A Comparison of Machine and Statistical Time Series Learning for Encrypted Traffic Prediction

Qing He*, Georgios P. Koudouridis†, György Dán*
*Division of Network and Systems Engineering, KTH Royal Institute of Technology, Stockholm, Sweden
email: qhe@kth.se, guyri@kth.se
†RAN System Lab., Stockholm Research Center, Huawei Technologies, Stockholm, Sweden
email: george.koudouridis@huawei.com

***Abstract*—This paper studies the utilization of machine and statistical learning methods for predicting encrypted user traffic. To this end, a reference system model and performance metrics for traffic prediction have been defined for enabling on-line training. Based on a collection of representative traffic data sets including various video and web traffic, two different classes of predictors have been evaluated. Our results show that very good prediction performance can be achieved using long short-term memory (LSTM) recurrent neural networks at the price of a significant computational cost for training.**

***Index Terms*—traffic prediction, ARIMA, LSTM, recurrent neural networks, wireless networks**

I. Introduction

Identifying network traffic accurately can improve network management and resource utilization, and thus benefit Internet Service Providers (ISPs) and mobile network operators (MNOs). Before end-to-end encryption became prevalent in Internet traffic, Deep Packet Inspection (DPI) and port based classification have been widely used for classifying traffic. However, these methods may fail as more and more traffic are being encrypted and applications are using dynamic ports.

Encryption and the use of dynamic ports have triggered a recent shift of attention to machine learning-based (ML) traffic classification and prediction [1]. ML-based classification uses statistical features of network traffic flows, and has been shown to be promising for classifying encrypted data for a variety of purposes [2]. Given the wide variety of potential combinations of features, algorithms and objectives, the number of ways in which ML can be used for classification and prediction is indeed quite diverse, both concerning the traffic properties used for classification (features), the ML algorithms used and their results. Most notable are the results in [3] that focus on identifying traffic flows based on a few initial packets upon a flow arrives. Targeting early classification of encrypted traffic, the studied features were the packet sizes, and the inter-packet times, and their statistics, including the average, the standard deviation, the maximum, the variance, and the geometric mean, obtained from the first six packets of each flow. The results show that most classifiers achieve similar performance when using features created based on the packet sizes and inter-arrival times for early stage traffic identification. At the same time, the data sets used in [3] contain 10 years old traffic types that are different from contemporary mobile broadband traffic carried by the Dynamic Adaptive Streaming over HTTP (DASH) protocol. Although more recent studies cope with cellular traffic classification, which is typically performed as an off-line method [4], [5], there is lack of studies that address encrypted traffic prediction to be used as an on-line traffic prediction method.

In this paper the objective is to identify different traffic models and parameters that can be used for on-line predicting user data traffic. Also, traffic parameters are limited to those that can be extracted from encrypted data traffic. We utilize the traffic parameters to evaluate and compare two classes of prediction methods based on (i) time-series autoregressive integrated moving average (ARIMA), and (ii) machine learning based long short-term memory (LSTM) recurrent neural network. The methods are applied for the payload prediction of flows and aggregates consisting of video (e.g., DASH) and non-video traffic. As it is expected, our results demonstrate that video traffic is easier to predict than non-video traffic and that machine learning can advantageously be used for traffic prediction.

The rest of this paper is structured as follows: Section II describes the system model and formulates the traffic prediction problem. Section III provides a short description of the methods. The results of the evaluation of these methods are presented, compared and analyzed in Section IV, while the paper is concluded in Section V.

II. System Model and Problem Formulation

A. System model and notation

We consider traffic destined to a user in a network. Without loss of generality, we assume that the system starts its operation at time $t = 0$, and scheduling decisions, and hence prediction decisions, are taken periodically with a periodicity of δ. We refer to a sequence of packets from a source socket to a destination socket as a flow, and we denote by f^k the k^{th} flow in the traffic (counted based on arrival of the first packet of the flow). We denote by X^k_{-i} the features created based on the first i packets of flow f^k, by A^k the arrival time of flow k, by D^k the departure time of flow k, and by N^k the number of packets in flow f^k. Furthermore, we denote by a^k_i the arrival time of packet i in flow k and by b^k_i is size (in bytes). For an aggregate, we denote by $P(t, t+\tau)$ the number of bytes that arrive in the interval $(t, t+\tau]$. For flow f^k we denote by $P^k(t, t+\tau)$ the

978-1-7281-4905-9/20/$31.00 ©2020 IEEE

number of bytes that arrive in the interval $(t, t+\tau]$. Furthermore, we denote by $X(t_1, t_2)$ the features created based on the packets that arrived during $[t_1, t_2]$.

