IBC2023

IMPROVING THE BROADCAST VIEWER EXPERIENCE THROUGH RAPID AND EARLY ELEPHANT FLOW DETECTION BY MACHINE LEARNING

Anthony Orme, Anthony Adeyemi-Ejeye, Andrew Gilbert

University of Surrey, UK

ABSTRACT

Keeping broadcast IP network latency low is critical in maintaining the immersive viewing experience, especially when delivering high quality broadcast media over the Internet or broadcast IP datacentres. The network and resource requirements of heavy-hitting broadcast media flows with high datarates and temporal longevity clash with the needs of latency sensitive short data flows, leading to switch buffer overload and network congestion resulting in dropped packets and increased latency due to TCP-RTOs (Transmission Control Protocol Retransmission Time-Out). Within broadcast datacentres the media flows often fall under elephant flow (EF) classification, with the short flows being classified as mice flows (MF). Rapid and early detection of EFs will allow the SDN controller to re-route them and reduce their impact on the MFs within the broadcast IP network. This reduces packet dropout so that the TCP-RTOs are not triggered resulting in latency being kept low and the immersive viewing experience being improved. Although EF detection has been researched extensively, this paper proposes a new approach to detect EFs for the broadcast network SDN controller within 500ms, thus allowing the SDN controller to re-route the EFs and reduce packet loss. This method uses machine learning with ensemble LSTM (Long Short-Term Memory) neural networks, with each LSTM being a different length so the ensemble can capture the non-linear characteristics of the varying flow sizes. The ensemble LSTM outputs are then concatenated and further processed by a neural network. Training is achieved by back propagating through the neural network and then each LSTM resulting in a greater inference EF detection accuracy for the broadcast IP network. Our approach was tested on industry standard datasets and achieved EF detection in under 500ms without needing to be reliant on statistical information provided by network switches thus further reducing latency and improving the immersive viewing experience, unlike other approaches.

INTRODUCTION

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. Still, it does this at the expense of latency as it is

connection-oriented between the client and server and relies on resend strategies to account for lost packets [2]. A client initiating a connection will wait for the server to acknowledge the request, after which the client sends the IP packets associated with the media being delivered. When all the data is sent, the client will close the connection.

TCP is used extensively for streaming video, audio, and metadata to viewers on their smart TVs and mobile devices. Consequently, the prevalence of TCP flows continues to grow significantly resulting in a greater number of EFs that need dynamic re-routing to reduce the risk of latency for viewers, especially when viewers exchange social media messages. Furthermore, as broadcast IP network infrastructures increase in complexity, which is inevitable if IP is going to deliver the flexibility it promises, then the prevalence of TCP cannot be ignored especially when broadcasters integrate uncompressed UDP streams, such as ST2110, with compressed TCP video and audio streams.

Elephant TCP flows tend to be temporarily long and fill up buffers within switches due to their high and steady data rates, especially when egress ports are heavily subscribed, and if most of the buffers are dedicated to these 'high steady states' then this can lead to packet loss for short packet bursts, resulting in increased latency and a poor user experience. Hash based ECMP (Equal Cost Multipathing) is often employed in networks to choose the shortest path for routing as it is simple to implement and doesn't require per-flow information from the switches. ECMP is unable to differentiate between MFs and EFs and suffers from hash collisions sometimes resulting in multiple EFs being mistakenly sent across the same link, thus further exasperating buffer overflow and packet loss [37]. Therefore, there is a need to remove congestion on heavily subscribed egress ports and reduce the risk of holding back MFs that are short-lived, and time-sensitive [1]. To resolve this, Liu [6] proposed a load balancing mechanism based on SDNs for routing EFs by gaining the topology and status of the entire network. They then split and send EFs through multiple paths based on the parameters of the states of the links. However, for SDN re-routing to be effective, EF detection must be as fast as possible; this is what our work proposes. Unlike other methods that require many seconds of TCP flow data to establish an EF, our method achieves EF detection in less than 500ms. This reduces the risk of buffer overflow and hence packet drop, resulting in an improved immersive viewing experience.

