IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 April 2024, accepted 29 April 2024, date of publication 8 May 2024, date of current version 15 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3398065

RESEARCH ARTICLE

Elephant Flow Classification on the First Packet With Neural Networks

BARTOSZ KĄDZIOŁKA, PIOTR JURKIEWICZ, ROBERT WÓJCIK, AND JERZY DOMŻAŁ
Institute of Telecommunications, AGH University of Krakow, 30-059 Kraków, Poland
Corresponding author: Bartosz Kądziołka (kadziolka@agh.edu.pl)

ABSTRACT Quick and accurate identification of the largest flows in the network would allow for the management of most traffic using dedicated, flow-specific routes and policies, thereby significantly reducing the overall number of entries in switch flow tables. Our analysis focuses on utilizing neural networks to classify elephant flows based on the first packet using 5-tuple packet header fields. The findings indicate that with simple neural networks comprising solely linear layers, it is possible to accurately detect elephant flows at their inception, thereby reducing the number of flow table entries – by up to a factor of 15 – while still effectively covering 80% of the network traffic with individual flow entries.

INDEX TERMS Flows, flow table, elephant, optimization, mice, traffic engineering, machine learning.

I. INTRODUCTION

Elephant flows, a term used to describe the largest data flows on the Internet, are commonly believed to carry the majority of the traffic. Yet, they pose a only small share of the overall flow number. In contrast, there are numerous *mice flows*, which, despite their abundance, contribute only a small fraction of the overall traffic. This distribution is even more skewed than the typical 80/20 split suggested by the Pareto principle. Recent research indicates that as little as 0.2-0.4% of all data flows may be responsible for up to 80% of the total Internet traffic, as shown in studies such as [1] and [2].

Recently, flow-based traffic engineering has gained traction as an effective method to manage the ever-growing network demand while maintaining Quality of Service (QoS) unaffected. In this method, a distinctive forwarding entry is allocated for every individual flow within the switch's memory. Each entry specifies the next hop in the flow's route, facilitating the utilization of diverse routes for flows originating from the same source-destination pairs. This capability enables the implementation of multipath routing. The selection of routes for new flows can be adjusted according to the current or anticipated network load, adaptively bypassing overloaded links. Additionally, such flow-based adaptive routing offers more stability than traditional dynamic load balancing [3].

The challenge with flow-based traffic engineering is that the number of simultaneous flows in a network often surpasses the capacity of flow tables in switches [4]. Additionally, in centralized software-defined networks, the controller's throughput for managing new flows is limited. Besides the capacity limitation, a smaller number of entries also accelerates table lookup and thus packet switching rate. A potential solution to this issue is to create individual entries solely for the largest flows. The majority of smaller flows could be therefore routed via the standard, shortest paths. This would notably decrease the number of entries in the flow tables, yet it would still allow to management of a significant portion of the traffic with dedicated, flow-specific entries.

Identifying the largest flows presents a significant challenge. Our goal is to detect them early enough to quickly allocate individual entries, ensuring that the majority of their packets will follow specific routing paths. Optimally, flows should be classified right from their first packet, to prevent the need for rerouting during the connection. Such an approach can rely only on data contained in packet headers and cannot utilize information about packet sequences, like packet sizes or interarrival times. Consequently, we investigate the application of neural networks to identify large flows based on the 5-tuples (protocol, source and destination IP addresses, and ports) data found in packet headers.

The associate editor coordinating the review of this manuscript and approving it for publication was Maurizio Casoni.

We evaluate the performance of several neural networks built with the help of the PyTorch library. A key aspect of our paper is its emphasis on metrics that are critical for traffic engineering in Software-Defined Network (SDN). Specifically, we concentrate on the amount of traffic carried by flows identified as elephants after their detection (referred to as *traffic coverage*), and the subsequent decrease in the number of individual flow entries required in the tables (which we refer to as *flow table occupancy reduction*). We evaluate six models of neural networks along with five different approaches to label normalization.

II. RELATED WORK

The concept of individually managing only elephant flows is not new. It was first proposed in 1999 with the idea of adaptive routing of significant data streams [5]. However, due to hardware limitations of that time, it remained an academic concept. It has seen a resurgence in popularity, especially with the emergence of software-defined networking. In this modern context, a controller equipped with extensive network knowledge can effectively handle large flows.

Hedera traffic engineering system, introduced in [6], was designed to reroute flows through an embedded controller once they exceed a certain threshold. This rerouting directs the flows along dynamically chosen paths. It operates under the assumption that edge devices gather all flow statistics using OpenFlow counters, focusing optimization efforts on non-edge devices. Another notable flow-based system is DevoFlow [7]. DevoFlow prioritizes elephant flows, employing sampling methods and threshold values for identification. However, its efficiency assessment is based solely on the aggregated flow of the entire network. A system akin to DevoFlow, detailed in [8], detects elephant flows at edge devices using a modified Bloom filter. This study's traffic model, which assumes that 20% of flows contribute to 80% of traffic, significantly deviates from actual patterns, as indicated by more recent studies [1], [2].

In the studies mentioned, basic techniques like sampling, counters, and threshold values are used for detecting large flows. Recently, more advanced machine-learning based systems were proposed. A decision tree model for identifying elephant flows was implemented and evaluated in [9], focusing mainly on the *accuracy* of detection. However, as we argue in this paper, accuracy might not be the most representative factor. In [10] Poupart et al. evaluated the effectiveness of three distinct Machine Learning (ML) methods in predicting flow size and classifying it as an elephant flow. Their dataset comprised three million flows, encompassing both TCP and UDP traffic. The key metrics in their analysis were the percentage of correctly identified large flows (*true positive rate*) and the percentage of correctly identified small flows (*true negative rate*).

In [11] Liu et al. suggest utilizing a random forest decision tree to identify eight key features for a classification model. They propose a two-tiered architecture consisting of pre-classification at the edge devices of the SDN network and precise classification at the central controller. The study introduces a novel classification system, categorizing flows into four types: elephant, cheetah, tortoise, and porcupine. The research focuses on the accuracy of classification and the delay involved in the classification process.

In the study [12] Hamdan et al. also examine a two-level classification architecture, with initial sorting done at the switches and the final classification at the central controller. Here, switches differentiate between mice and potential elephant flows using the count-min sketch algorithm, while a decision tree in the controller performs the final classification. Notably, the switch algorithm is periodically updated with a new dataset from the controller. This study employed real traffic models for algorithm validation, but like [11], it primarily emphasized classifier precision. A single-level count-min sketch-based lightweight flow table optimization scheme was proposed in 2022 by He et al. [13]. Another sketch-based mechanism was also proposed in 2022 by Qian et al. in [14]. Different from common sketches, they utilize TCAM-based flow table to store elephant flow labels so as to solve the contradiction between the record of elephant flow labels and the accuracy of mouse flows. Notably, in both papers real ISP packet traces were used to evaluate the proposed mechanisms.

In another study, da Silva et al. [15] proposed a Locally Weighted Regression (LWR) algorithm for predicting the size and duration of a new flow, utilizing patterns from previous flows and their immediate correlation with the new flow. Another system proposed in 2022 by the same author uses hashing mechanism based on the Cuckoo Search meta-heuristic [16]. There is also a recent proposal of a threshold-agnostic heavy-hitter classification system is [17] by Pekar et al. Similarly, the system uses template matching to classify elephant flows according to the per-flow packet size distribution observed in initial packets.

