_{max} - l_{min}} \quad (1)$$

3.6. Model

The TabNet [7] model is a type of neural network architecture designed specifically for tabular data. Developed by researchers at Google Cloud AI, TabNet uses sequential attention mechanisms to selectively choose which features to process at each decision step, effectively enabling the model to make decisions based on important, learned features from the data. This selective feature processing allows TabNet to interpret and learn from the data in a way similar to how decision trees isolate important features but with the added flexibility and power of a neural network. TabNet's design also promotes interpretable decision-making, which is a valuable attribute for applications requiring transparency in how input features affect predictions. This model has been shown to perform competitively on various benchmark datasets, outperforming traditional ensemble models like random forests and gradient-boosting machines in some cases.

3.7. Model Decision

In regression analysis, the algorithm predicts a continuous outcome, which, for our study, corresponds to the anticipated flow size in bytes. To illustrate the relationship between flow table reduction and traffic coverage, retraining and refitting the model repeatedly is unnecessary. We can simulate decision-making adjustments by modifying the threshold for classifying a flow as an elephant based on its predicted size. Here, the term *label* denotes the true flow size as extracted from the dataset.

3.8. Evaluation

Current research in the field largely neglects metrics essential for assessing the effectiveness of flow-based traffic engineering. Many studies emphasize the accuracy of flow classification, measuring success through parameters such as the true positive rate, true negative rate, and precision in predicting flow size and duration. Yet, these metrics offer limited insights into the practical implementation of algorithms in this research area. Crucially, the misclassification of a network's largest flow disproportionately affects overall traffic coverage compared to the misclassification of smaller flows. **The metrics commonly used in existing literature fail to capture this significant disparity. Apart from the metrics, it is also unknown which information is the most important for predicting which flow belongs to which class**.

To bridge these gaps, we introduce new metrics specifically designed to evaluate ML models in the context of detecting elephant flows for traffic engineering purposes. We employ two particular metrics for this evaluation: **the reduction in the number of flow table entries created** and **the percentage of traffic covered**. These metrics are intended to provide a more relevant assessment of how well the models perform in practical traffic management scenarios, focusing on optimizing network efficiency and capacity utilization.

It is important to understand the inherent trade-off between these metrics. Increasing the threshold for elephant flow detection results in a larger reduction in the number of flow table entries, but it also diminishes the percentage of traffic that is covered. Striking the right balance between these factors is crucial for optimizing network efficiency and maintaining high QoS.

Additionally, we assess the **information contained in the input 5-tuple (entropy) and its significance in influencing the output predictions' (feature importance)**. Our analysis explores the impact of information across the two proposed input data approaches. This study aims to provide a more in-depth understanding of which elements of the input data are more relevant than others.

4. Results

Out of 504 distinct results (two input data types, three dataset ratios, seven TabNet hyperparameter combinations, three batch size and learning rate combinations, two loss functions, and two normalization types) we selected the best result per input data type

and dataset ratio. In this research, the best means the largest flow table reduction at 80% traffic coverage.

4.1. Flow Table Reduction vs. Traffic Coverage

The visual representations illustrate the reduction in flow table operations and achieved traffic coverage. Remarkably, the y-axis exhibits a logarithmic scale. On the y-axis, each unit corresponds to a multiplier (e.g., 1000 indicates a reduction by a factor of 1000, resulting in the number of created flow table entries being 1/1000 of its original value). The goal is to minimize creation of individual flow entries while preserving optimal traffic coverage. A model is deemed more effective as its curve approaches the top-right corner of the graph.

The black line, identified as **Data**, illustrates the projected performance derived from the validation dataset, comprising 1,303,496 flows. This projection is predicated on the assumption of perfect prediction of each flow's size on its initial packet. This methodology, described in [35] as the *first* method, involves selecting the smallest subset of the largest flows, arranged by size in descending order, which collectively represent a predetermined percentage of the total network traffic.

Figures 2–4 present results for bit vector input data representation and the balanced dataset with ratios of 10%, 20%, and 100%, whereas Figures 5–7 present results for octet input data representation and the balanced dataset with ratios of 10%, 20%, and 100%. Additionally, as seen in Figures 4 and 7 we were unable to draw the reduction vs. coverage result for the MAE with MINMAX normalization type, due to the fact that obtained results did not fit in the traffic coverage area of interest (50–100%). It seems that the model in these configurations was extremely underfitted, and it was not able to sufficiently recognize trends and patterns based on the input data.

In Table 4, we presented the five top-performing configurations. The table illustrates how varying training configurations can impact the effectiveness of TabNet models in reducing created flow entries number while maintaining constant 80% traffic coverage.