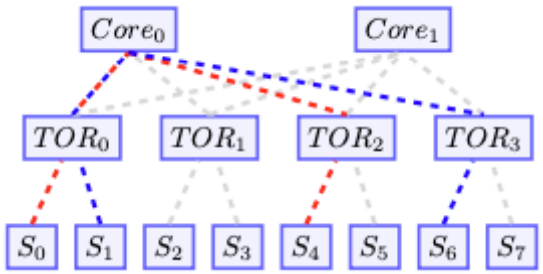

Fig. 1. Congestion and packet loss can occur between TOR_0 (Top of Rack) and the core switch ($Core_0$) when device S_0 is sending short bursts of MFs and S_1 is sending EFs

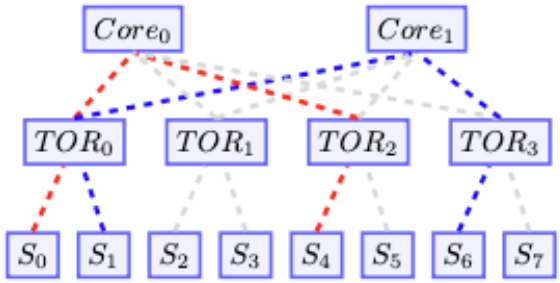

Fig. 2. A solution where TOR_0 routes the traffic from device S_1 via core switch $Core_1$. However, TOR_0 must detect the EF early.

Data centre measurements [22] [23] have shown that 80% of the total flows within the network are less than a few milliseconds long and less than 10KB in size. The majority of

traffic volume is represented in the top 10% of large flows (EFs), and any significant bandwidth traffic (e.g greater than 1MBps) is often considered an EF [26]. Any competition between MFs and EFs for network resources often results in MFs being starved of bandwidth which often leads to dropped packets and increased latency [25]. Furthermore, re-routing the EFs to allow MFs greater bandwidth can potentially improve the network throughput [24]. The SDN controller does not need to process all EFs, only those that significantly impact the network performance. Inefficient management will fill network buffers with EFs, thus leading to queuing delays and dropped packets. Consequently, rapid EF detection is essential to reducing network congestion [27].

Figure 1 demonstrates how EF detection can be used to stop EF and MF conflicts by dynamically re-routing EFs. However, early EF detection is essential to reduce the risk of the switch buffers overflowing. Buffers that overflow will drop packets which in turn will lead to massively increased latency in TCP flows.

EF detection might appear relatively trivial as a network operator could argue that any TCP flow below a threshold of 250ms (for example) is a MF, and anything greater is an EF. However, only the EFs that significantly impact the network performance need to be re-routed, that is, EFs that are greater than 10s [28], and waiting for 10s to detect an EF is not viable in real world networks. Hence, our proposal can detect an EF in less than 500ms and therefore classify a TCP flow as either a MF or EF in under 500ms.

Several methods of EF detection techniques have been previously proposed [24], [25], [29]–[35]. However, they rely on short flow thresholds in the switch, which can lead to high rates of false positives and negatives. Some methods require periodic extraction of the flow statistics [24], [25], [33], [34] from the network switches to the SDN controller, which in itself may increase network traffic, thus contributing to congestion. This further leads to a significant increase in flow detection and re-routing latency.

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

- The tokenisation of the TCP data streams into 10ms bins enables machine learning approaches to model the continuous flow data.
- The introduction of a data driven temporal time prediction model, using an ensemble model of LSTMs (Long Short- Term Memory) layers to capture both short- and long-term temporal information about the data flow and classify a TCP stream as either a MF or EF, with low computational overhead.
- Extensive testing of the proposed method on the industry standard CAIDA [8].

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 [3] uses a method of detection by calculating the flow datarate 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 [4] uses the datarate method and defines EFs as flows with a datarate larger than xKB/sec. Papagiannaki [5] uses a combined method of deriving a moving average of a datarate during a given measure and determining the length of the flow duration. The third standard deviation of the average flow duration forms the threshold along with the data rate.