CrossBal, a hybrid load balancing system based on Deep Reinforcement Learning (DRL) that focuses its efforts on high-impact elephant flows was proposed in [18]. The system uses three-level elephant flow detection, which include threshold-based preliminary filtering. Flowlets of detected elephant flows are then rerouted to achieve efficent netowrk load-balancing. Deep learning system was also proposed by Wassie et al. in [19]. It uses deep autoencode and gradient boosting, with autoML predicting algorithms, including XGBoost, GBM.

All the studies mentioned above concentrate on flow classification after observing several initial packets. In contrast, our primary objective is to identify a flow as rapidly as possible, ideally from its first packet. Durner et al. in [20] demonstrated flow classification on the first packet, using only features extracted from the 5-tuple (src IP, dst IP, src port, dst port, protocol) and the size of the first packet. In [21] Hardegen et al. suggested multiclass prediction as an alternative to binary classification (elephant/mouse), employing a deep neural network to predict flow characteristics from the 5-tuple

of the first packet. A similar approach was used in [22] for predicting a flow's bit rate from the 5-tuple of its first packet.

The recent 2023 work in this area by Gomez et al. [23] focuses on classification immediately after a flow's first packet. However, this study, like the others, primarily evaluates classification accuracy, without considering the impact on the number of entries in the flow table and achievable traffic coverage. The most recent paper from 2024 by Xie et al. [24] on the other hand proposes 2-stage decision tree based system. The first stage of elephant flow pre-classification is performed solely using the data contained in the first packet header. What is notable, the authors implemented their system in the P4 language, however they verified it only in a switch emulator, not on a real device.

There are also recent works which incorporate neural networks to classify network flows, but for the QoS, rather than traffic engineering purposes. In [25] Alkhalidi et al. propose a one-dimensional convolutional neural network to classify flows into several classes based on the packet header. Interesting novelty is the proposal of automatic selection of only selected bits of packets headers, which can reduce the number of features, and thus the processing time and energy consumption, at the same time giving satisfactory accuracy. In [26] Yaseen et al. propose usage of a similar system to classify traffic and assign a DSCP field. The system has been implemented within an SDN controller and evaluated in the Mininet emulator in an emergency traffic prioritization scenario.

Although the mentioned studies focus on flow classification, they emphasize classification accuracy, overlooking the practical effectiveness of these algorithms for traffic engineering objectives. For example, incorrectly classifying the largest flow in a network significantly affects traffic coverage, more so than misclassifying a smaller flow. The metrics used in these studies fail to recognize this disparity. In particular, none of the previous research has examined metrics such as the reduction in flow table entries or the volume of traffic handled post-classification. These factors are vital for evaluating switches and controller load and the impact on traffic engineering and overall system performance.

III. METHODOLOGY

Prediction of flow size based on its initial packet is feasible using a type of ML known as regression. Regression is one of the main supervised learning methods, requiring labeled input data to enable the model to classify or predict. All neural network models, presented in this paper, were built with PyTorch [27].

A. DATA PREPARATION

The performance of any ML algorithm is heavily dependent on the dataset used. In our evaluation, we rely on data regarding the length and size distributions of flows from a dataset gathered over 30 days in a large campus network [1]. For data processing, we utilized the package detailed in [28].

1) DATASET

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 [29]. 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. As demonstrated in [29], this anonymization process does not affect the performance of the ML models.

2) INPUT FEATURES

The input data derived from the 5-tuple comprises the following details: source IP address, destination IP address, source port, destination port, and transport layer protocol, collectively accounting for 104 bits. We explore one particular representation of this input data:

- **bits**: Each field in the header is divided into individual bits, creating 104 distinct features. These are represented as binary values (0/1).

The dataset was partitioned into training and validation sets, comprising 80% and 20% of the data, respectively. This distribution implies that the training sets included 5,213,988 flows, while the validation sets contained 1,303,496 flows.

B. BALANCING DATASET

Balancing the training dataset played a crucial role. In the training dataset which contains 5,213,988 flows the number of mice flows significantly outnumbers the elephant flows. To achieve satisfactory accuracy it was necessary to resolve this issue. Every result presented in this paper was achieved with the model being trained on the balanced dataset with different ratios. Balancing the dataset was achieved in the following steps:

1) Set the *ratio* – for instance 10%.
2) Calculate *balanced dataset size* – size of the initial training dataset * ratio (5,213,988 * 10% = 521,398 flows).
3) Sort the initial training dataset in a descending order – largest flows at the beginning.
4) Get the first half of the *balanced dataset size* number of flows from the beginning of the ordered list from the previous step.
5) The second half of the *balanced dataset size* number of flows is selected randomly from the rest of the initial dataset.

TABLE 1. Variable hyperparameters.

Batch Size	Learning Rate
1280	1e-3
2560	3e-3
5120	6e-3

TABLE 2. Constant hyperparameters.

Epochs	Optimizer	Loss Function
100	Adam	Mean Absolute Error

C. TRAINING

This phase included hyperparameters selection, labels normalization and the training of neural network. The flowchart of training phase is presented in the Figure III-C3.

1) HYPERPARAMETERS

Each model was trained on the shuffled, balanced dataset. Then, its performance was evaluated on the validation data. We performed training and validation with a couple of hyperparameter combinations. Hyperparameters that varied are shown in the Table 1, meanwhile, hyperparameters which remained constant across all training sessions are presented in the Table 2.

The general rule is that with the bigger batch size, one can obtain more stable training – it reduces the chance for the model to be overtrained. Together with batch size scaling we decided to scale the learning rate. This resulted in the model being able to faster find the local minima and maxima without the need to scale a number of epochs adequately.

2) LABELS NORMALIZATION

Every training and validation was performed with different types of normalization. In total, we analyzed 5 different approaches to label normalization. We refer to the normalization types as NONE, ONE CLIP, ONE NO CLIP, SQUARED CLIP, and SQUARED NO CLIP:

- NONE normalization type means that there is no normalization of the labels. We train and evaluate the model on the original labels. Original labels range from 64 (smallest size of the flow) to 3,218,210,994 bytes (biggest size of the flow).
- ONE CLIP means that every label in both training and validation datasets is divided by one gigabyte (1,000,000,000). After division values from the datasets are clipped to range 0 – 1. This means that every value smaller than 0 becomes 0, and, every value larger than 1 becomes 1. This is expressed by the Equation 1.
- ONE NO CLIP similarly to ONE CLIP divides the datasets by one gigabyte. However, in this case, values outside the interval are not clipped to the interval edges. This is expressed by the Equation 2.
- SQUARED CLIP means that every label in both training and validation datasets is squared and divided by one gigabyte. The result is clipped to the range 0 – 1. This is expressed by the Equation 3.
- SQUARED NO CLIP means that every label in both training and validation datasets is squared and divided by one gigabyte. This is expressed by the Equation 4.

