Towards Rapid Elephant Flow Detection Using Time Series Prediction for OTT Streaming

1st Anthony Orme
University of Surrey
Surrey, UK
a.orme@surrey.ac.uk

2nd Anthony Adeyemi-Ejeye
University of Surrey
Surrey, UK
femi.ae@surrey.ac.uk

3rd Andrew Gilbert
University of Surrey
Surrey, UK
a.gilbert@surrey.ac.uk

***Abstract*—Broadcast television traditionally employs a unidirectional transmission path to deliver low latency, high-quality media to viewers. To expand their viewing choices, audiences now demand internet OTT (Over The Top) streamed media with the same quality of experience they have become accustomed to with traditional broadcasting. Media streaming over the internet employs elephant flow characteristics and suffers long delays due to the inherent and variable latency of TCP/IP. This paper proposes to perform rapid elephant flow detection on IP networks within 200ms using a data-driven temporal sequence prediction model, reducing the existing detection time by half.**

Early detection of media streams (elephant flows) as they enter the network allows the controller in a software-defined network to re-route the elephant flows so that the probability of congestion is reduced and the latency-sensitive mice flows can be given priority.

We propose a two-stage machine learning method that encodes the inherent and non-linear temporal data and volume characteristics of the sequential network packets using an ensemble of Long Short-Term Memory (LSTM) layers, followed by a Mixture Density Network (MDN) to model uncertainty, thus determining when an elephant flow (media stream) is being sent within 200ms of the flow starting. We demonstrate that on two standard datasets, we can rapidly identify elephant flows and signal them to the controller within 200ms, improving the current count-min-sketch method that requires more than 450ms of data to achieve comparable results.

***Index Terms*—Video streaming, audio streaming, elephant flow, mice flow, latency, TCP, LSTM, MDN, machine learning, SDN, Temporal Sequence Prediction**

I. Introduction

The internet has continued to grow and increase in popularity and has helped drive the wide-scale adoption of handheld devices such as mobile phones, laptop computers, and smart televisions [1]. This increase, in turn, has led to significantly greater consumption of live and prerecorded media over the internet. Furthermore, the bi-directional nature of internet connected devices has encouraged greater communal engagement than traditionally found in the home viewing environment.

As Internet Protocol (IP) packets are distributed asynchronously and randomly, there will be statistical peaks and troughs in the number of packets available in the network at any time. While IP packet loss is inherent within networks, Transmission Control Protocol (TCP) provides reliable IP packet delivery at the expense of increased latency as it relies on resend strategies to account for lost packets [2].

Elephant flows (EFs) describe a particular type of TCP flow that tends to be temporally long and has a high data rate, and mice flows (MFs) are TCP flows that are short-lived and time sensitive [3]. Heavy hitting EFs, and hence media flows, within a network lead to the switch and router buffers filling and possibly overflowing, resulting in congestion, especially for time-sensitive mouse flows. Therefore, there is a need to remove congestion on heavily subscribed network switch egress ports to reduce the risk of holding back mice flows that are short-lived and time-sensitive [3]. Datacentre measurements [4] have shown that 80% of flows within a network are less than a few milliseconds long and less than 10KB in size. While the majority of traffic volume is represented in the top 10% of large flows (EFs), any significant bandwidth traffic (e.g.. greater than 5Mbps) is often considered an EF [5].

Any competition between MFs and EFs for network resources often results in MFs being starved of bandwidth, leading to dropped packets and increased latency [6]. As viewers continue exchanging social media messages through MFs, these can be significantly and negatively influenced, thus degrading the immersive viewing experience. Re-routing the EFs to allow MFs greater bandwidth can potentially improve the network throughput [7].

To resolve this, Liu [8] proposed a load-balancing mechanism based on Software Defined Networks (SDNs) for routing EFs. They then split and send EFs through multiple paths based on the parameters of the states of the links. However, EF detection must be determined early on; this is what our work achieves. Rapid EF detection is essential to reducing network congestion [9] and improving the immersive viewing experience.