Chao [18] uses a machine learning method called Stream Mining based on the Hoeffding Tree [19]. This operates on a continuous data stream using the labelled CAIDA dataset. Their labelled set classifies an elephant for all TCP flows greater than 5000ms and average data rate greater than 50MB/s. We present our EF detection by comparing our results to Chao [18] and determine that the ensemble LSTM detects EFs with the same accuracy as Chao 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 TCP EFs.

To predict temporal sequences, RNNs and their variants, including LSTMs [9] and Gated Recurrent Units [10] have shown to learn and generalise the properties of temporal data sequences successfully. Graves [11] was able to predict isolated handwriting sequences, Alahi [12] 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 [14] that has proven to excel in sequence prediction in NLP tasks. In computer networking, LSTMs are used for various network management and optimisation techniques, including network traffic modelling [15]. This enables better load balancing, traffic engineering, and performance diagnostics. Li [16] provides a method of detecting Distributed Denial of Service (DDoS) attacks by combining LSTMs and Bayes methods. Furthermore, Dey [17] provides an anomaly detection solution that detects any breaches of the control plane within software defined networks.

METHOD

We propose a data driven architecture; a neural network formed of LSTM layers that can identify patterns within processed TCP sequence data to classify EFs as shown in Figure 3. We tokenise the TCP data stream into a discrete set of bins; this vector of quantise TCP data is used to train a temporal prediction model via an ensemble of LSTM layers. An LSTM can learn information about temporal input data within a defined temporal window, and using several LSTMs in parallel allows for multiple window lengths of the tokenised TCP flow data to be analysed. This captures both short- and long-term temporal information about the TCP packets. The results of the ensemble streams are then concatenated via a further multi-Layer MLP and a softmax to convert the vector into a two hot vector, which is then classified into a MF or EF.

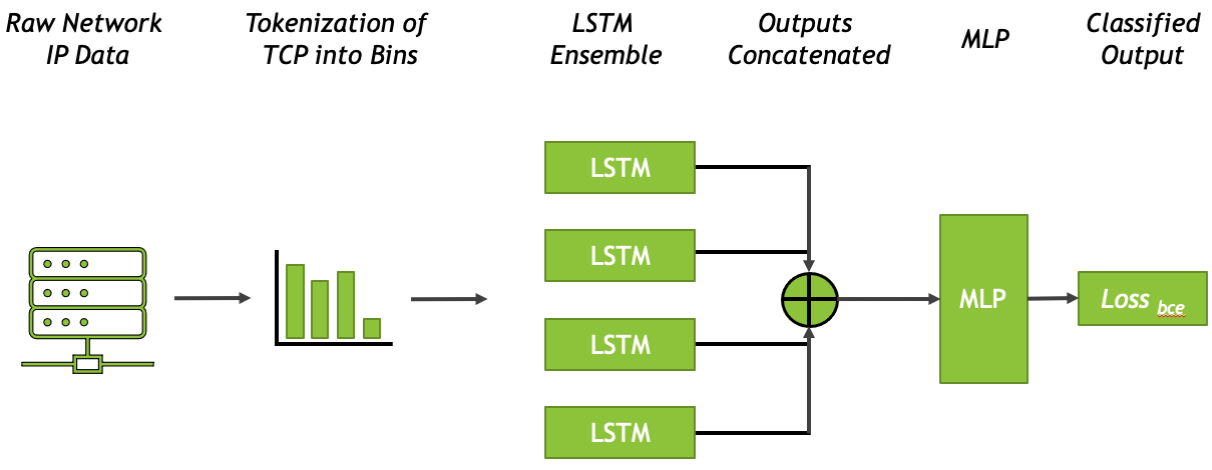

Fig. 3. An overview of the approach.