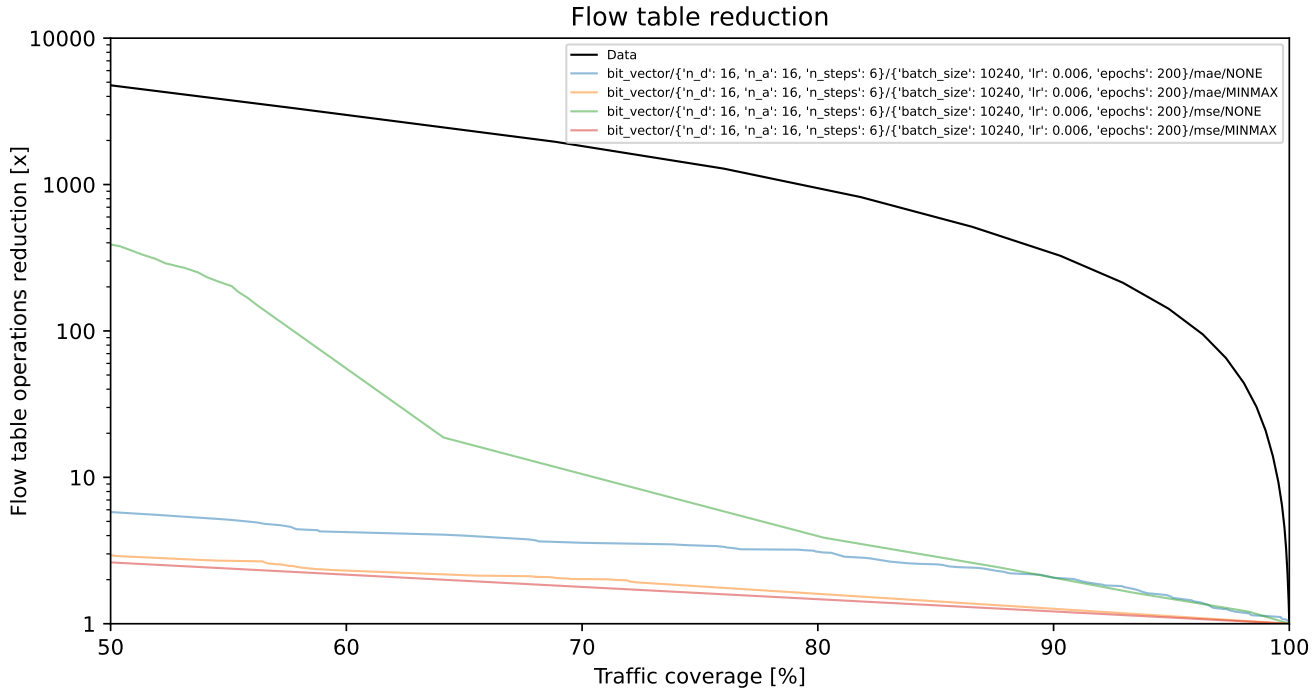

Figure 2. Flow table reduction for the balanced dataset with *10% ratio* and *bit vector* input data.

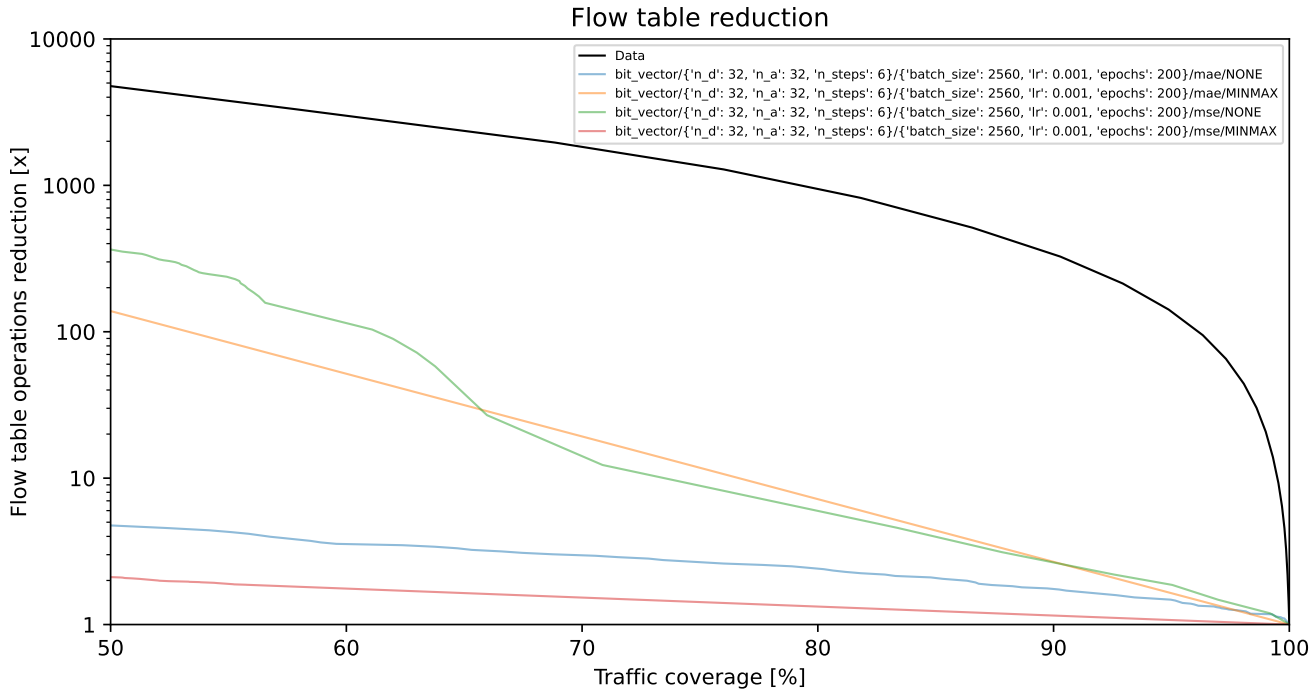

Figure 3. Flow table reduction for the balanced dataset with *20% ratio* and *bit vector* input data.