B. Prediction Problem

For the prediction problem let us define for $P^k(t, t+\tau)$ the predictor $p^k(X^k(A^k, t), t, \tau) \in \mathbb{R}_+$, and for $P(t, t+\tau)$ the predictor $p(X(0, t), t, \tau) \in \mathbb{R}_+$, i.e., the predictor for the aggregate. Values of interest for the prediction horizon in the considered application domain are $\tau \in \{10ms, 50ms, 100ms\}$. We define the prediction error for flow k as the squared sum of the prediction errors

$$U^k_{p(\tau)} = \frac{\sum_{s=\lceil A^k/\delta \rceil}^{\lfloor D^k/\delta \rfloor} (p^k(X^k(A^k, s\delta), s\delta, s\delta+\tau) - P^k(s\delta, s\delta+\tau))^2}{\lfloor (D^k - A^k)/\delta \rfloor}. \tag{1}$$

Based on $U^k_{p(\tau)}$ we can express the aggregate prediction error for the entire data set as

$$U_{p(\tau)} = \frac{\sum_{k=1}^{K} U^k_{p(\tau)} \lfloor (D^k - A^k)/\delta \rfloor}{\sum_{k=1}^{K} \lfloor (D^k - A^k)/\delta \rfloor}. \tag{2}$$

Similarly, we define the aggregate prediction error based on the predictor for aggregate data in $[T_1, T_2]$ as

$$U^a_{p(\tau)} = \frac{\sum_{s=\lceil T_1/\delta \rceil}^{\lfloor T_2/\delta \rfloor} (p(X(T_1, s\delta), s\delta, s\delta+\tau) - P(s\delta, s\delta+\tau))^2}{\lfloor (T_2 - T_1)/\delta \rfloor}. \tag{3}$$

It is worthwhile to note the difference between $U_{p(\tau)}$ and $U^a_{p(\tau)}$. While the former can only be computed if flow level information is available for prediction, the latter can be computed with and without flow level information. Thus, we can express (3) based on the flow level predictor errors

$$U^{a(f)}_{p(\tau)} = \frac{\sum_{s=\lceil T_1/\delta \rceil}^{\lfloor T_2/\delta \rfloor} (\sum_{k \in \mathcal{K}_s} (\mathcal{A}_k - \sum_{k \in \mathcal{K}_s} \mathcal{P}_k)^2}{\lfloor (T_2 - T_1)/\delta \rfloor}, \tag{4}$$

where $\mathcal{A}_k = p^k(X^k(A^k, s\delta), s\delta, s\delta+\tau)$, $\mathcal{P}_k = P^k(s\delta, s\delta+\tau)$, and $\mathcal{K}_s = \{k = 1, 2, \ldots, K : A^k \leq s\delta < s\delta+\tau \leq D^k\}$.

Consequently, we will focus on $U^a_{p(\tau)}$ and $U^{a(f)}_{p(\tau)}$. Doing so allows us to quantify the impact of having access to flow level information (e.g., through DPI of IP addresses and port numbers in the bearer) on the prediction accuracy. The objective of a predictor is to minimize the prediction error given the available information. We can thus formulate the prediction problem without having access to flow level information as follows.

$$\begin{aligned}
&\text{minimize} && U^a_{p(\tau)} && \text{(5a)}\\
&\text{subject to} && (3), \\
& && T_1 + \delta + \tau \leq T_2, && \text{(5b)}\\
& && p(X(T_1, s\delta), s\delta, s\delta+\tau) \geq 0, \\
& && s = \lceil T_1/\delta \rceil, \lceil T_1/\delta \rceil + 1, \ldots, \lfloor T_2/\delta \rfloor. && \text{(5c)}
\end{aligned}$$

When flow level information is available, we can formulate the prediction problem as follows.