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. The packets are 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, 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 are aggregated into defined size bins (x), each of 10ms in size 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, X, with the total sequence length of n, the TCP flow consists of the n bins, $X = \{x^0, x^1, x^2, ..., x^{n-1}\}$. Then the full extracted set of TCP flows X can be represented by $X = \{X^{(0)}, X^{(1)}, ..., X^{(n-1)}\}$. This tokenisation of the raw data into discrete bins enables temporal sequence prediction via multiple LSTM layers to learn the relationship between EF and MF TCP flows.

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 amount of router resources needed to collate the data is small compared to collating and communicating detailed information, such as timestamps, to the SDN controller. Furthermore, the network traffic to communicate the aggregated data bins to the SDN controller is also reduced.

LSTM Machine Learning Layer

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 [9] 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 gradient problem common in RNNs through a specialised gating mechanism.

Given an input vector $J^i(t)$ at time *t* consisting of binned packet data and resulting output joint vector $J^o(t)$. The aim is to learn the function that minimises the loss between the input vector and the output vector $J^o = o_t \circ \tanh(c_t)$ (where $\circ$ denotes the Hadamard product), o_t is the output gate, and c_t is the memory cell. A combination of the previous memory c_{t-1} multiplied by a *forget-gate*. Thus, intuitively, it combines the previous memory and the new input. For example, the old memory could be completely ignored (forget gate all 0's) or completely ignore the newly computed state (input gate all 0's), but in practice, it is between those two extremes.

LSTM Sequencing

It is possible to treat packets with their timestamps as individual data points and present these to the model; however, we concluded that the data rate of the reporting traffic from the

router to the SDN controller would be excessive. Our solution, a data driven temporal time prediction model, 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 router and SDN controller.

To establish the relationships between packets and hence detect EFs in the least possible time, each time bin forms part of a unique temporal sequence presented to each layer of the LSTM. These sequence lengths relate directly to the time taken to detect an EF and vary in length to enable a greater detection granularity. A short sequence length consisting of five 10ms time bins will detect short-term TCP flow activity, which could well be MFs, and hence would discount them from the final classification. Nevertheless, it is equally possible that specific EFs could burst quickly in the first 50ms, and the five-bin sequence layer of the model would train and detect these. The maximum sequence length is 50, representing 500ms (10ms · 50). By choosing different sequence lengths, data patterns representing the characteristics of EF data rates and time can be better detected.

Ensemble LSTM Method

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 [7], we propose to use multiple LSTMs with different sequence sizes and combine their outputs to enable both a short- and long-term temporal model to be created.

Figure 4 illustrates the proposed architecture, where there is a set of parallel LSTM layers, where each LSTM has its unique sequence or window length, the outputs are a fixed size and are concatenated via an MLP layer; this, in turn, provides the detection output.

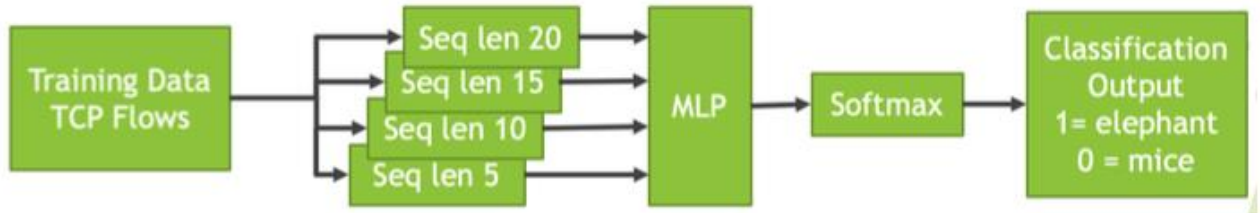

Fig. 4. Ensemble LSTM block diagram showing the configuration of the individual LSTM modules concatenated via an MLP

By varying the sequence length for each LSTM and the hyperparameters, such as learning rate and hidden layer size, the model can learn the hyperparameters to model relationships over a range of window sizes to improve the detection accuracy of EFs. The sequence length aims to be as low as possible because this research aims to provide a method of detecting EFs in the shortest time possible. Given k possible sequence windows (for this work, the windows are 5,20,35,50 tokens), it is possible to create an ensemble of parallel LSTM layers concatenated into a single vector R which can then be classified using the softmax layer into EFs and MFs.