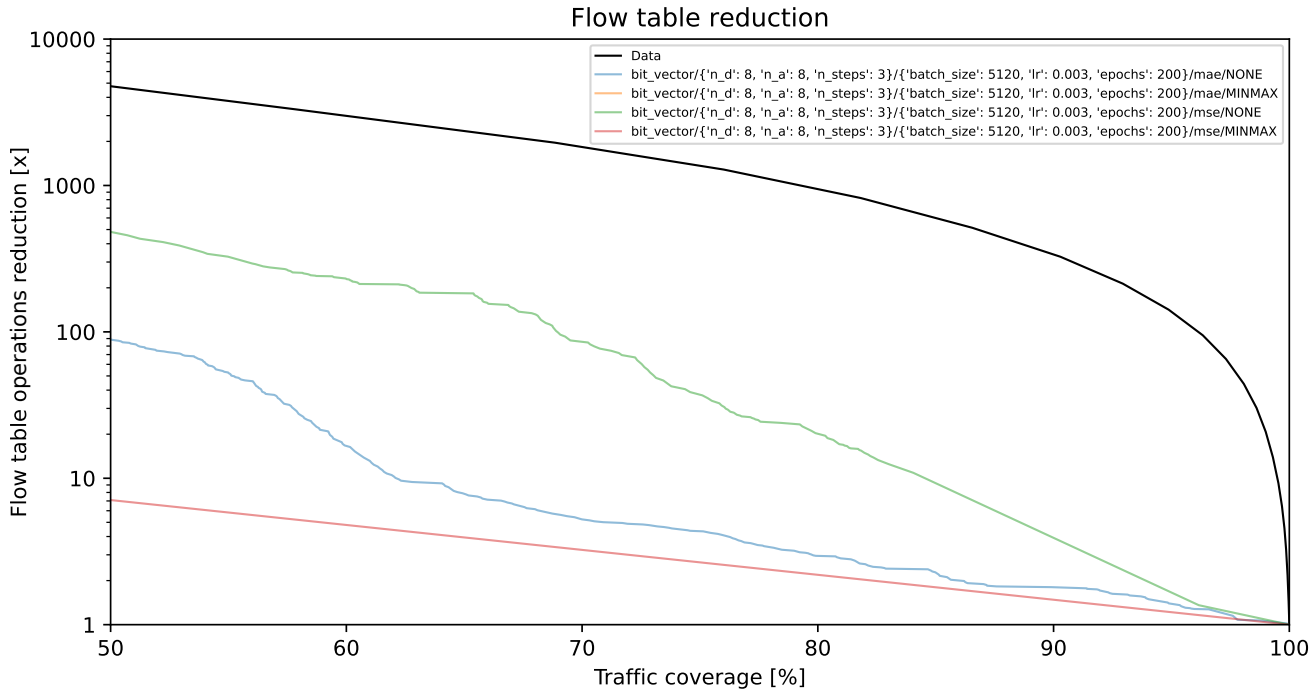

Figure 4. Flow table reduction for the unbalanced dataset (*100% ratio*) and *bit vector* input data.

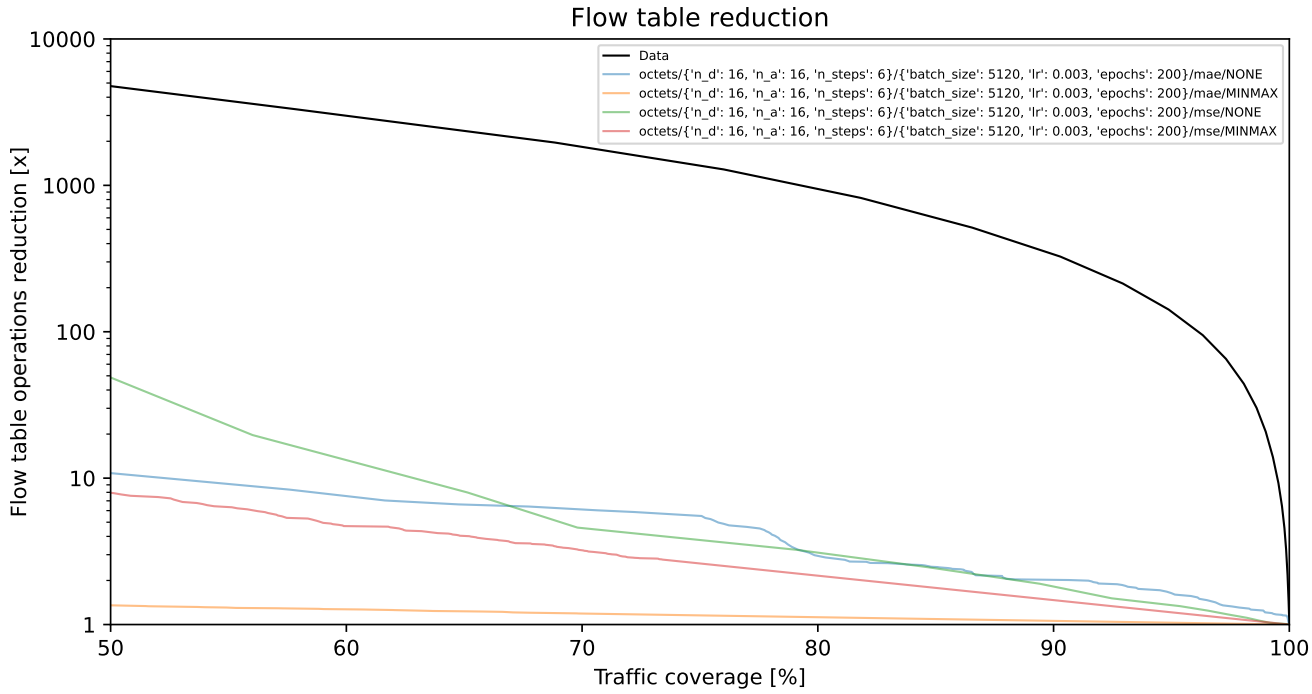

Figure 5. Flow table reduction for the balanced dataset with *10% ratio* and *octets* input data.