Several EF detection methods have been proposed previously [10]. However, they rely on short flow thresholds in the switch, which can lead to high rates of false positives and negatives. While others require periodic extraction of the flow statistics [7] from the network switches to the SDN controller. This may increase network traffic and cause congestion, thus increasing latency. Protocols such as OpenFlow provide basic metrics such as TCP flow rates and IP addresses. However, this method does not work for EF detection using neural network-based machine learning as the data available is too coarse and lacks detail due to the switch measures adopted.

Therefore, we propose a more nuanced data driven approach with the following key contributions:

- A proposed method of combining an ensemble LSTM [11] and MDN [12] with low computational overhead that achieves EF and MF detection within 200ms by capturing long and short term temporal information and modelling uncertainty thus detecting long term media flows to reduce congestion and improve the viewer experience.
- Enabling ML approaches to model continuous flow data by tokenising TCP data streams into 10ms bins.
- A comprehensive evaluation of the proposed ensemble LSTM and MDN method on the open source and repre-

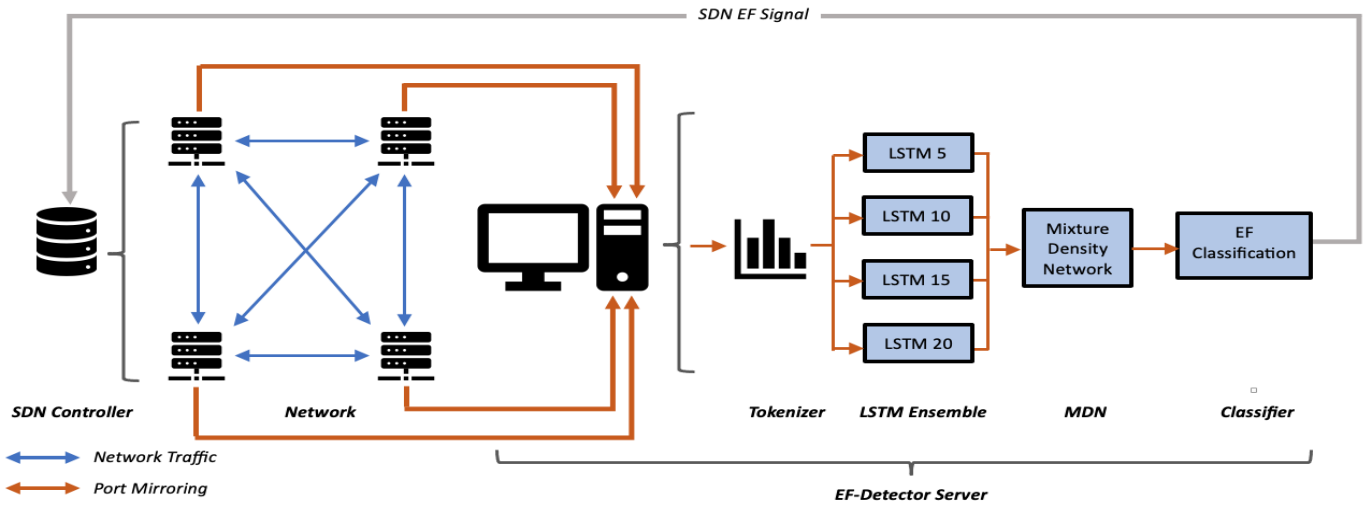

Fig. 1. SDN configuration using switch EF-Detector server and SDN controller.

sentative CAIDA [13] and MAWI [14] datasets.

II. Background and Motivation

A. Elephant Flow related Work

EF definitions can be based on three methods of detection: flow duration, flow data rate, and a combination of flow duration and data rate. Estan [15] uses a method of detection by calculating the flow data rate as a percentage of the overall data rate during a given measure (1 second, 1 minute or 1 hour), and those exceeding a threshold of 0.1% are classified as EFs. Lan [16] uses the datarate method and defines EFs as flows with a data rate larger than xKb/sec. Papagiannaki [17] uses a combined method of deriving a moving average of a data rate during a given measure and determining the length of the flow duration.