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:

$$T(l_i) = \begin{cases} 0 & \text{if } \frac{l_i}{1000000} < 0, \\ 1 & \text{if } \frac{l_i}{1000000} > 1, \\ \frac{l_i}{1000000} & \text{otherwise.} \end{cases} \tag{1}$$

$$T(l_i) = \frac{l_i}{1000000} \tag{2}$$

$$T(l_i) = \begin{cases} 0 & \text{if } \frac{l_i^2}{1000000} < 0, \\ 1 & \text{if } \frac{l_i^2}{1000000} > 1, \\ \frac{l_i^2}{1000000} & \text{otherwise} \end{cases} \tag{3}$$

$$T(l_i) = \frac{l_i^2}{1000000} \tag{4}$$

3) DECISION (MOUSE/ELEPHANT)

In regression analysis, the output of the algorithm is a continuous value. In our case, it is the predicted size of the flow in bytes. To generate a curve of flow table reduction in the function of traffic coverage, it is not necessary to repeatedly undergo the training and fitting process. Instead, we can simulate the decision-making by adjusting the elephant threshold on the predicted flow sizes. In this context, a label refers to the actual flow size obtained from the dataset.

D. EVALUATION

Current research in the field largely overlooks metrics crucial for evaluating the effectiveness of flow-based traffic engineering. Instead, most studies focus on the accuracy of flow classification, gauged by parameters such as the true positive rate, true negative rate, and precision in predicting flow size and duration. However, these metrics alone provide limited insight into the practical application of the algorithms in our area of research. Importantly, the misclassification of the network's largest flow has a far greater impact on overall traffic coverage compared to the misclassification of smaller flows. The metrics used in existing literature do not adequately address this significant difference.

Addressing these gaps, we propose new metrics to evaluate ML models in the context of detecting elephant flows for traffic engineering. We used two specific metrics for the evaluation: **the reduction in flow table occupancy** and **the percentage of traffic covered**.

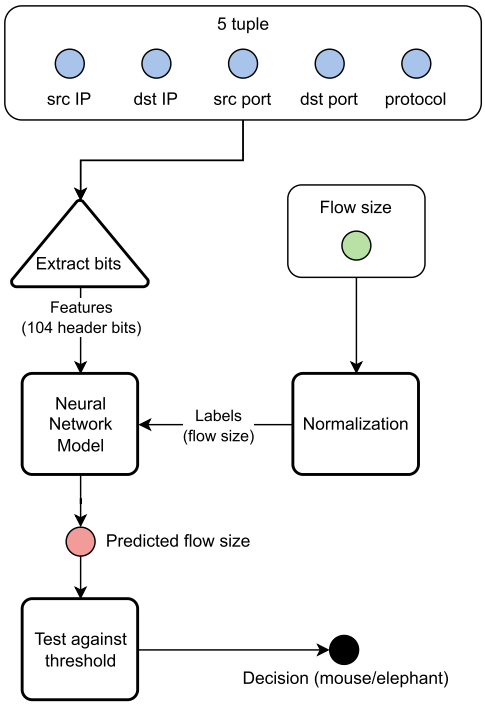

FIGURE 1. Training of the neural network.

It's crucial to recognize the inherent tradeoff between these metrics. Raising the threshold for elephant flow detection leads to a greater reduction in the number of flow table entries, but at the same time, it reduces the fraction of traffic that is covered. This balance is key to optimizing both network efficiency and the QoS.

E. IMPLEMENTATION CONSIDERATIONS

Although this paper focuses on an isolated problem of elephant flow classification, in this section we present considerations of how the proposed neural network model can be used in a traffic engineering system. This is just an example of an implementation – several different approaches can be considered. Moreover, the proposed model can be used in other applications and is not limited only to TE systems.

Figure 2 presents a flowchart of switch dataplane operation. Such packet processing pipeline can be implemented for example in a P4 switch. Each incoming packet (which includes the first packets of new flows, but also subsequent packets of existing flows) is looked up against the flow table to find a corresponding flow entry. The purpose of the flow table is to store full flow entries for elephant flows, which may include the next hops of a path individually assigned for that flow and other flow-specific QoS attributes. When the entry is present, the packet is forwarded according to the next hop assigned for that flow.

When there is no flow entry in the flow table, the hash of the packet's 5-tuple is looked up against a Bloom filter. The role of the Bloom filter is to be a cache of seen (and already classified) mice flows. When the flow is present in the Bloom

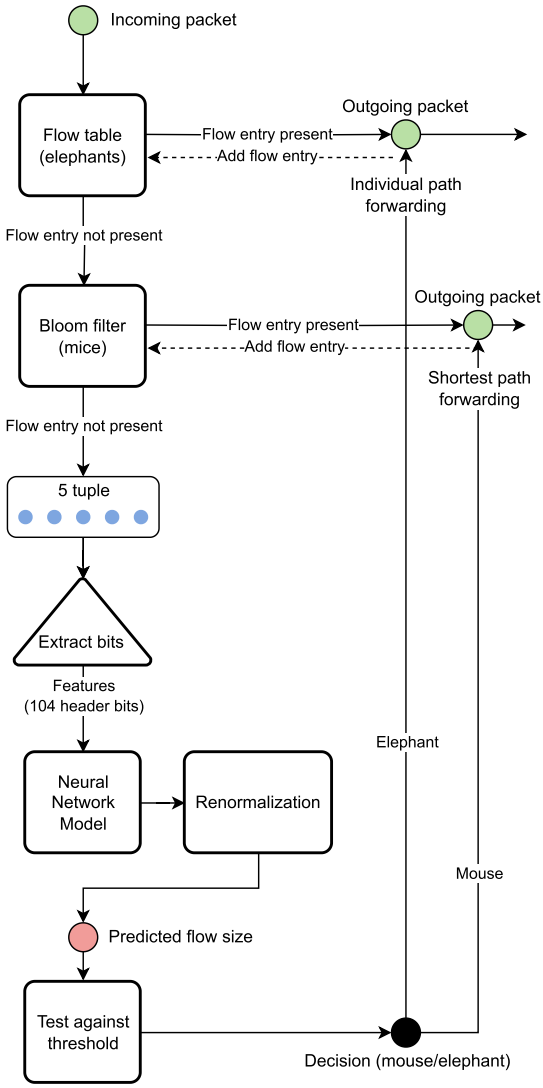

FIGURE 2. Example of an implementation in a TE system.

TABLE 3. Model names.

Model	Name
(a)	FirstRegression
(b)	FallingRegression
(c)	GrowingRegression
(d)	ThreeLayerNeuralNetworkRegression
(e)	FourLayerNeuralNetworkRegression
(f)	TenLayerNeuralNetworkRegression

filter, it means that it is a mouse flow. The packet is then forwarded according to the shortest paths.

If the flow is not present in the Bloom filter, it means that the incoming packet is the first packet of a new flow and the flow needs to be classified. First, the feature bits are extracted from the packet's header 5-tuple. Then they are fed to the neural network model which was earlier trained offline based on previous traffic observations. The regression model predicts a normalized flow size, which is then renormalized.