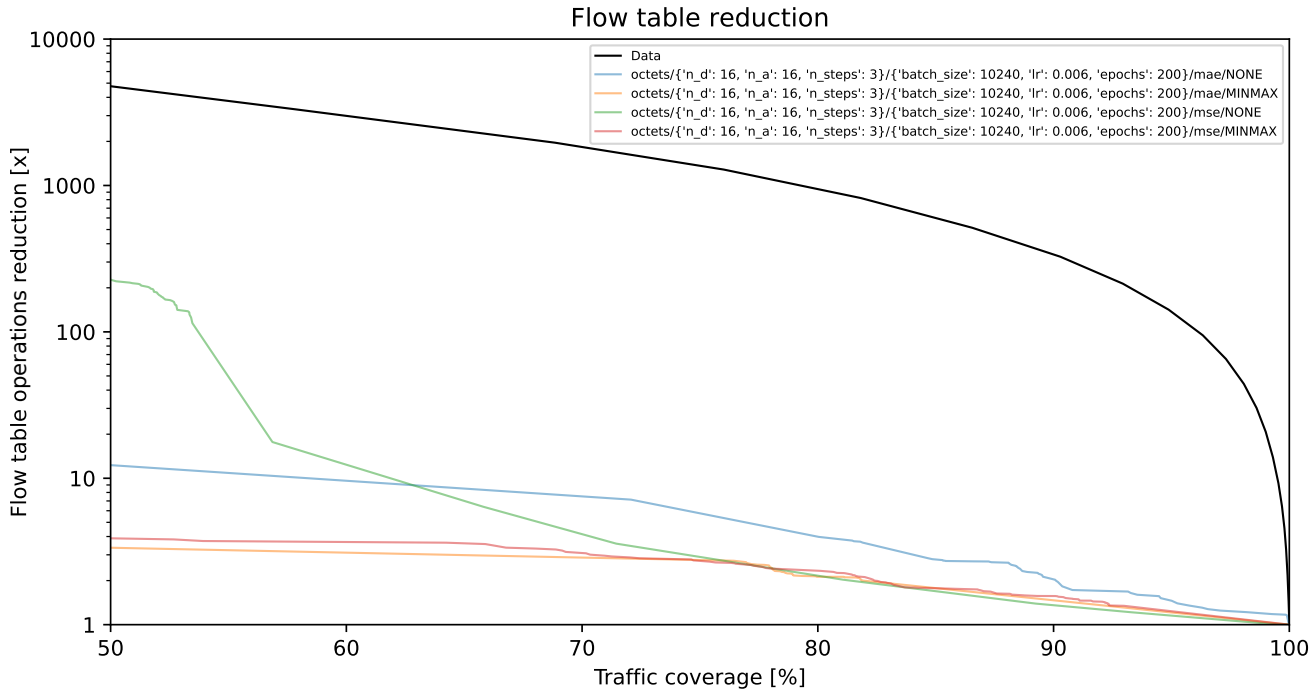

Figure 6. Flow table reduction for the balanced dataset with *20% ratio* and *octets* input data.

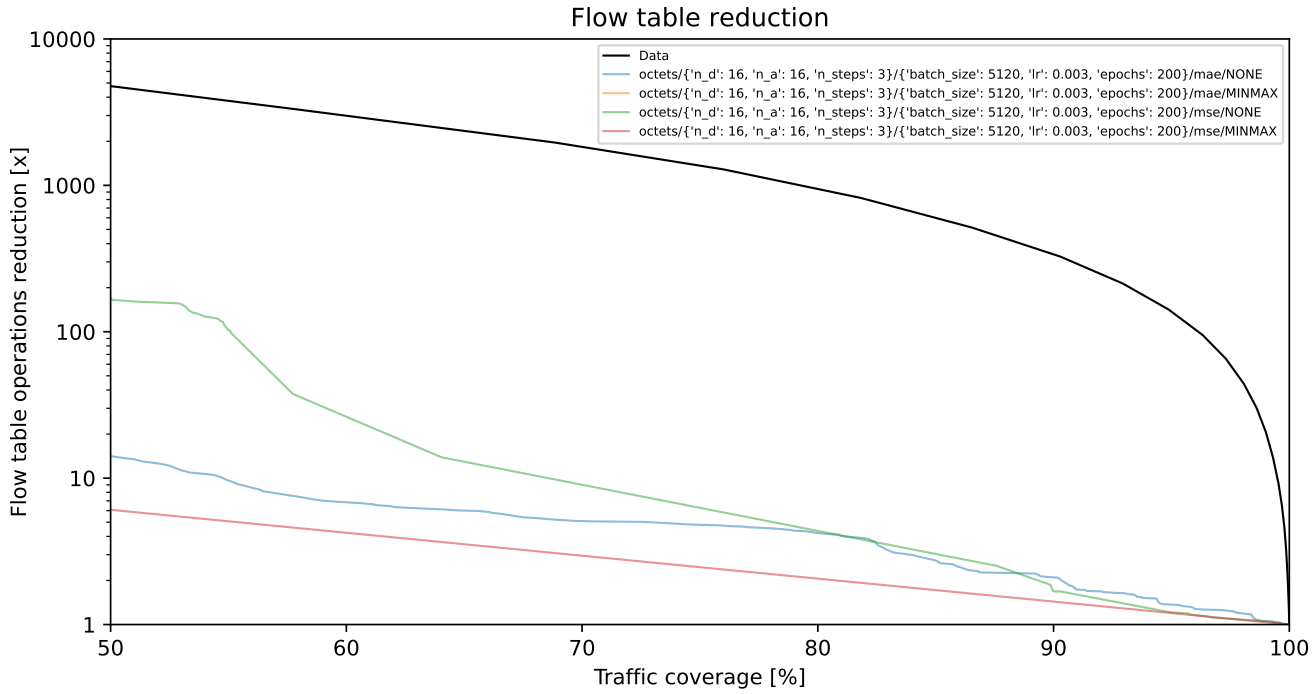

Figure 7. Flow table reduction for the unbalanced dataset (*100% ratio*) and *octets* input data.

Table 4. Top 5 results and hyperparameter values for the 80% traffic coverage.

Input	Dataset Ratio	Model Parameters	Normalization	Loss Function	Batch Size	Learning Rate	Flow Table Reduction (Best)	Flow Table Reduction (SD)
bits	100%	8/8/3	NONE	MSE	5120	0.003	20.14	5.32
bits	100%	16/16/3	NONE	MSE	2560	0.001	18.74	3.79
bits	100%	8/8/3	NONE	MSE	2560	0.001	14.25	2.98
bits	10%	32/32/3	NONE	MSE	10,240	0.006	12.92	2.61
bits	20%	16/16/3	NONE	MSE	2560	0.001	12.59	2.51

4.2. Feature Entropy and Importance

To provide additional insight into which features are most essential for providing an accurate prediction, we performed an analysis of the information amount contained in the input 5-tuple (feature entropy) and its significance in influencing the model (feature importance). The results of the analysis are presented in Figures 8 and 9.

Feature importance analysis was performed for the all input data variations and all dataset balancing ratios. In entropy analysis, we calculated the entropy for both input data representations. In this context, entropy measures the average amount of information contained in a feature (byte or bit, depending on the input data representation). Higher entropy indicates greater randomness, while lower entropy indicates less varied values. We express the entropy in bits. This tells us how many bits on average are needed to encode the information contained in a particular feature.