B. Detection Methods

Chao [18] uses a machine learning method called Stream Mining based on the Hoeffding Tree [19], operating on a continuous data stream using the labelled CAIDA dataset. Hamdan [20] uses the count-min sketch method [21] to detect mice flow candidates on the switch and then sends the data to a controller to further validate an EF using a Very Fast Decision Tree algorithm (VFDT) [22], the SDN is then signalled and makes the relevant EF routing within the network to avoid congestion. Chao and Hamdan label their EFs as for all TCP flows greater than 5s and an average data rate greater than 5Mb/s.

To predict temporal sequences, RNNs and their variants, including LSTMs [23] and Gated Recurrent Units [11] have shown to learn and generalise the properties of temporal data sequences successfully. Graves [24] was able to predict isolated handwriting sequences, and Alahi [25] was also able to predict human trajectories of crowds by modelling each human with an LSTM and jointly predicting the paths. More recently, researchers turned their attention to the transformer architecture [26] that has proven to excel in sequence prediction in NLP tasks. However, the sparse nature of our CAIDA and MAWI datasets has delivered sub-optimal results for our transformer experiments. In computer networking, LSTMs are used for various network management and optimisation techniques, including network traffic modelling [27]. This enables better load balancing, traffic engineering, and performance diagnostics.

C. Fixed Time Length Detection

Key to our method is that the IP packets are aggregated and tokenized into $10ms$ time bins so that EF detection can achieved within a fixed time period of 200ms, regardless of the length of the overall flows. This differs from other methods such as those provided by [28] [29] and [30] where time series analysis is achieved using individual packets. Methods such as [28] highlight that they can achieve high levels of EF detection using 15 consecutive IP packets, consequently, it could take many seconds to detect and EF as 15 IP packets could occur in a significant length of time. Our method detects EFs within 200ms regardless of the length of the MF and EFs.

D. Modelling Uncertainty

One of the challenges neural networks exhibit is that their practical implementations continue to be black-box designs with little regard for modelling interpretability and output stochasiticity [31]. To overcome this, Swamy [31] proposed a method to improve interpretability and estimation of uncertainty based on a framework using mixture density networks. To enhance network traffic classification and verification due to the number of internet applications, Alizadeh uses a variation of MDNs called Gaussian Mixture Models [32].

III. Design Method

We present our EF detection by comparing our results to the count-min sketch methods provided by Hamdan [20] and determine that the ensemble-LSTM-MDN detects EFs with the same accuracy as Hamdan but in half the time. This is important for SDN networks as the controller's response time is significantly improved, leading to much faster re-routing of long and heavy-hitting EFs, thus reducing MF latency. By signalling EFs only to the SDN controller using a localised

dedicated out-of-band server, we significantly reduce the network traffic to the SDN controller, thus further optimising network latency, as demonstrated in Figure 1, which shows a configuration of the ensemble-LSTM-MDN EF-Detector and how it can be used in an optimised real-life application. The following are required:

- Each switch sends TCP flows to EF-Detector using port mirroring such as Cisco's SPAN
- The EF-Detector receives TCP flows using a dedicated NIC with TCP offload engine, such as the NVidia ConnectX-7 Adapter to reduce network processing.
- The EF-Detector applies the windowing function [33] and tokenises data into twenty bins of 10ms each, discretising the time series data.
- The ensemble-LSTM-MDN EF-Detector sends a message to the SDN controller when an EF has been detected.
- The SDN controller re-routes network traffic when it receives a message from EF-Detector to reduce congestion.

We propose a data-driven architecture: a neural network of LSTM layers and an additional Mixture Density Network (MDN) to identify patterns within processed TCP sequence data to quickly and reliably classify EFs, as shown in Figure 1. We tokenise the TCP data stream into a discrete set of bins; this vector of quantised TCP data is used to train a temporal prediction model via an ensemble of LSTM layers. An LSTM can learn information about the temporal input data within a defined sequence window. Using several LSTMs in parallel allows for multiple window lengths of the tokenised TCP flow data, capturing both short and long-term temporal information about the TCP packets. The outputs of the LSTM ensemble are connected to an MDN layer that adds further detail by modelling uncertainty to the EF prediction by applying Gaussian approximations to enable multimodal modelling of the packet data. A classifier assesses the MDN outputs and provides a classification into a two-hot vector consisting of an EF or MF.