$$\begin{aligned}
&\text{minimize} && U^{a(f)}_{p(\tau)} && \text{(6a)}\\
&\text{subject to} && (4) \\
& && T_1 + \delta + \tau \leq T_2, && \text{(6b)}\\
& && p^k(X^k(A^k, s\delta), s\delta, s\delta+\tau) \geq 0, \\
& && s = \lceil T_1/\delta \rceil, \lceil T_1/\delta \rceil + 1, \ldots, \lfloor T_2/\delta \rfloor. && \text{(6c)}
\end{aligned}$$

III. Data Traffic Prediction

We consider two classes of predictors for data traffic prediction focusing solely on short term traffic prediction using the performance metrics defined in Section II-B. The input to these predictors is a sequence values, one for each time period. Thus, the packet traces have to be converted to a sequence of values, from time stamps and packet lengths. There could be a variety of ways for doing the conversion, however, our approach was to compute the number of bytes that arrived during a time period, and use this integer number as the time series data. Based on this representation it is then straightforward to use the above models for predicting the number of bytes that would arrive in the next δms, based on past packet arrivals.

By using the notation introduced in Section II-B, we can denote by $P(t)$ the number of bytes that arrived during the t^{th} prediction interval, counting from T_1, and can express the predictor $p(t)$ for $P(t)$ with the following two models.

A. Autoregressive integrated moving average

Autoregressive integrated moving average (ARIMA) models are widely used for time series analysis, including forecasting, in econometrics and in engineering [6]. The formulation of ARIMA(p,d,q) is given as

$$\begin{aligned}
\Delta^d p(t) \quad = \quad & \alpha + \phi_1 \Delta^d P(t-1) + \phi_2 \Delta^d P(t-2) + \cdots \\
& \cdots + \phi_p \Delta^d P(t-p) + \\
& A_t - \theta_1 A_{t-1} - \theta_2 A_{t-2} - \cdots - \theta_q A_{t-q},
\end{aligned} \tag{7}$$

where A_t is a (weak) white noise process, and

$$\alpha = \left(1 - \sum_{i=1}^{p} \phi_i\right) \mu, \tag{8}$$

with μ denoting the process mean. If A_t follows the standardized student's t distribution, then

$$A_t = T_v \sqrt{\frac{v-2}{v}}, \tag{9}$$

where A_t is a Student's t distribution with degrees of freedom $v > 2$.

B. Long Short-Term Memory Recurrent Neural Network

Long short-term memory (LSTM) neural networks are recurrent neural networks that are able to learn long-term

Table I
RMSE results for 10ms, 50ms, and 100ms prediction intervals and for different types of traffic. The rows specify the training data sets, and hence the model, the columns stand for the testing data set.

Prediction interval	Model	Long DASH	Short DASH	Live Video	Non-video
10 ms	Long-DASH.ARIMA(5,0,0)	3397	4970	9159	59
	Short-DASH.ARIMA(5,0,0)	3397	4970	9159	59
	Live-Video.ARIMA(6,0,6)	3397	5490	9110	2131
	Non-Video.ARIMA(5,1,1)	3818	4970	9159	59
	Average bytes	1840	497	6033	38
50 ms	Long-DASH.ARIMA(1,0,0)	14777	18573	34173	172
	Short-DASH.ARIMA(1,0,0)	14777	18573	34173	172
	Live-Video.ARIMA(1,0,0)	31892	33062	13074	29558
	Non-Video.ARIMA(1,0,0)	14777	18573	34173	172
	Average bytes	9201	2485	30165	192
100 ms	Long-DASH.ARIMA(3,0,0)	25142	34607	65574	428
	Short-DASH.ARIMA(6,0,1)	28356	26254	65542	689
	Live-Video.ARIMA(4,0,0)	59264	62331	18556	56261
	Non-Video.ARIMA(1,0,0)	27154	35280	65626	362
	Average bytes	18403	4971	60330	385

dependencies in time series data [7]–[9], unlike feed-forward neural networks. The equations of LSTM are as follows.