Header feature bit number	Header field	Entropy: Octets [bits]	Entropy: Bit vector [bits]	Feature importance: Octets 10% ratio	Feature importance: Octets 20% ratio	Feature importance: Octets 100% ratio	Feature importance: Bit vector 10% ratio	Feature importance: Bit vector 20% ratio	Feature importance: Bit vector 100% ratio
0	Source IP address	6.57	0.99	0.02	0.00	0.00	0.00	0.00	0.00
1		6.57	0.99	0.02	0.00	0.00	0.00	0.00	0.00
2		6.57	0.99	0.02	0.00	0.00	0.00	0.00	0.00
3		6.57	0.96	0.02	0.00	0.00	0.00	0.00	0.00
4		6.57	0.97	0.02	0.00	0.00	0.00	0.00	0.00
5		6.57	0.99	0.02	0.00	0.00	0.00	0.00	0.00
6		6.57	0.98	0.02	0.00	0.00	0.00	0.00	0.00
7		6.57	0.98	0.02	0.00	0.00	0.00	0.01	0.00
8		6.05	0.96	0.15	0.01	0.04	0.00	0.01	0.00
9		6.05	0.89	0.15	0.01	0.04	0.00	0.00	0.00
10		6.05	0.76	0.15	0.01	0.04	0.00	0.08	0.00
11		6.05	0.89	0.15	0.01	0.04	0.00	0.00	0.00
12		6.05	0.93	0.15	0.01	0.04	0.00	0.00	0.00
13		6.05	0.90	0.15	0.01	0.04	0.00	0.00	0.00
14		6.05	0.92	0.15	0.01	0.04	0.01	0.00	0.00
15		6.05	0.97	0.15	0.01	0.04	0.01	0.00	0.00
16		7.20	0.88	0.02	0.00	0.00	0.00	0.00	0.00
17		7.20	0.90	0.02	0.00	0.00	0.01	0.00	0.00
18		7.20	0.93	0.02	0.00	0.00	0.00	0.00	0.00
19		7.20	0.99	0.02	0.00	0.00	0.01	0.00	0.00
20		7.20	1.00	0.02	0.00	0.00	0.00	0.01	0.00
21		7.20	1.00	0.02	0.00	0.00	0.00	0.00	0.00
22		7.20	1.00	0.02	0.00	0.00	0.00	0.00	0.00
23		7.20	1.00	0.02	0.00	0.00	0.00	0.00	0.00
24		5.85	0.79	0.01	0.11	0.04	0.00	0.05	0.00
25		5.85	0.84	0.01	0.11	0.04	0.00	0.00	0.00
26		5.85	0.80	0.01	0.11	0.04	0.00	0.00	0.00
27		5.85	0.95	0.01	0.11	0.04	0.02	0.00	0.00
28		5.85	0.94	0.01	0.11	0.04	0.00	0.00	0.00
29		5.85	0.90	0.01	0.11	0.04	0.01	0.00	0.00
30		5.85	0.94	0.01	0.11	0.04	0.00	0.00	0.00
31		5.85	0.93	0.01	0.11	0.04	0.00	0.00	0.00
32	Destination IP address	4.71	0.80	0.03	0.00	0.01	0.00	0.00	0.00
33		4.71	0.83	0.03	0.00	0.01	0.00	0.01	0.00
34		4.71	0.83	0.03	0.00	0.01	0.00	0.00	0.00
35		4.71	0.84	0.03	0.00	0.01	0.01	0.00	0.00
36		4.71	0.80	0.03	0.00	0.01	0.00	0.00	0.00
37		4.71	0.83	0.03	0.00	0.01	0.00	0.01	0.00
38		4.71	0.77	0.03	0.00	0.01	0.00	0.01	0.00
39		4.71	0.79	0.03	0.00	0.01	0.00	0.00	0.00
40		4.24	0.83	0.19	0.00	0.00	0.00	0.00	0.00
41		4.24	0.78	0.19	0.00	0.00	0.00	0.00	0.00
42		4.24	0.79	0.19	0.00	0.00	0.00	0.00	0.00
43		4.24	0.81	0.19	0.00	0.00	0.00	0.00	0.00
44		4.24	0.79	0.19	0.00	0.00	0.00	0.00	0.00
45		4.24	0.80	0.19	0.00	0.00	0.05	0.00	0.00
46		4.24	0.86	0.19	0.00	0.00	0.00	0.01	0.00
47		4.24	0.84	0.19	0.00	0.00	0.00	0.02	0.00
48		7.30	0.90	0.19	0.00	0.01	0.00	0.01	0.00
49		7.30	0.91	0.19	0.00	0.01	0.00	0.00	0.00
50		7.30	0.92	0.19	0.00	0.01	0.00	0.00	0.00
51		7.30	0.99	0.19	0.00	0.01	0.01	0.00	0.00
52		7.30	1.00	0.19	0.00	0.01	0.00	0.00	0.00
53		7.30	1.00	0.19	0.00	0.01	0.00	0.00	0.00
54		7.30	1.00	0.19	0.00	0.01	0.00	0.01	0.00
55		7.30	0.99	0.19	0.00	0.01	0.00	0.00	0.00
56		5.95	0.80	0.02	0.07	0.02	0.00	0.00	0.00
57		5.95	0.84	0.02	0.07	0.02	0.00	0.01	0.00
58		5.95	0.82	0.02	0.07	0.02	0.00	0.01	0.00
59		5.95	0.93	0.02	0.07	0.02	0.01	0.00	0.00
60		5.95	0.93	0.02	0.07	0.02	0.00	0.00	0.00
61		5.95	0.91	0.02	0.07	0.02	0.01	0.00	0.00
62		5.95	0.92	0.02	0.07	0.02	0.00	0.01	0.00
63		5.95	0.93	0.02	0.07	0.02	0.00	0.00	0.00

Figure 8. Feature entropy (calculated) and importance (from TabNet) (bits 0–63).