A. Input data Packet Tokenisation

The initial stage is to process the raw data into defined TCP flows. The IP packets are extracted from the raw network data and then decoded into TCP flows into a tuple with the following defined specification - $[ip\_src, ip\_dst, port\_src, port\_dst]$, where ip_src is the IP packet source address, ip_dst is the IP packet destination address, $port\_src$ is the TCP port source address, and $port\_dst$ is the TCP port destination address.

Each flow is unique and has a start and end sequence. To tokenise the data, the number of bytes in the IP packets associated with each flow is aggregated into defined size bins (x), where $x = 10ms$ for this work. Each bin contains the accumulated data for the TCP flow identified in the associated record. Each of these bins provides the average data rate for the flow at $10ms$ intervals. Therefore for a given TCP flow, $\boldsymbol{X}$, with a the total sequence length of n, the TCP flow consists of the n bins, $\boldsymbol{X} = \{x^0, x^1, x^2, ..., x^{n-1}\}$. Then the full extracted set of TCP flows $\mathcal{X}$ can be represented by $\mathcal{X} = \{\boldsymbol{X^{(0)}}, \boldsymbol{X^{(1)}}, ..., \boldsymbol{X^{(t-1)}}\}$. However, applying the windowing technique requires only the first 20 bins, so n is restricted to a value of 20, and t is limited to the number of TCP flows received within the $1s$ window. This tokenisation of the raw data into discrete bins enables temporal sequence prediction via multiple LSTM layers to learn the relationship between the EFs and MFs, and it keeps memory and resource utilisation low in the EF-Detector server.

Tokenisation further improves EF detection as the temporal element of the data packets is implied within each bin, and the robustness of the measurement is maintained. The network switch resources needed to mirror the TCP flows to the SPAN port are small. The EF-Detector server, which is directly connected to the network switch, efficiently collates the TCP streams and tokenises the data for processing by its ensemble-LSTM-MDN inference software; it then communicates an EF detection to the SDN controller with the EFs timestamps, IP addresses, and TCP port numbers. Thus allowing the SDN controller to re-route the EF to reduce latency for the time-sensitive MFs. Furthermore, the network traffic to communicate the aggregated data bins to the SDN controller is negated.

It is possible to treat packets with their timestamps as individual data points and present these to the model; however, we concluded that aggregating the TCP/IP packets into time bins could provide a more generalised solution. Our solution, a data-driven temporal time prediction model with uncertainty estimation, has led to the adoption of time bins. Aggregating the packets into $10ms$ time bins maintains the accuracy of the data packet size information, keeps an element of the timing information, and keeps network traffic low between the EF-Detector server and the SDN controller.

B. LSTM and MDN Layers

Given the temporal nature of TCP packet information represented in the tokenised data, it is desirable to learn and identify the rich temporal patterns between flows to classify EFs and MFs. Long Short-Term Memory (LSTM) layers [11] have provided excellent performance in exploiting longer-term temporal correlations compared to standard recurrent neural networks on many tasks. LSTM layers can store and access information over long periods but mitigate the vanishing and exploding gradient problem common in RNNs through a specialised gating mechanism.

The LSTM is particularly well suited for determining patterns in sequences. However, the size of the window of sequences it can sample is fixed, leading it to potentially learning patterns associated with a specific sequence length. Therefore, to expand and model over a range of window sequence sizes, inspired by [34], we propose to use multiple LSTMs with different sequence sizes and combine their outputs to create both a short and long-term temporal model. The concatenated LSTM layer consisting of 256 nodes connects directly to the MDN input, thus providing it with a rich source of feature embedding in the latent space. While the LSTMs find temporally dependent long and short-term patterns within the network data stream, the MDN models uncertainty to improve the detections provided by the ensemble-LSTM. Neural networks often contain two or more distinct values, i.e., for some x values, $f(x)$ has two or more distinct modes; therefore, $f(x)$ is not a function but is a multi-valued relation. In such a circumstance, the neural network classification will provide a value that is an average conditioned on the target data [12] and consequently lacks accuracy. One method of solving this challenge is to compute the conditional probability density, which is the MDN's approach.