Implementation Details

The model was written in python using the PyTorch library, uses the adam optimiser, and a learning rate of 0.001. It is trained on a single RTX5000 GPU and takes around 6 hours to train. Inference took less than 1ms, which could easily be facilitated on a low power GPU such as the Nvidia TX2 module edge device. Consequently, an EF detection is processed and reported to the SDN controller in just over 500ms from the start of the TCP flow. We propose that the EF detection modules be placed at the input to the edge switches within the network and report any EFs to the SDN controller directly. This reduces the switches and SDN controller's processing overhead and provides accurate and rapid EF detection. There is no processing overhead in the switch or SDN controller. The proposed LSTM model does not store any data points as it processes each flow as a sequence of time bins; consequently, the inference, and hence detection time is fixed and doesn't vary as a function of the number of flows within the stream as in Hamdan [36].

Practical Example

A broadcast use-case solution is demonstrated in Figure 5 showing a REMI type configuration. The vision engineer resides in the studio and controls the shading and iris functions of the remote camera located at the OB, and a camera operator resides at the OB to pan, tilt, focus and zoom the camera. If the video and control data are routed over the same network, even if a CDN is engaged, there is a high probability that the time sensitive OCP control data will be delayed and subject to variable and undesirable latency due to the dominance of the video and audio streams (EFs) in the CDN, especially when multiple cameras are employed. This will result in the shading and iris control functions being compromised as the vision engineer will experience poor adjustment over the iris and colour balance; the camera video will be intermittently over- and under-exposed, and over- and under-colour corrected resulting in a poor viewing experience. By automatically detecting the long EFs (video and audio) and routing them over a CDN, the OCP data and other time sensitive data can then be routed over a separate CDN, thus improving the accuracy of the OCP control for the vision engineer resulting in consistently accurately exposed and colour corrected high quality images which in turn will deliver an improved viewing experience.

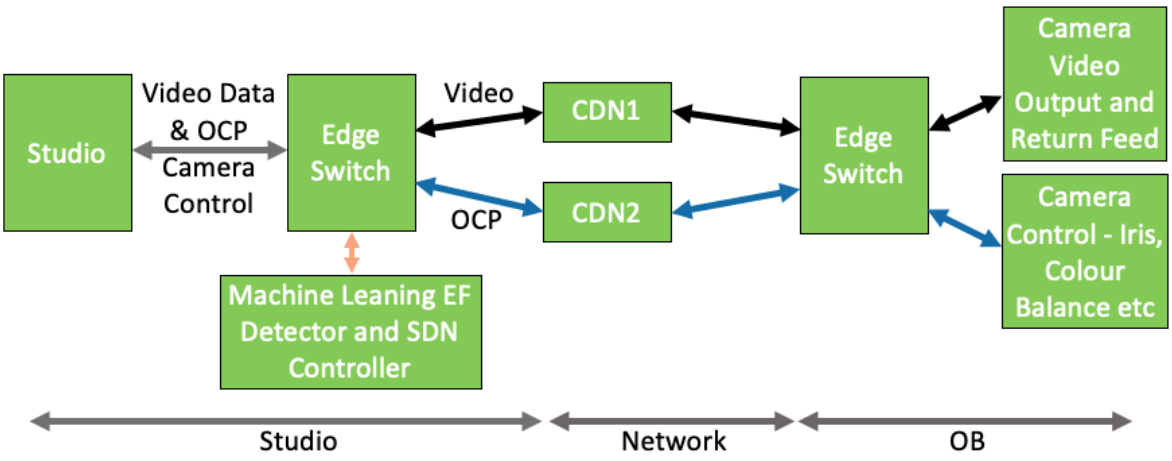

Fig 5. A REMI type studio and OB configuration with the vision engineer shading the cameras from the studio, and the camera operator positioning and framing the camera at the OB.