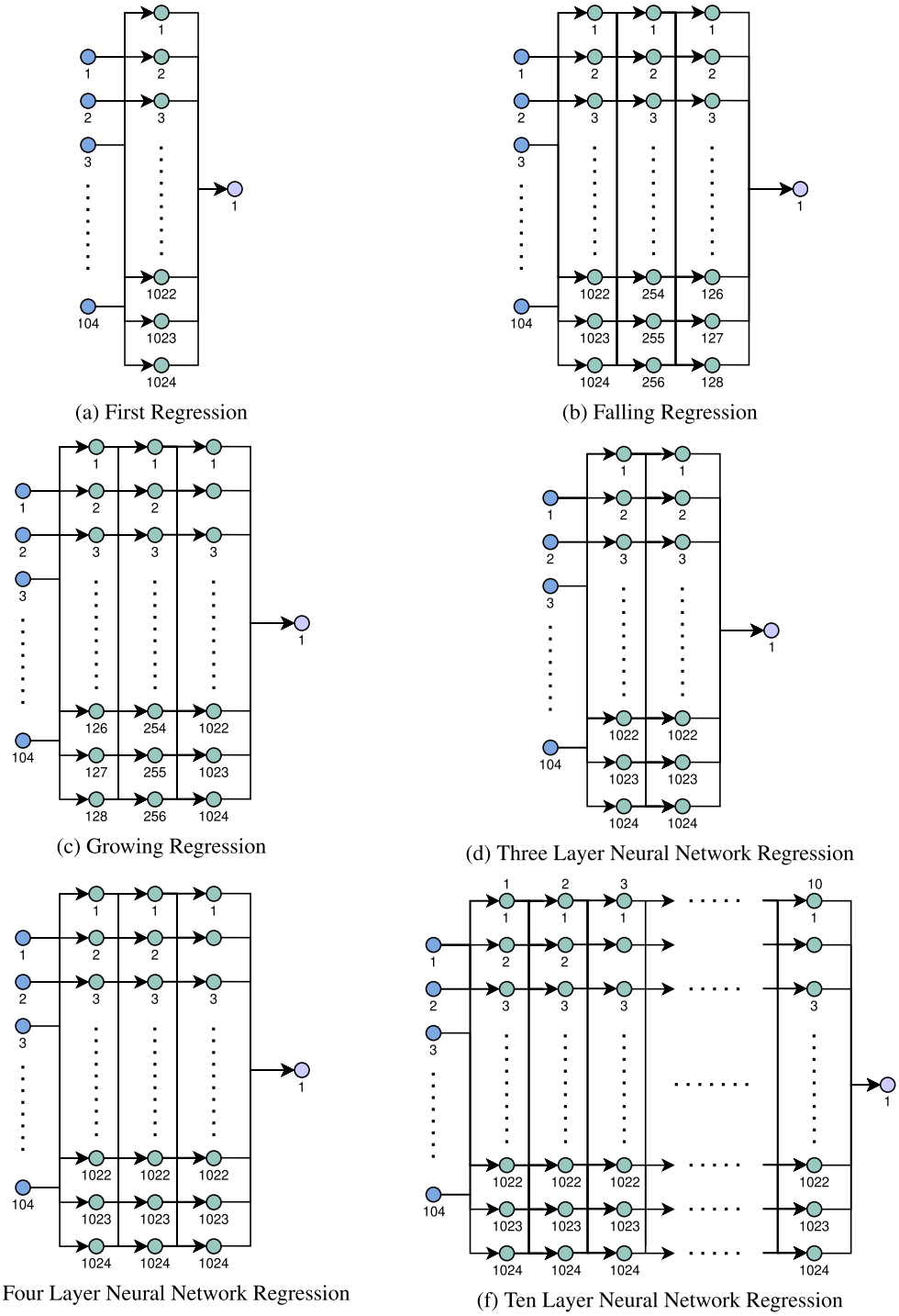

FIGURE 3. Neural network models.

The predicted flow size is then tested against the elephant size threshold, which can be tuned by the network administrator according to the current needs. Finally, the incoming flow is classified either as a mouse or an elephant.

Flows classified as elephants are added to the flow table along with an individual path. The path may be selected locally from the set of pre-determined alternative paths. The packet can be also forwarded to the controller to perform load-aware selection based on the global view of the network. Subsequent packets of that flow will be forwarded according to the flow table entry and not again classified.

For flows classified as mice, the hash of 5-tuple is calculated. Then the hash is added to the Bloom filter. This

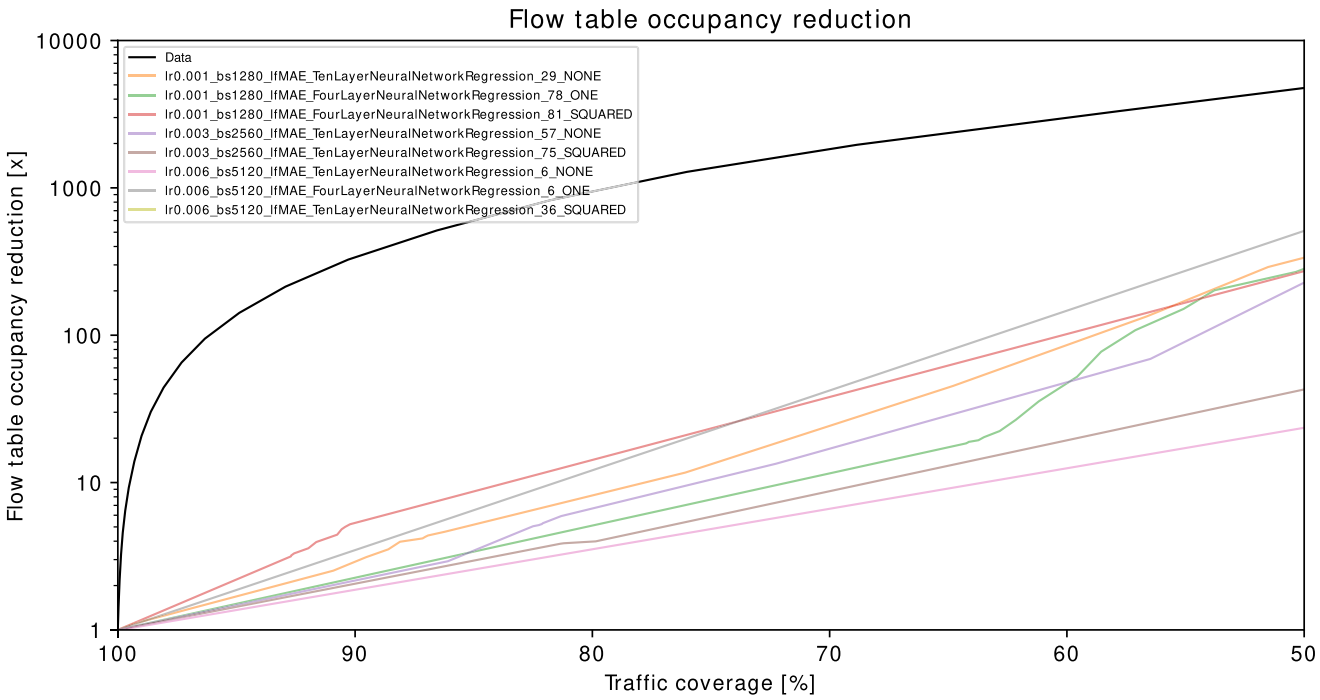

FIGURE 4. Flow table reduction for the balanced dataset with 6% ratio and *clipping* in the normalization.