Given the time bin inputs $\{i_0, i_1, ...i_n\}$, where n is the number of time bins per LSTM, each LSTM produces outputs

x_t as follows where h_t is the hidden states of the LSTM units:

$$x_t, h_t = LSTM(i_t, h_{t-1}) \tag{1}$$

The ensemble provides a concatenation of each LSTM such that $\mathbf{X}_t = \{x_t^0, x_t^1, x_t^2, x_t^3\}$ for the $50, 100, 150$, and $200ms$ LSTMs respectively at time t, which forms the input vector to the MDN neural network.$\mathbf{X}$ is fed through the neural network to form the parametised vector $\mathbf{Z}$ which then consists of $\{\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\alpha}\}$, where $\boldsymbol{\mu} = \{\mu_0, \mu_1, ...\mu_M\}$ are the averages, $\boldsymbol{\sigma} = \{\sigma_0, \sigma_1, ...\sigma_M\}$ are the variances, $\boldsymbol{\alpha} = \{\alpha_0, \alpha_1, ...\alpha_M\}$ are the coefficient weights, and M is the number of MDN components. Given the set of modelled features $\mathbf{X}_t$, we model the conditional probability of the EF detection, y_t, at time t:

$$p(y_t|\mathbf{X_t}) = \sum_{i=1}^{M} \alpha_i(\mathbf{X}_t)\phi(y_t|\mathbf{X}_t) \tag{2}$$

Where M is the number of mixture components, $\alpha(\mathbf{X}_t)$ is the mixture coefficient, which represents the probability of the EF at y_t, and $\phi(y_t|\mathbf{X}_t)$ is the conditional density of the EF at y_t. Thus, Equation 2 demonstrates that the probability density of the EF is represented as a linear combination of the mixture components. To achieve classification, then the most likely value for y_t is given by the maximum of the conditional density $p(y_t|\mathbf{X}_t)$ [12]. As the density function is represented as a mixture model, the location of its global maximum is a problem in non-linear optimisation. However, the techniques for this are computationally costly, so an optimised method is provided in Equation 3 where it is assumed that the density functions are not too strongly overlapping [12], and this approximation provides the centre of the highest component for the most likely value of y_t. This corresponds to the centre $\boldsymbol{\mu}_m$, representing the most likely approximation. If this value is greater than 50%, then y_t is classified as an EF; otherwise, y_t is classified as an MF.

$$y_t = \max_m \left\{ \frac{\alpha_m \mathbf{X}_t}{\sigma_m \mathbf{X}_t} \right\} \tag{3}$$

Where $m \in M$ is the number of MDN components.

C. Training and Inference

Training for the CAIDA dataset took approximately half an hour, and for the MAWI dataset, it took five hours. Once the training was complete, inference for the complete ensemble-LSTM-MDN model occurred within $1ms$ to provide an EF or MF classification using a single NVidia RTX5000 GPU operating in a 2x Intel Xeon Silver 4210R 2.4G server.

IV. Experiments

We evaluate our approach using the standard 6hr CAIDA [13] and 24hr MAWI dataset [14]. These were chosen as they contain EFs representative of heavy hitting media flows found in OTT streaming. To label the datasets, an EF is defined following the method of Hamdan [20]; EFs have a data rate greater than 5Mbps and are longer than $5s$. Each dataset consists of more MFs than EFs, with the MFs, on average, being significantly lower in data rate and time than the EFs. Using the data with these proportions would cause a significant skew and bias in learning and overfit. Therefore, each dataset was randomly subsampled so that the ratio of MFs to EFs is 50:50. Each dataset was further partitioned to provide a train: test ratio of 90:10. CAIDA consisted of 6,343 training records and 704 test records, and MAWI consisted of 42,804 training records and 4,756 test records.