EXPERIMENTS

To evaluate our approach, we use the standard CAIDA dataset [8] and provide several baseline comparison methods verified via defined metrics.

Dataset Pre-processing

The CAIDA dataset [8] was chosen as it provides a continuous data centre layer-2 stream recording of six hours of data, thus providing many opportunities for long EFs to establish themselves. The accurate detection was possible for EFs and MFs during pre-processing to provide detailed training and test datasets, where an EF is defined following the method of Chao [18]. The dataset consists of 300,908 MFs, and 33,434 EFs, giving the ratio of the number of mice to EFs as 90:10. However, the data rate of MFs to EFs is 31:69, showing that 69% of the data are from EFs are responsible for 10% of the number of TCP flows.

To reduce the risk of overfitting during training, the dataset was balanced with a MF to EF ratio of 50:50. A random subsample of MFs was used to match the number of EFs and provide 33,434 MFs, with a total of 66,868 flows in the dataset.

Test Metrics

We use several standard metrics to evaluate the performance of our proposed approach, precision, recall, f-measure, MCC, and Mean Absolute Percentage Error. F-measure is a measure of classification accuracy. Mean absolute percentage error (MAPE) is a measure of how accurate a prediction system is and measures the accuracy as a percentage. Mathews Correlation Coefficient (MCC) takes into consideration all four measures (true positive, true negative, false positive and false negative) within the binary classification of the confusion matrix to provide a value within the range +1 and −1. Value +1 is 100% correlated and −1 is 100% uncorrelated.

Baseline Methods

Three methods were chosen to baseline our proposed EF detection model against: Gudibanda [20], Mohammed [21] and Chao [18]. Gudibanda [20] uses the Gaussian Naive Bayes method to make the assumption of conditional independence between every pair of features for the class. This method does assume all the features are independent, and this may well not be the case due to the cyclical nature of TCP flows. Mohammed [21] 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 is a data driven method that uses an incremental decision tree.

Results

Table 1 shows the proposed ensemble method LSTM-4 compared to the baselines predicting EFs and MFs using the first 500ms of the TCP flows from the CAIDA datasets.

Table 1 shows that Chao's method provides the closest performance to our proposed approach. In contrast, the others perform poorly with the limited amount of data provided

with the 500ms across two standard metrics. The Bayesian method with a MAPE of 0.11 and f-measure of only 0.56 is demonstrating a near random outcome. And the Linear Regression is similar, this further indicates that the solution is non-linear and too complex for the Linear Regression method to work effectively.

Method	MCC↑	f-measure ↑	MAPE ↓
Naive Bayes [20]	0.12	0.56	0.11
Linear Regression [21]	0.16	0.66	0.14
Chao [18]	0.72	0.80	0.08
Proposed LSTM-4	**0.77**	**0.89**	**0.09**

Table 1 – Performance of the proposed method against baselines for EF detection using the first 500ms of TCP flows

CONCLUSION

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

For broadcasters this will further improve the immersive viewing experience as there will be fewer dropped packets as heavy hitting media TCP EFs will be rapidly detected and then re-routed using SDNs to improve network optimisation and data throughput. Furthermore, the computational overhead on the switches and SDN is very low.

To improve the metrics, and hence EF detection speed, future work will explore reducing the token time length to be lower than 10ms. This should increase the number of tokens available to the LSTMs, which will improve the detection accuracy and hence improve the metrics, although there is the potential for over-fitting.

The four LSTM ensemble has achieved the required result by predicting EFs with an f-measure metric of 0.89 within 500ms.

As broadcasters continue to optimise IP systems to provide the flexibility and scalability that IP promises, then network engineers are going to need to consider more efficient and dynamic methods of traffic management, especially when considering media flows. Our method improves EF detection times so that SDNs can efficiently route heavy hitting media flows to reduce buffer overflow and packet loss, hence further improving the immersive viewing experience.