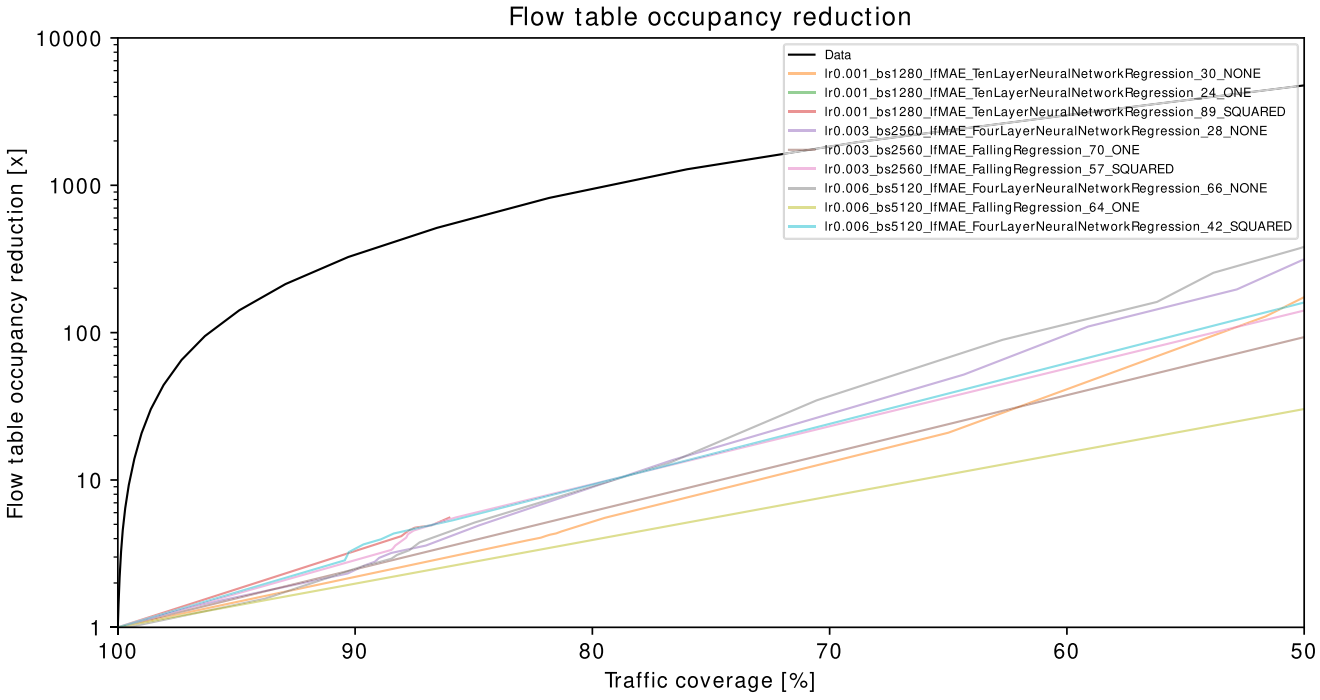

FIGURE 5. Flow table reduction for the balanced dataset with 10% ratio and *clipping* in the normalization.

will prevent subsequent packets of already classified mice flows from being again classified. However, there are possible false positive matches with Bloom filters. That means that new flows, which were not yet classified, may be considered as already classified mice flows. Such flows will still be forwarded, but using shortest paths, as in the classic routing. This will reduce the effectiveness of the TE system, but will not break its operation. Moreover, the probability of false positives can be strictly controlled by adjusting the Bloom filter size.

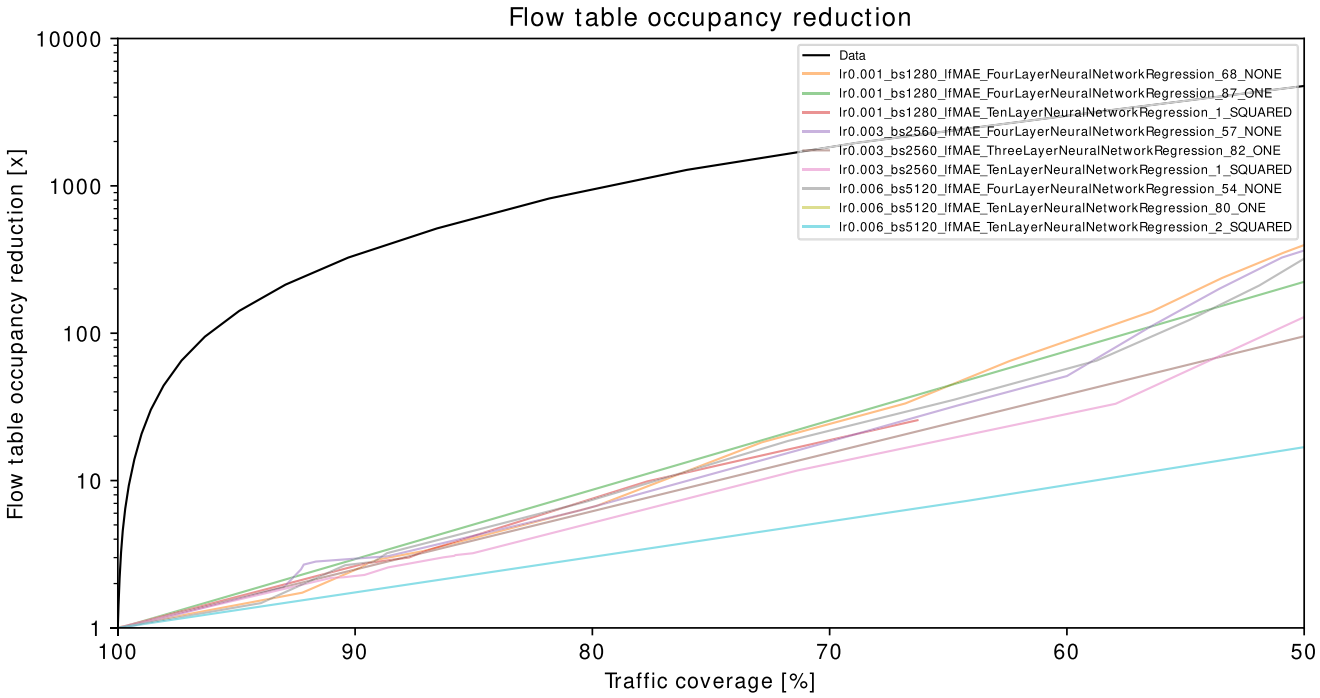

FIGURE 6. **Flow table reduction for the balanced dataset with 20% ratio and *clipping* in the normalization.**

Some kind of aging and cleanup for entries in the Bloom filter will also need to be implemented. Exemplary design can include a rotating ring of Bloom subfilters. For example, in the case of a 2-second inactive flow timeout and 10 subfilters, each subfilter will encompass 200 ms of traffic. All subfilters will be looked up in parallel to find an entry, but new entries will be added only to the subfilter currently at the first position. The ring will be shifted by one every 200 ms and the subfilter on the last position will be zeroed after each shift. On a hit for an already existing flow, an entry will also need to be added to the current first subfilter to ensure that only entries for inactive flows will be subject to aging and cleanup.

IV. MODELS

Models analyzed in this research were built from scratch using the PyTorch library. The difference between those models lies only in the number of layers and number of neurons. The building blocks of the models are the same – all models are built on top of the Linear layers and Rectified Linear Activation Function (ReLU). Additionally, all models are fully connected feedforward models with the same number of neurons in the input and output layers. As it was mentioned earlier, the input layer covers 104 features and the output layer represents the predicted flow size, which is a single output. Model names are presented in the Table 3.

Figure 3a presents FirstRegression architecture – it is a neural network with 104 neurons in the input layer, one hidden layer with 1024 neurons, and an output layer with a single neuron.

FallingRegression is presented in Figure 3b. It has 3 hidden layers where the first hidden layer has 1024 neurons, the second hidden layer has 256 neurons and the last hidden layer has 128 neurons.