A. Baseline Methods

Four methods were chosen to baseline our proposed EF detection model against: Gudibanda [35], Mohammed [36], Chao [18] and Hamdan [20]. Gudibanda [35] uses the Gaussian **Naive Bayes** method to assume conditional independence between every pair of features for the class. This method assumes all the features are independent, which may not be the case due to the cyclical nature of TCP flows. Mohammed [36] reviews and proposes **Linear Regression**, which assumes the detection fits a constant slope. Although this model is predominantly used to predict continuous variables, it was used in these tests to determine if the relationship between the features and the detection was easily predictable. Chao [18] uses a **Hoeffding Tree** [19], which assumes that the distribution does not change significantly over time. And Hamdan [20] uses the **Count-Min-Sketch** to determine the frequency of the table of events. We use several standard metrics to evaluate the performance of our proposed approach, **precision**, **recall**, **f-measure** and **MCC** [37].

B. Results

Table I and Table II provide a summary of the key measures for each method and a comparison to our ensemble-LSTM-MDN proposal. Our method continually demonstrates higher metrics and lower errors than the other methods, especially considering the time it takes to detect the EF. For the CAIDA dataset in Table II, Gudibanda [35] and Mohammed [36] achieve an f-measure of 0.89 after $350ms$, and Hamdan [20] achieves an f-measure of 0.80 after $500ms$. However, for the MAWI dataset in Table I, Gudibanda [35] only achieves an f-measure of 0.72, Mohammed [36] achieves an f-measure of 0.83 and Hamdan [20] achieves an f-measure of 0.91 after $400ms$, compared to our results using the ensemble-LSTM-MDN of 0.91 after $200ms$. The MAWI dataset in Table I achieves better results due to the significant increase in number of training records available compared to the CAIDA dataset. Due to the limited number of training records available in the CAIDA dataset, the ensemble-LSTM-MDN will be overfitting, which implies bias in the dataset. Hence, the performance is not as good as the MAWI dataset. Using larger datasets allows the model to be more general and have less bias without overfitting the data (i.e. without leading to increased variance) [12]. Both Chao [18] and Hamdan [20] perform better as the number of $10ms$ bins that are presented increases to $500ms$ for the decision tree and count-min-sketch models. However, this is to be expected as their accuracy increases with the amount of data presented to them.

To motivate the use of an ensemble of LSTMs, Table III shows the performance of the metrics for a single LSTM layer for the four sequence window sizes used by the ensemble architecture, and then for increasing numbers of parallel LSTMs. The comparison shows how the metrics steadily improve as more LSTMs are added to the ensemble. A single LSTM cannot capture enough information to compete with the multiple LSTM models, with even the dual LSTM improving EF detection metrics. Tests with five LSTMs were conducted but did not improve on the results in Table III due to overfitting. To motivate the inclusion of the MDN network, Table IV

Time Bin Length	Naive Bayes [35]		Linear Regression [36]		Hoeffding Tree [18]		Count-Min Sketch [20]		ensemble-4 LSTM-MDN	
	MCC	FMeas	MCC	FMeas	MCC	FMeas	MCC	FMeas	MCC	FMeas
100ms	0.38	0.72	0.52	0.79	0.47	0.76	0.69	0.78	**0.74**	**0.82**
200ms	0.38	0.72	0.54	0.82	0.53	0.75	0.76	0.84	**0.84**	**0.91**
300ms	0.38	0.72	0.61	0.82	0.67	0.80	0.80	0.88	**0.87**	**0.92**
400ms	0.34	0.71	0.58	0.81	0.85	0.85	**0.84**	**0.91**	0.88	0.92
500ms	0.34	0.71	0.34	0.71	**0.99**	**0.99**	0.85	0.91	0.87	0.92

TABLE I
MAWI 2022 DATASET COMPARISON TO OTHER METHODS. THE HIGHLIGHT SHOWS THE 200MS DETECTION TARGET TIME.