References

[1] Mori, Tatsuya and Uchida, Masato and Kawahara, Ryoichi and Pan, Jianping and Goto, Shigeki, “Identifying EFs through period- ically sampled packets”, 2004 ACM SIGCOMM Internet Measurement Conference

[2] Postel, Jon, “Transmission Control Protocol - RFC793”, 1981 Internet Requests for Comments

[3] Estan, Cristian and Varghese, George “New directions in traffic measurement and accounting”, Proceedings of the 2002 conference on Applications, technologies

[4] Lan, Kun-Chan and Heidemann, John, “On the correlation of Internet flow characteristics”, 2003, Citeseer

[5] Papagiannaki, Konstantina and Taft, Nina and Bhattacharyya, Supratik and Thiran, Patrick and Salamatian, Kave´ and Diot, Christophe, “A pragmatic definition of elephants in Internet backbone traffic”, 2002, Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement

[6] Liu, Jing and Li, Jie and Shou, Guochu and Hu, Yihong and Guo, Zhigang and Dai, Wei, “SDN based load balancing mechanism for elephant flow in data center networks”, 2014 International Symposium on Wireless Personal Multimedia Communications

[7] Choi, Jae Young and Lee, Bumshi, “Combining LSTM network Ensemble via adaptive weighting for improved time series forecasting”, 2018, Mathematical problems in engineering

[8] CAIDA-Dataset, “Center for Applied Internet Data Analysis”, caida.org, 2021

[9] Hochreiter, Sepp and Schmidhuber, Ju ̈rgen, “Long short-term memory”, 1997, Neural computation

[10] Chung, Junyoung and Gulcehre, Caglar and Cho, KyungHyun and Ben- gio, Yoshua, “Empirical evaluation of gated recurrent neural networks on sequence modelling”, 2014, arXiv preprint arXiv:1412.3555

[11] Graves, Alex, “Generating sequences with recurrent neural net- works”,2013,arXiv preprint arXiv:1308.0850

[12] Alahi, Alexandre and Goel, Kratarth and Ramanathan, Vignesh and Robicquet, Alexandre and Fei-Fei, Li and Savarese, Silvio, “Social lstm: Human trajectory prediction in crowded spaces”, 2016 Conference on Computer Vision and Pattern Recognition

[13] Ferdoush, Zannatul and Mahmud, Booshra Nazifa and Chakrabarty, Amitabha and Uddin, Jia, “A short-term hybrid forecasting model for time series electrical-load data using random forest and bidirectional long short-term memory.”, 2021, Journal of Electrical & Computer Engineering

[14] Bras ̧oveanu,AdrianMPandAndonie,Ra ̆zvan,“Visualizingtransformers for nlp: a brief survey”,2020 Information Visualisation (IV)

[15] Lazaris, Aggelos and Prasanna, Viktor K, “An LSTM framework for modelling network traffic”, 2019 Symposium on Integrated Network and Service Management

[16] Li,YanandLu,Yifei,“LSTM-BA:DDoSdetectionapproachcombining LSTM and Bayes”, 2019 Conference on Advanced Cloud and Big Data

[17] Dey, Samrat Kumar and Rahman, Md Mahbubur, “Effects of machine learning approach in flow-based anomaly detection on software-defined networking”,2019 Symmetry Journal

[18] Dhao and Lin, “Flow Classification for Software-Defined Data Centres Using Stream Mining”, 2019 Transactions on Services Computing

[19] Kumar,ArvindandKaur,ParminderandSharma,Pratibha,“Asurveyon Hoeffding tree stream data classification algorithms”, 2015 CPUH-Res.

[20] A.Gudibanda,J.Ros-Giralt,A.CommikeandR.Lethin,“FastDetection of EFs with Dirichlet-Categorical Inference” 2018 Innovating the Network for Data-Intensive Science

[21] A. R. Mohammed, S. A. Mohammed and S. Shirmohammadi,“Machine Learning and Deep Learning Based Traffic Classification and Prediction in Software Defined Networking,” 2019 International Symposium on Measurements & Networking , 2019

[22] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: Measurements & analysis,” in Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf. (IMC), 2009, pp. 202–208.