Header feature bit number	Header field	Entropy: Octets [bits]	Entropy: Bit vector [bits]	Feature importance: Octets 10% ratio	Feature importance: Octets 20% ratio	Feature importance: Octets 100% ratio	Feature importance: Bit vector 10% ratio	Feature importance: Bit vector 20% ratio	Feature importance: Bit vector 100% ratio
64	Source port	4.76	0.80	0.10	0.11	0.18	0.00	0.00	0.00
65		4.76	0.83	0.10	0.11	0.18	0.00	0.01	0.44
66		4.76	0.83	0.10	0.11	0.18	0.00	0.08	0.01
67		4.76	0.84	0.10	0.11	0.18	0.00	0.00	0.00
68		4.76	0.81	0.10	0.11	0.18	0.00	0.00	0.00
69		4.76	0.84	0.10	0.11	0.18	0.00	0.00	0.00
70		4.76	0.78	0.10	0.11	0.18	0.00	0.00	0.00
71		4.76	0.78	0.10	0.11	0.18	0.00	0.01	0.00
72		4.32	0.83	0.15	0.38	0.01	0.00	0.00	0.00
73		4.32	0.78	0.15	0.38	0.01	0.00	0.00	0.00
74		4.32	0.79	0.15	0.38	0.01	0.00	0.00	0.00
75		4.32	0.82	0.15	0.38	0.01	0.01	0.00	0.09
76		4.32	0.78	0.15	0.38	0.01	0.00	0.00	0.00
77		4.32	0.80	0.15	0.38	0.01	0.00	0.00	0.00
78		4.32	0.86	0.15	0.38	0.01	0.00	0.01	0.00
79		4.32	0.84	0.15	0.38	0.01	0.00	0.00	0.00
80	Destination port	6.42	0.99	0.08	0.29	0.02	0.00	0.00	0.00
81		6.42	0.98	0.08	0.29	0.02	0.00	0.00	0.00
82		6.42	0.98	0.08	0.29	0.02	0.00	0.00	0.00
83		6.42	0.93	0.08	0.29	0.02	0.00	0.00	0.00
84		6.42	0.98	0.08	0.29	0.02	0.00	0.00	0.00
85		6.42	0.99	0.08	0.29	0.02	0.03	0.00	0.07
86		6.42	0.97	0.08	0.29	0.02	0.00	0.00	0.00
87		6.42	0.97	0.08	0.29	0.02	0.00	0.01	0.00
88		5.76	0.94	0.01	0.01	0.00	0.21	0.00	0.00
89		5.76	0.89	0.01	0.01	0.00	0.00	0.19	0.00
90		5.76	0.74	0.01	0.01	0.00	0.01	0.00	0.00
91		5.76	0.88	0.01	0.01	0.00	0.00	0.00	0.00
92		5.76	0.90	0.01	0.01	0.00	0.01	0.09	0.04
93		5.76	0.87	0.01	0.01	0.00	0.00	0.01	0.00
94		5.76	0.89	0.01	0.01	0.00	0.02	0.00	0.00
95		5.76	0.97	0.01	0.01	0.00	0.12	0.00	0.00
96	Transport protocol	1.11	0.00	0.04	0.01	0.67	0.00	0.08	0.00
97		1.11	0.00	0.04	0.01	0.67	0.15	0.00	0.00
98		1.11	0.00	0.04	0.01	0.67	0.05	0.08	0.02
99		1.11	0.99	0.04	0.01	0.67	0.07	0.00	0.16
100		1.11	0.00	0.04	0.01	0.67	0.02	0.09	0.04
101		1.11	1.00	0.04	0.01	0.67	0.09	0.00	0.12
102		1.11	1.00	0.04	0.01	0.67	0.05	0.00	0.00
103		1.11	1.00	0.04	0.01	0.67	0.00	0.01	0.00

Figure 9. Feature entropy (calculated) and importance (from TabNet) (bits 64–103).

5. Discussion

The flow table reduction results show the superiority of MSE over the MAE loss functions. MSE employs an error amplification mechanism. For larger errors, the squared term magnifies their impact, which accelerates the minimization process during training. MSE amplifies the influence of outliers, which seems to fit our scenario much better than the MAE, which, on the other hand, treats all errors equally, minimizing the impact of outliers on the loss function. Additionally, as can be seen in the reduction results, TabNet worked much better on unnormalized labels rather than normalized labels. Regarding the TabNet parameters, the best results were obtained with the width (both the decision prediction layer and attention embedding for each mask) set to 8. The best reduction rate achieved for the 80% traffic coverage was 20.14. As shown in Figure 10 this is a 25% higher reduction rate than achieved previously with neural networks comprising solely linear layers, which provided only 15-fold reduction for the best parameter combination [36].

Feature entropy analysis shows that the most predictable fields are related to the transport protocol, and source port in both input data representations. This is expected, as the transport protocol field contains mostly one of the two values: 6 for TCP and 17 for UDP. The least predictable (most random) fields are the addresses (both source and destination) and destination port.

As the results show, only a fraction of initial input data is significant for the model in predicting the flow size. Features are also not equally important across dataset balancing ratios. In the *octets* data, different features like transport protocol, ports, and addresses

dominate depending on the ratio. For the 100% ratio, the transport protocol has significantly higher weight than the other features. For the 20% ratio, the source and destination ports are the most important, while for the 10% ratio, the source and destination addresses are the most important. Conversely, in the *bit vector* data, the transport protocol and destination port are consistently important, while the source and destination addresses are not, and the source port's importance declines at lower ratios.

The variation in feature importance across different dataset ratios can be attributed to the nature of the balancing process itself. At higher ratios, where the dataset is more imbalanced, the model may rely heavily on more generalized features such as transport protocols that are universally present in all flows. However, at lower ratios, where the dataset is more balanced, the model can discern more nuanced patterns and dependencies, leading to a higher significance of specific features like source and destination addresses. This deeper exploration reveals that feature importance is inherently tied to the composition and characteristics of the training data, impacting the model's predictive behavior depending on the dataset's balance.