Time Bin Length	Naive Bayes [35]		Linear Regression [36]		Hoeffding Tree [18]		Count-Min Sketch [20]		ensemble-4 LSTM-MDN	
	MCC	FMeas	MCC	FMeas	MCC	FMeas	MCC	FMeas	MCC	FMeas
100ms	0.33	0.75	0.33	0.75	0.27	0.64	0.00	0.00	**0.70**	**0.72**
200ms	0.66	0.86	0.67	0.86	0.25	0.23	0.18	0.11	**0.78**	**0.86**
300ms	0.70	0.87	0.72	0.89	0.66	0.86	0.44	0.50	**0.79**	**0.87**
400ms	0.74	0.89	0.74	0.89	**0.80**	**0.90**	0.61	0.70	0.79	0.87
500ms	0.74	0.89	0.74	0.89	0.63	0.75	**0.70**	**0.80**	0.88	0.81

TABLE II
CAIDA 2019 DATASET COMPARISON TO OTHER METHODS. THE HIGHLIGHT SHOWS THE 200MS DETECTION TARGET TIME.

Architecture	Seq Length	MCC↑	f-Measure↑
Single LSTM	10	0.51	0.59
Single LSTM	20	0.59	0.65
Single LSTM	30	0.63	0.69
Dual LSTM	5, 10	0.51	0.59
Dual LSTM	25, 30	0.66	0.70
Triple LSTM	5, 10, 15	0.55	0.61
Triple LSTM	15, 20, 30	0.68	0.72
Proposed LSTM-4	5, 10, 15, 20	**0.68**	**0.72**

TABLE III
PERFORMANCE OF PROPOSED METHOD FOR INCREASING NUMBER OF LSTMS (WITHOUT MDN) FOR CAIDA DATASET

Time Bin Length	ensemble-4 LSTM		ensemble-4 LSTM-MDN	
	MCC	FMeas	MCC	FMeas
100ms	0.72	0.76	0.74	0.82
200ms	0.79	0.85	0.84	0.91
300ms	0.80	0.86	0.87	0.92
400ms	0.81	0.87	0.88	0.92
500ms	0.79	0.86	0.87	0.91

TABLE IV
MAWI DATASET COMPARING ENSEMBLE-LSTM TO ENSEMBLE-LSTM-MAWI. THE HIGHLIGHT SHOWS THE 200MS DETECTION TARGET TIME.

compares EF detection by comparing the ensemble-LSTM with the ensemble-LSTM-MDN. The results show a consistent improvement when using the MDN layer.

V. CONCLUSION AND FUTURE WORK

The results have demonstrated that the proposed data-driven ensemble-LSTM-MDN model provides consistently better short-term EF detection results than the other methods considered. The speed with which detection can be achieved will reduce the possibility of latency, allowing network operators to improve load balancing and lower latency, especially for time-sensitive events.

Future work will explore reducing the token time length to lower than $10ms$ to improve the metrics and, hence, EF detection speed. This should increase the number of tokens available to the ensemble-LSTM-MDN model, enhancing the detection accuracy and, thus, improving the metrics. However, there is the potential for over-fitting. The four ensemble-LSTM-MDN model achieved the required result by predicting EFs with an f-measure metric of 0.91 within $200ms$. The MDN layer is of particular interest and requires further investigation as it models uncertainty in the dataset. This will improve detection as the input IP packets continuously vary in the time domain.

REFERENCES

[1] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching television over an IP network," *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, 2008.

[2] W. Eddy, *Transmission Control Protocol (TCP)*, RFC 9293, 2022.

[3] T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto, "Identifying elephant flows through periodically sampled