[23] T.Benson,A.Akella,andD.A.Maltz,“Network traffic characteristics of data centres in the wild,” in Proc. 10th Annu. Conf. Internet Meas. (IMC), 2010, pp. 267–280

[24] K.Lou,Y.Yang,andC.Wang,“An elephant flow detection method based on machine learning,” in Proc. 7th Int. Conf. Smart Comput. Commun., 2019, pp. 212–220.

[25] W. Wang, Y. Sun, K. Salamatian, and Z. Li, “Adaptive path isolation for elephant and MFs by exploiting path diversity in datacentres,” IEEE Trans. Netw. Service Manage., vol. 13, no. 1, pp. 5–18, Mar. 2016.

[26] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic flow scheduling for datacentre networks,” in Proc. 7th USENIX Conf. Netw. Syst. Design Implement., 2010, pp. 89–92.

[27] Y.Tian,J.Liu,Y.-X.Lai,Z.-S.Bao,andW.-B.Zhang,“TPEFD:AnSDN- based efficient elephant flow detection method,” Chin. J. Netw. Inf. Secur., vol. 3, no. 5, pp. 70–76, 2017.

[28] F. Tang, H. Zhang, L. T. Yang and L. Chen, ”Elephant Flow Detection and Load-Balanced Routing with Efficient Sampling and Classification,” in IEEE Transactions on Cloud Computing, vol. 9, no. 3, pp. 1022-1036, 1 July-Sept. 2021, doi: 10.1109/TCC.2019.2901669.

[29] M. Afaq, S. Rehman, and W.-C. Song, “Large flows detection, marking, and mitigation based on sFlow standard in SDN,” J. Korea Multimedia Soc., vol. 18, no. 2, pp. 189–198, Feb. 2015.

[30] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, “OpenSample: A low-latency, sampling-based measurement platform for commodity SDN,” in Proc. IEEE 34th Int. Conf. Distrib. Comput. Syst., Jun. 2014, pp. 228–237.

[31] Y.Afek,A.Bremler-Barr,S.LandauFeibish,andL.Schiff,“Samplingand large flow detection in SDN,” ACM SIGCOMM Comput. Commun. Rev., vol. 45, no. 5, pp. 345–346, Aug. 2015.

[32] F. Tang, H. Zhang, L. T. Yang, and L. Chen, “Elephant flow detection and differentiated scheduling with efficient sampling and classification,” IEEE Trans. Cloud Comput., early access, Feb. 26, 2019, doi: 10.1109/TCC.2019.2901669.

[33] C.-Y.Lin,C.Chen,J.-W.Chang,andY.H.Chu,“Elephant flow detection in datacenters using OpenFlow-based hierarchical statistics pulling,” in Proc. IEEE Global Commun. Conf., Dec. 2014, pp. 2264–2269.

[34] W. Liu, W. Qu, Z. Liu, K. Li, and J. Gong, “Identifying EFs using a reversible MultiLayer hashed counting Bloom filter,” in Proc. IEEE 14th Int. Conf. High Perform. Comput. Commun. IEEE 9th Int. Conf. Embedded Softw. Syst., Jun. 2012, pp. 246–253.

[35] V. Mann, A. Vishnoi, and S. Bidkar, “Living on the edge: Monitoring network flows at the edge in cloud data centers,” in Proc. 5th Int. Conf. Commun. Syst. Netw. (COMSNETS), Jan. 2013, pp. 1–9.

[36] Mosab Hamdan 1, Bushra Mohammed 1, Uuman Humayun 1, Ahmed Abdelaziz 2, Suleman Khan 3, M. Akhtar Ali, Flow-Aware Elephant Flow Detection for Software-Defined Networks, IEEE Access

[37] Xu, Hong, and Baochun Li. "TinyFlow: Breaking elephants down into mice in data center networks." *2014 IEEE 20th International Workshop on Local & Metropolitan Area Networks (LANMAN)*. IEEE, 2014.