GrowingRegression is a reverted FallingRegression. Its first hidden layer has 128 neurons, the second 256 neurons and the third has 1024 neurons. It is presented in the Figure 3c.

ThreeLayerNeuralNetworkRegression, FourLayerNeuralNetworkRegression, and TenLayerNeuralNetworkRegression share the same number of neurons in hidden layers – 1024. They differ in the number of hidden layers. The architectures of those models are presented in Figures 3d,3e, and 3f.

V. RESULTS

The graphs depict how flow table occupancy changes in relation to traffic coverage. Notably, **the x-axis is inverted, and the y-axis uses a logarithmic scale**. In the y-axis the [x] axis unit is a multiplier (for example 1000 means that we obtained reduction by a factor of 1000, so the flow table occupancy is 1/1000 of the initial one). The objective here is to maximize flow table occupancy reduction while maintaining optimal traffic coverage. The closer the curve approaches the top-left corner of the graph, the more effective the model is.

The black line, labeled as **Data**, represents the projected performance based on a distribution fit to the validation dataset, which consists of 1,303,496 flows. This is under the assumption of perfect prediction of each flow's size

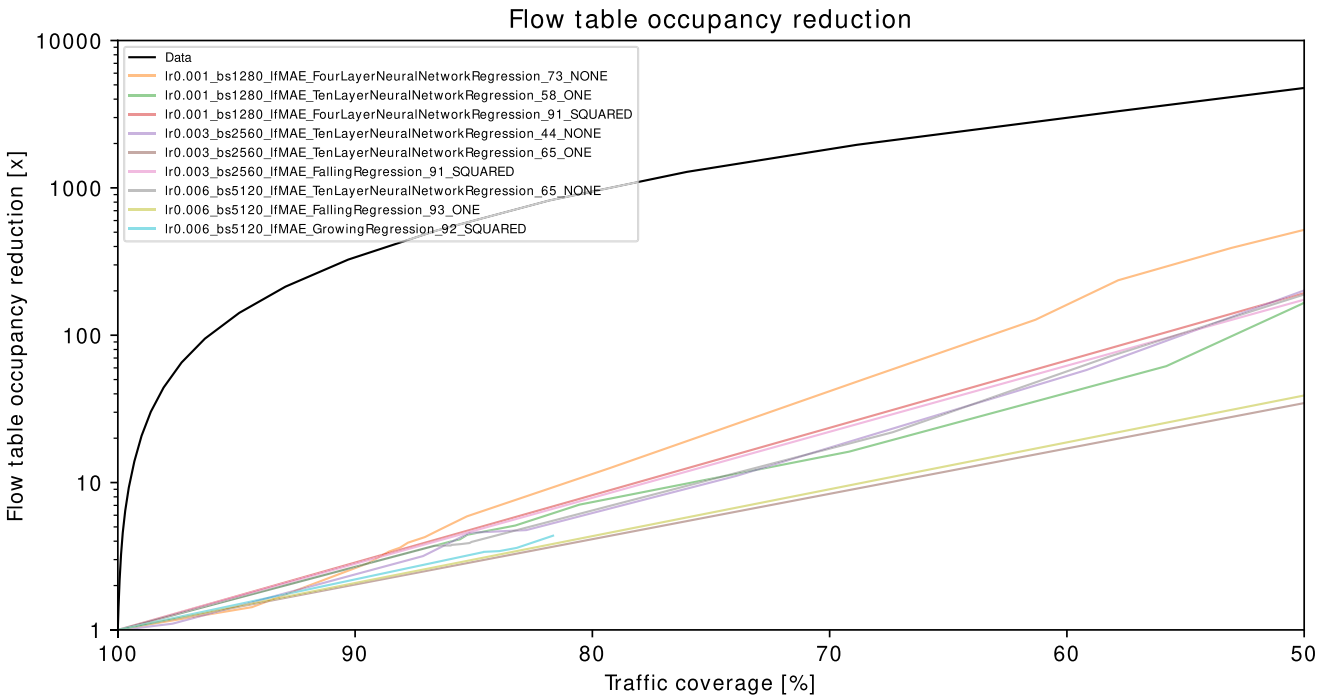

FIGURE 7. Flow table reduction for the balanced dataset with 6% ratio and *no clipping* in the normalization.

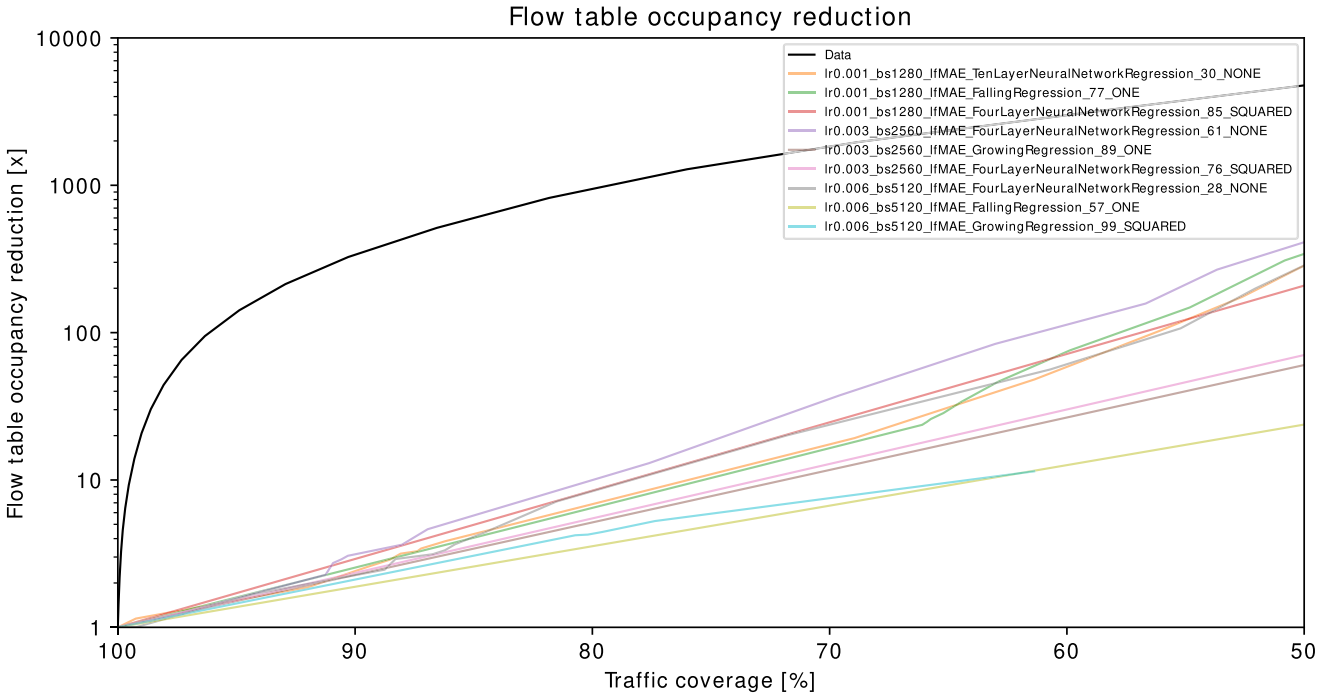

FIGURE 8. Flow table reduction for the balanced dataset with 10% ratio and *no clipping* in the normalization.

from its first packet. The benchmark for assessment is the theoretical flow table occupancy reduction rate curves of a model that can perfectly predict the size of all flows from their first packet. This approach is detailed in [30] as the *first* method. The process involves selecting the smallest number of largest flows, ordered by size in descending order, which cumulatively accounts for a predetermined percentage of total network traffic.