packets," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004.
[4] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010.
[5] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, *et al.*, "Hedera: Dynamic flow scheduling for data center networks.," in *Nsdi*, San Jose, USA, 2010.
[6] W. Wang, Y. Sun, K. Salamatian, and Z. Li, "Adaptive path isolation for elephant and mice flows by exploiting path diversity in datacenters," *IEEE Transactions on Network and Service Management*, 2016.
[7] C. K.Lou Y.Yang, "An elephant flow detection method based on machine learning," in *Smart Computing and Communication*, 7th International Conference Smart Computing and Communication, 2019.
[8] J. Liu, J. Li, G. Shou, Y. Hu, Z. Guo, and W. Dai, "Sdn based load balancing mechanism for elephant flow in data center networks," in *2014 International Symposium on Wireless Personal Multimedia Communications (WPMC)*, IEEE, 2014.
[9] L. TIANYu, B.-s. LAIYing-xu, *et al.*, "Tpefd: An sdn-based efficient elephant flow detection method," *Chinse Journal of Network Information Security*, 2017.
[10] F. Tang, H. Zhang, L. T. Yang, and L. Chen, "Elephant flow detection and load-balanced routing with efficient sampling and classification," *IEEE Transactions on Cloud Computing*, 2019.
[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.
[12] C. M. Bishop, *Mixture density networks*, 1994.
[13] *Caida anonymized internet traces 2019*, https://catalog.caida.org/dataset/passive_2019_pcap.
[14] *Mawi working group traffic archive*, 2022.
[15] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, 2002.
[16] K. Lan and J. Heidemann, "On the correlation of internet flow characteristics," Technical Report ISI-TR-574, USC/ISI, Tech. Rep., 2003.
[17] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot, "A pragmatic definition of elephants in internet backbone traffic," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002.
[18] S.-C. Chao, K. C.-J. Lin, and M.-S. Chen, "Flow classification for software-defined data centers using stream mining," *IEEE Transactions on Services Computing*, 2016.
[19] A. Kumar, P. Kaur, and P. Sharma, "A survey on hoeffding tree stream data classification algorithms," *CPUH-Res. J*, 2015.
[20] M. Hamdan, B. Mohammed, U. Humayun, *et al.*, "Flow-aware elephant flow detection for software-defined networks," *IEEE Access*, 2020.
[21] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *Journal of Algorithms*, 2005.
[22] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000.
[23] Z. Ferdoush, B. N. Mahmud, A. Chakrabarty, and J. Uddin, "A short-term hybrid forecasting model for time series electrical-load data," *International Journal of Electrical and Computer Engineering*, 2021.
[24] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.
[25] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
[26] A. M. Braşoveanu and R. Andonie, "Visualizing transformers for nlp: A brief survey," in *2020 24th International Conference Information Visualisation (IV)*, IEEE, 2020.
[27] A. Lazaris and V. K. Prasanna, "An lstm framework for modeling network traffic," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019.
[28] G. Wassie Geremew, J. Ding, *et al.*, "Elephant flows detection using deep neural network, convolutional neural network, long short-term memory, and autoencoder," *Journal of Computer Networks and Communications*, 2023.
[29] P. Jurkiewicz, "Flow-models 2.0: Elephant flows modeling and detection with machine learning," *SoftwareX*, 2023.
[30] J. d. M. Bezerra, A. J. Pinheiro, C. P. de Souza, and D. R. Campelo, "Performance evaluation of elephant flow predictors in data center networking," *Future Generation Computer Systems*, 2020.
[31] G. Swamy, A. Das, and S. Niranjan, "Causal interpretability and uncertainty estimation in mixture density networks*," in *ICANN 2023*, 2023.
[32] H. Alizadeh, H. Vranken, A. Zúquete, and A. Miri, "Timely classification and verification of network traffic using gaussian mixture models," *IEEE Access*, 2020.
[33] R. E. Sibai, Y. Chabchoub, J. Demerjian, Z. Kazi-Aoul, and K. Barbar, "Sampling algorithms in data stream environments," in *2016 International Conference on Digital Economy (ICDEc)*, 2016.
[34] J. Y. Choi and B. Lee, "Combining lstm network ensemble via adaptive weighting for improved time series forecasting," *Mathematical problems in engineering*, 2018.
[35] A. Gudibanda, J. Ros-Giralt, A. Commike, and R. Lethin, "Fast detection of elephant flows with dirichlet-categorical inference," in *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*, IEEE, 2018.
[36] A. R. Mohammed, S. A. Mohammed, and S. Shirmohammadi, "Machine learning and deep learning based traffic classification and prediction in software defined networking," in *2019 IEEE International Symposium on Measurements & Networking (M&N)*, IEEE, 2019.
[37] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score

and accuracy in binary classification evaluation," *BMC genomics*, 2020.