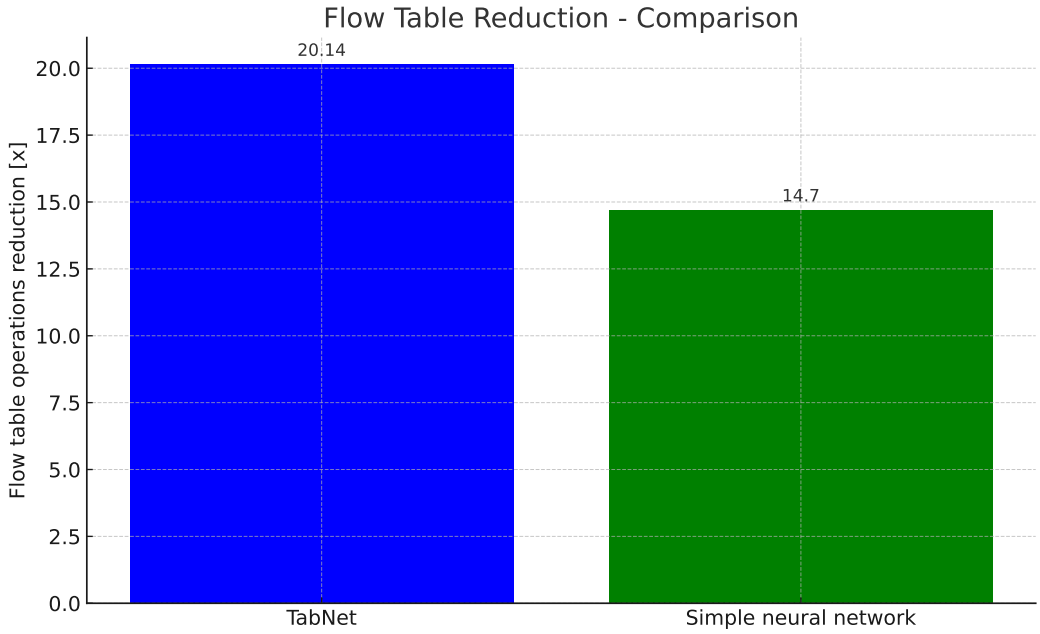

Figure 10. Comparison of TabNet and the best neural network from [36].

6. Conclusions

As demonstrated in this study, employing a TabNet model to identify elephant flows from the initial packets enables a reduction in the number of flow table entries by approximately 20-fold while still encompassing 80% of the traffic. The reduction in number of required flow table entries can not only enable flow-based traffic engineering on switches with limited capacities but also positively influence flow table lookup, consequently enhancing the switching rate. We also evaluated the significance of the information carried by the initial packet 5-tuple. It was determined that only a subset of all features is truly important for the model in providing accurate results. Utilizing this subset of the input data, one can achieve faster training and inference time, which can result in quicker elephant flow classification and minimization of the additional latency.

Author Contributions: Conceptualization, B.K. and P.J.; methodology, P.J.; software, B.K. and P.J.; validation, P.J.; formal analysis, B.K.; investigation, B.K.; resources, P.J.; data curation, P.J.; writing—original draft preparation, B.K.; writing—review and editing, P.J., R.W. and J.D.; visualization, P.J.; supervision, R.W. and J.D.; project administration, R.W. and J.D.; funding acquisition, R.W. and J.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Polish Ministry of Science and Higher Education with the subvention funds of the Faculty of Computer Science, Electronics and Telecommunications of AGH University.

Data Availability Statement: The anonymized input data are available in the GitHub repository: https://github.com/piotrjurkiewicz/flow-models (accessed on 20 June 2024).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jurkiewicz, P.; Rzym, G.; Boryło, P. Flow length and size distributions in campus Internet traffic. *Comput. Commun.* **2021**, *167*, 15–30. [CrossRef]
2. Megyesi, P.; Molnár, S. Analysis of elephant users in broadband network traffic. In Proceedings of the Meeting of the European Network of Universities and Companies in Information and Communication Engineering, Chemnitz, Germany, 28–30 August 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 37–45. [CrossRef]
3. Akyildiz, I.F.; Lee, A.; Wang, P.; Luo, M.; Chou, W. A roadmap for traffic engineering in SDN-OpenFlow networks. *Comput. Networks* **2014**, *71*, 1–30. [CrossRef]
4. Kreutz, D.; Ramos, F.M.V.; Veríssimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
5. Mendiola, A.; Astorga, J.; Jacob, E.; Higuero, M. A Survey on the Contributions of Software-Defined Networking to Traffic Engineering. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 918–953. [CrossRef]
6. Shen, G.; Li, Q.; Ai, S.; Jiang, Y.; Xu, M.; Jia, X. How Powerful Switches Should be Deployed: A Precise Estimation Based on Queuing Theory. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 811–819. [CrossRef]
7. Arik, S.Ö.; Pfister, T. Tabnet: Attentive interpretable tabular learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 2–9 February 2021; Volume 35, pp. 6679–6687. [CrossRef]
8. Shaikh, A.; Rexford, J.; Shin, K.G. Load-sensitive Routing of Long-lived IP Flows. *ACM SIGCOMM Comput. Commun. Rev.* **1999**, *29*, 215–226. [CrossRef]
9. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: Dynamic flow scheduling for data center networks. In Proceedings of the NSDI'10: 7th USENIX Conference on Networked Systems Design and Implementation, San Jose, CA, USA, 28–30 April 2010; p. 19. [CrossRef]
10. Curtis, A.R.; Mogul, J.C.; Tourrilhes, J.; Yalagandula, P.; Sharma, P.; Banerjee, S. DevoFlow: Scaling flow management for high-performance networks. In Proceedings of the ACM SIGCOMM Computer Communication Review, Toronto, ON, Canada, 15–19 August 2011; ACM: New York, NY, USA, 2011; Volume 41, pp. 254–265. [CrossRef]
11. Xu, H.; Huang, H.; Chen, S.; Zhao, G. Scalable software-defined networking through hybrid switching. In Proceedings of the IEEE INFOCOM 2017—IEEE Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9. [CrossRef]
12. Xiao, P.; Qu, W.; Qi, H.; Xu, Y.; Li, Z. An efficient elephant flow detection with cost-sensitive in SDN. In Proceedings of the 2015 1st International Conference on Industrial Networks and Intelligent Systems (INISCom), Tokyo, Japan, 2–4 March 2015; pp. 24–28. [CrossRef]
13. Poupart, P.; Chen, Z.; Jaini, P.; Fung, F.; Susanto, H.; Geng, Y.; Chen, L.; Chen, K.; Jin, H. Online flow size prediction for improved network routing. In Proceedings of the 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 8–11 November 2016; pp. 1–6. [CrossRef]
14. Liu, W.X.; Cai, J.; Wang, Y.; Chen, Q.C.; Zeng, J.Q. Fine-grained flow classification using deep learning for software defined data center networks. *J. Netw. Comput. Appl.* **2020**, *168*, 102766. [CrossRef]
15. Hamdan, M.; Mohammed, B.; Humayun, U.; Abdelaziz, A.; Khan, S.; Ali, M.A.; Imran, M.; Marsono, M.N. Flow-aware elephant flow detection for software-defined networks. *IEEE Access* **2020**, *8*, 72585–72597. [CrossRef]
16. He, H.; Peng, Z.; Zhou, X.; Wang, J. LFOD: A Lightweight Flow Table Optimization Scheme in SDN Based on Flow Length Distribution in the Internet. In Proceedings of the 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), Takamatsu, Japan, 28–30 September 2022; pp. 1–6. [CrossRef]
17. Qian, Z.; Gao, G.; Du, Y. Per-Flow Size Measurement by Combining Sketch and Flow Table in Software-Defined Networks. In Proceedings of the 2022 IEEE International Conference on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking (ISPA/BDCloud/SocialCom/SustainCom), Melbourne, Australia, 17–19 December 2022; pp. 644–651. [CrossRef]
18. da Silva, M.V.B.; Jacobs, A.S.; Pfitscher, R.J.; Granville, L.Z. Predicting Elephant Flows in Internet Exchange Point Programmable Networks. In Proceedings of the International Conference on Advanced Information Networking and Applications, Matsue, Japan, 27–29 March 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 485–497. [CrossRef]
19. Brito da Silva, M.V.; Schaeffer-Filho, A.E.; Granville, L.Z. HashCuckoo: Predicting Elephant Flows using Meta-Heuristics in Programmable Data Planes. In Proceedings of the GLOBECOM 2022—2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; pp. 6337–6342. [CrossRef]