Figures 4, 5, and 6 present results for normalization types with clipping to the desired range, whereas Figures 7, 8, and 9

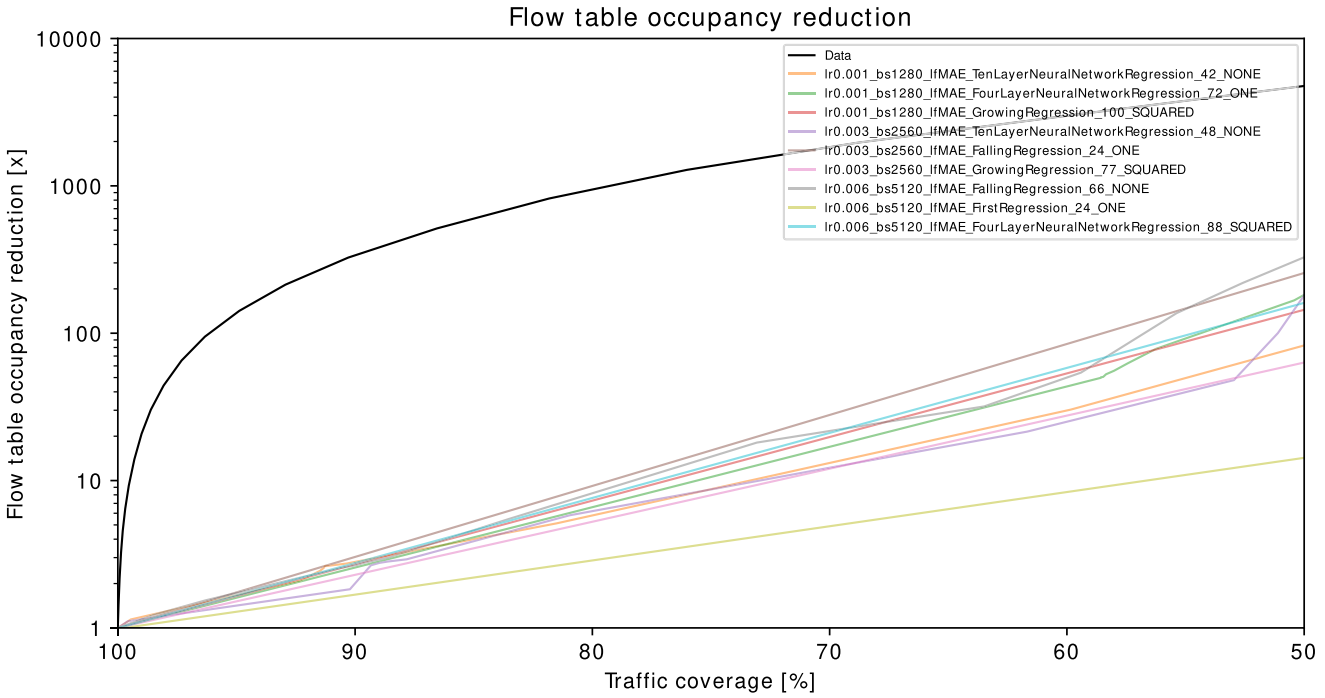

FIGURE 9. Flow table reduction for the balanced dataset with 20% ratio and *no clipping* in the normalization.

TABLE 4. Best results.

Dataset ratio	Model	Normalization	Batch size	Learning rate	Epoch	**Flow table reduction**	**Traffic coverage**
6%	FourLayerNeuralNetworkRegression	SQUARED CLIP	1280	0.001	81	14.7	80%
10%	FallingRegression	SQUARED CLIP	2560	0.003	57	9.1	80%
20%	FourLayerNeuralNetworkRegression	ONE CLIP	1280	0.001	87	9.3	80%
6%	FourLayerNeuralNetworkRegression	NONE	1280	0.001	73	11.7	80%
10%	FourLayerNeuralNetworkRegression	NONE	2560	0.003	61	10.8	80%
20%	FallingRegression	ONE NO CLIP	2560	0.003	24	9.0	80%

present results for normalization types without clipping to the desired range. In every figure within the best hyperparameter combination, the best model and training epoch were selected. For example, the appearance of TenLayerNeuralNetworkRegression_29_NONE curve means that without labels normalization the TenLayerNeuralNetworkRegression model at epoch 29 had the best performance.

As can be seen, in some cases, the result does not reach 50% of traffic coverage. Especially for the SQUARED normalization type, the model predicts negative values which after renormalization become zero's. In this scenario, data points received from the flow table reduction algorithm very often do not reach 50% of traffic coverage.

VI. DISCUSSION

In summary, Table 4 illustrates how varying training configurations can impact the effectiveness of machine learning models in reducing flow table sizes while maintaining consistent traffic coverage. For every dataset ratio, within the best hyperparameter combination, the best model and training epoch were selected. They are presented in the Table 4. The use of different normalization techniques seems to have a noticeable impact on the flow table reduction metric, with SQUARED CLIP generally leading to higher reduction percentages, especially when combined with the FourLayerNeuralNetworkRegression model and a 6% dataset ratio. Increasing the dataset ratio from 6% to 20% does not linearly improve the flow table reduction, suggesting that simply increasing the amount of training data is not sufficient to enhance model performance in terms of this metric.

VII. CONCLUSION

As presented in this research, using rather simple neural networks based only on linear layers to detect elephant flows on the first packets, it is possible to reduce the number of flow table entries approximately 15-fold while still covering 80% of the traffic. Such a reduction of flow table can have positive impact on the flow table lookup, therefore increasing the switching rate. We plan to further research more complex, state-of-the-art models with multiple parameter variations and tune them specifically for the given task. The starting point will be TabNet [31].

Future research on reducing flow table size should also consider the significance of input data, particularly the critical features that contribute to accurate elephant flow classification. Identifying these key features will allow us to streamline the model's complexity by eliminating irrelevant input data features. Simplifying the model in this way leads to faster training and evaluation times. This is particularly advantageous as it facilitates quicker elephant flow classification, thereby minimizing the introduction of additional latency in the network. Such efficiency in model operation is essential for maintaining optimal network performance and responsiveness.

DATA AVAILABILITY

The anonymized input data is available in the GitHub repository: https://github.com/piotrjurkiewicz/flow-models

REFERENCES

[1] P. Jurkiewicz, G. Rzym, and P. Boryło, "Flow length and size distributions in campus Internet traffic," *Comput. Commun.*, vol. 167, pp. 15–30, Feb. 2021.

[2] P. Megyesi and S. Molnár, "Analysis of elephant users in broadband network traffic," in *Proc. Meeting Eur. Netw. Universities Companies Inf. Commun. Eng.* Springer, 2013, pp. 37–45.

[3] P. Jurkiewicz, R. Wójcik, J. Domżał, and A. Kamisiński, "Testing implementation of FAMTAR: Adaptive multipath routing," *Comput. Commun.*, vol. 149, pp. 300–311, Jan. 2020.

[4] G. Shen, Q. Li, S. Ai, Y. Jiang, M. Xu, and X. Jia, "How powerful switches should be deployed: A precise estimation based on queuing theory," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 811–819.