20. Pekar, A.; Duque-Torres, A.; Seah, W.K.; Caicedo Rendon, O.M. Towards threshold-agnostic heavy-hitter classification. *Int. J. Netw. Manag.* **2022**, *32*, e2188. [CrossRef]
21. Coelho, B.L.; Schaeffer-Filho, A.E. CrossBal: Data and Control Plane Cooperation for Efficient and Scalable Network Load Balancing. In Proceedings of the 2023 19th International Conference on Network and Service Management (CNSM), Niagara Falls, ON, Canada, 30 October–2 November 2023; pp. 1–9. [CrossRef]
22. Wassie, G.; Ding, J.; Wondie, Y. Traffic prediction in SDN for explainable QoS using deep learning approach. *Sci. Rep.* **2023**, *13*, 20607. [CrossRef] [PubMed]
23. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the KDD '16: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; pp. 785–794. [CrossRef]
24. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
25. Durner, R.; Kellerer, W. Network Function Offloading Through Classification of Elephant Flows. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 807–820. [CrossRef]
26. Hardegen, C.; Pfülb, B.; Rieger, S.; Gepperth, A. Predicting Network Flow Characteristics Using Deep Learning and Real-World Network Traffic. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 2662–2676. [CrossRef]
27. Hardegen, C.; Pfülb, B.; Rieger, S.; Gepperth, A.; Reißmann, S. Flow-based Throughput Prediction using Deep Learning and Real-World Network Traffic. In Proceedings of the 2019 15th International Conference on Network and Service Management (CNSM), Halifax, NS, Canada, 21–25 October 2019; pp. 1–9. [CrossRef]
28. Gómez, J.; Riaño, V.H.; Ramirez-Gonzalez, G. Traffic Classification in IP Networks Through Machine Learning Techniques in Final Systems. *IEEE Access* **2023**, *11*, 44932–44940. [CrossRef]
29. Xie, S.; Hu, G.; Xing, C.; Liu, Y. Online Elephant Flow Prediction for Load Balancing in Programmable Switch-Based DCN. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 745–758. [CrossRef]
30. Alkhalidi, N.A.; Yaseen, F.A. FDPHI: Fast Deep Packet Header Inspection for Data Traffic Classification and Management. *Int. J. Intell. Eng. Syst.* **2021**, *14*, 373–383. [CrossRef]
31. Yaseen, F.A.; Alkhalidi, N.A.; Al-Raweshidy, H.S. She networks: Security, health, and emergency networks traffic priority management based on ml and sdn. *IEEE Access* **2022**, *10*, 92249–92258. [CrossRef]
32. Jurkiewicz, P. Flow-models: A framework for analysis and modeling of IP network flows. *SoftwareX* **2022**, *17*, 100929. [CrossRef]
33. Jurkiewicz, P. flow-models 2.0: Elephant flows modeling and detection with machine learning. *SoftwareX* **2023**, *24*, 101506. [CrossRef]
34. Xu, J.; Fan, J.; Ammar, M.; Moon, S.B. On the Design and Performance of Prefix-Preserving IP Traffic Trace Anonymization. In Proceedings of the IMW '01: 1st ACM SIGCOMM Workshop on Internet Measurement, New York, NY, USA, 1–2 November 2001; pp. 263–266. [CrossRef]
35. Jurkiewicz, P. Boundaries of Flow Table Usage Reduction Algorithms Based on Elephant Flow Detection. In Proceedings of the 2021 IFIP Networking Conference (IFIP Networking), Espoo and Helsinki, Finland, 21–24 June 2021; pp. 1–9. [CrossRef]
36. Kądziołka, B.; Jurkiewicz, P.; Wójcik, R.; Domżał, J. Elephant Flow Classification on the First Packet with Neural Networks. *IEEE Access* **2024**, *12*, 65298–65309. [CrossRef]