[5] A. Shaikh, J. Rexford, and K. G. Shin, "Load-sensitive routing of long-lived IP flows," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 215–226, Oct. 1999.

[6] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, San Jose, CA, USA, 2010, vol. 10, no. 8, pp. 89–92.

[7] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *Proc. ACM SIGCOMM Conf.*, 2011, vol. 41, no. 4, pp. 254–265.

[8] H. Xu, H. Huang, S. Chen, and G. Zhao, "Scalable software-defined networking through hybrid switching," in *Proc. IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.

[9] P. Xiao, W. Qu, H. Qi, Y. Xu, and Z. Li, "An efficient elephant flow detection with cost-sensitive in SDN," in *Proc. 1st Int. Conf. Ind. Netw. Intell. Syst. (INISCom)*, Mar. 2015, pp. 24–28.

[10] P. Poupart, Z. Chen, P. Jaini, F. Fung, H. Susanto, Y. Geng, L. Chen, K. Chen, and H. Jin, "Online flow size prediction for improved network routing," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–6.

[11] W.-X. Liu, J. Cai, Y. Wang, Q. C. Chen, and J.-Q. Zeng, "Fine-grained flow classification using deep learning for software defined data center networks," *J. Netw. Comput. Appl.*, vol. 168, Oct. 2020, Art. no. 102766.

[12] M. Hamdan, B. Mohammed, U. Humayun, A. Abdelaziz, S. Khan, M. A. Ali, M. Imran, and M. N. Marsono, "Flow-aware elephant flow detection for software-defined networks," *IEEE Access*, vol. 8, pp. 72585–72597, 2020.

[13] H. He, Z. Peng, X. Zhou, and J. Wang, "LFOD: A lightweight flow table optimization scheme in SDN based on flow length distribution in the Internet," in *Proc. 23rd Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2022, pp. 1–6.

[14] Z. Qian, G. Gao, and Y. Du, "Per-flow size measurement by combining sketch and flow table in software-defined networks," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2022, pp. 644–651.

[15] M. V. B. da Silva, A. S. Jacobs, R. J. Pfitscher, and L. Z. Granville, "Predicting elephant flows in Internet exchange point programmable networks," in *Proc. 33rd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*. Springer, 2019, pp. 485–497.

[16] M. V. Brito da Silva, A. E. Schaeffer-Filho, and L. Z. Granville, "HashCuckoo: Predicting elephant flows using meta-heuristics in programmable data planes," in *Proc. IEEE Global Commun. Conf.*, Dec. 2022, pp. 6337–6342.

[17] A. Pekar, A. Duque-Torres, W. K. G. Seah, and O. M. Caicedo Rendon, "Towards threshold-agnostic heavy-hitter classification," *Int. J. Netw. Manage.*, vol. 32, no. 3, p. e2188, May 2022.

[18] B. L. Coelho and A. E. Schaeffer-Filho, "CrossBal: Data and control plane cooperation for efficient and scalable network load balancing," in *Proc. 19th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2023, pp. 1–9.

[19] G. Wassie, J. Ding, and Y. Wondie, "Traffic prediction in SDN for explainable QoS using deep learning approach," *Sci. Rep.*, vol. 13, no. 1, Nov. 2023.

[20] R. Durner and W. Kellerer, "Network function offloading through classification of elephant flows," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 2, pp. 807–820, Jun. 2020.

[21] C. Hardegen, B. Pfülb, S. Rieger, and A. Gepperth, "Predicting network flow characteristics using deep learning and real-world network traffic," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2662–2676, Dec. 2020.

[22] C. Hardegen, B. Pfülb, S. Rieger, A. Gepperth, and S. Reißmann, "Flow-based throughput prediction using deep learning and real-world network traffic," in *Proc. 15th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2019, pp. 1–9.

[23] J. Gómez, V. H. Riaño, and G. Ramirez-Gonzalez, "Traffic classification in IP networks through machine learning techniques in final systems," *IEEE Access*, vol. 11, pp. 44932–44940, 2023.

[24] S. Xie, G. Hu, C. Xing, and Y. Liu, "Online elephant flow prediction for load balancing in programmable switch-based DCN," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 1, pp. 745–758, Feb. 2024.

[25] N. Alkhalidi and F. Yaseen, "FDPHI: Fast deep packet header inspection for data traffic classification and management," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 4, pp. 373–383, Aug. 2021.

[26] F. A. Yaseen, N. A. Alkhalidi, and H. S. Al-Raweshidy, "SHE networks: Security, health, and emergency networks traffic priority management based on ML and SDN," *IEEE Access*, vol. 10, pp. 92249–92258, 2022.

[27] N. Ketkar, J. Moolayil, N. Ketkar, and J. Moolayil, "Introduction to PyTorch," in *Deep Learning With Python: Learn Best Practices of Deep Learning Models With PyTorch*, 2021, pp. 27–91.

[28] P. Jurkiewicz, "Flow-models: A framework for analysis and modeling of IP network flows," *SoftwareX*, vol. 17, Jan. 2022, Art. no. 100929.

[29] P. Jurkiewicz, "Flow-models 2.0: Elephant flows modeling and detection with machine learning," *SoftwareX*, vol. 24, Dec. 2023, Art. no. 101506.

[30] P. Jurkiewicz, "Boundaries of flow table usage reduction algorithms based on elephant flow detection," in *Proc. IFIP Netw. Conf. (IFIP Networking)*, Jun. 2021, pp. 1–9.

[31] S. O. Arík and T. Pfister, "TabNet: Attentive interpretable tabular learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 6679–6687.

BARTOSZ KĄDZIOŁKA received the B.S. and M.S. degrees in electronics and telecommunications engineering from the AGH University of Krakow, Poland, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree with the Institute of Telecommunications. His research interests include software-defined networking, traffic engineering, and network optimization.

PIOTR JURKIEWICZ received the B.S. and M.S. degrees in electronics and telecommunications engineering from the AGH University of Krakow, Poland, in 2012 and 2015, respectively, where he is currently pursuing the Ph.D. degree with the Institute of Telecommunications. His research interests include software-defined networking, flow-based traffic engineering, and multipath and adaptive routing.

ROBERT WÓJCIK received the Ph.D. and D.Sc. degrees (Hons.) in telecommunications from the AGH University of Krakow, Poland, in 2011 and 2019, respectively. He is currently a Professor with the Institute of Telecommunications, AGH University of Krakow. He is the coauthor of more than 70 research publications, including 21 research papers in JCR journals and several patents. He was involved in several international EU-funded scientific projects, such as SmoothIT, NoE BONE, Euro-NF, and smaller national projects. He was a leader of three national science projects. Recently, he has been working on several collaborative research projects involving specialists from industry and academia. His current research interests include multipath routing, flow-aware networking, quality of service, and network neutrality.

JERZY DOMŻAŁ received the M.S., Ph.D., and D.Sc. degrees in telecommunications from the AGH University of Krakow, Poland, in 2003, 2009, and 2016, respectively. He is currently a Professor and the Director of the Institute of Telecommunications, AGH University of Krakow. He is especially interested in optical networks and services for future internet. He is the author or coauthor of many technical papers, two patent applications, and two books. His international training at Universitat Politècnica de Catalunya, Barcelona, Spain, in April 2005; Universidad Autónoma de Madrid, Madrid, Spain, in March 2009; and Stanford University, USA, from May 2012 to June 2012.