$$i_t = \sigma\big(P(t)U^i + p(t-1)W^i\big) \tag{10a}$$
$$f_t = \sigma\big(P(t)U^f + p(t-1)W^f\big) \tag{10b}$$
$$o_t = \sigma\big(P(t)U^o + p(t-1)W^o\big) \tag{10c}$$
$$\tilde{C}_t = \tanh\big(P(t)U^g + p(t-1)W^g\big) \tag{10d}$$
$$C_t = \sigma\big(f_t * C_{t-1} + i_t * \tilde{C}_t\big) \tag{10e}$$
$$p(t) = \tanh(C_t) * o_t \tag{10f}$$

Here t corresponds to a timestamp and i, f, and o, are the input, forget and output gates respectively. They have the exact same equations but with different parameter matrices W, which is the recurrent connection at the previous hidden layer and current hidden layer. U is the weight matrix connecting the inputs to the current hidden layer, σ is the sigmoid activation function for forget gate and tanh is the hyperbolic tangent activation function for cell state. $\tilde{C}$ is a candidate hidden state that is computed based on the current input and the previous hidden state. C is the internal memory of the unit, and is a combination of the previous memory, multiplied by the forget gate, and the newly computed hidden state, multiplied by the input gate. The predictor $p(t)$ is the output hidden state, computed by multiplying the memory with the output gate.

IV. Prediction Evaluation

A. Data sets and Evaluation Realization

The traffic of the computer was generated using various web browsers, Chrome and Firefox, while browsing a variety of web pages, among them web pages with video content, including YouTube, FaceBook, LinkedIn. For capturing incoming traffic on a desktop computer, Wireshark was used [10]. In total we captured four traces of 60 minutes duration each. We refer to these as the *Long DASH*, *Short DASH* , *Live Video* , and *Non-video* data sets. The captured data consists of packet arrival time stamps and packet sizes. In addition, we also capture the IP header information.

For each data set, we used 80% of the data for training and the remaining 20% of the data for testing. For the ARIMA and LSTM predictors we train a model on each data set, and test all four the trained models on all data sets for the same prediction interval.

B. Prediction Results

We start the evaluation with the ARIMA model, and we consider the prediction intervals of $10ms$, $50ms$ and $100ms$. Table I shows the root mean square error (RMSE) obtained using the ARIMA models, and as a comparison it also shows the average number of bytes per prediction interval. The rows specify the training data sets, and hence the model, while the columns stand for the testing data set. Based on the ACF and the PACF of the data sets we identified the best fit model for each data set, and used maximum likelihood estimation for finding the optimal parameters. The ACF and PACF are commonly used for analyzing the correlation structure of time series and help in identifying the order of the ARIMA model to be used for prediction [11]. As an example for the prediction interval of $100ms$, the time series, the ACF, and the PACF for the four data sets are shown in Figure 2. The best fit ARIMA models for the 100ms prediction interval are all different, like for 10ms and unlike for 50ms. For the 50ms prediction interval all models are ARIMA(1,0,0), but the parameters of the models trained on different data are likely to be different, hence even in this case we have one model per data set. For each prediction interval, Table I shows the RMSE of the ARIMA models when applying the four models on the four data sets. As a basis for comparison, the last row shows the ground truth of the average number of bytes per prediction interval for the data sets.

Next, we present results for the LSTM model, using a small network that has an input layer with one input, a hidden layer with one neuron, and an output layer that makes a single value prediction. The LSTM network is trained for 200 epochs and a batch size of 1 is used. As the output given by a recurrent neural network may vary with different initial conditions, we repeat the experiments for each case multiple times, and show the

Table II
LSTM results for 10 ms, 50 ms ans 100 ms prediction intervals and different traffic types. The rows specify the training data sets, and hence the model, the columns stand for the testing data set.

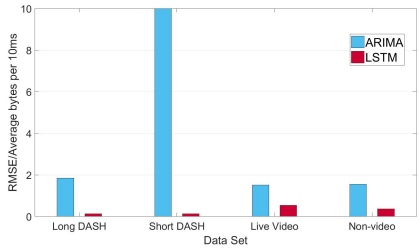
Prediction Interval	Model	Long DASH	Short DASH	Live Video	Non-video
10 ms	Long-DASH.LSTM	229	256	3820	43
	Short-DASH.LSTM	237	65	5821	35
	Live-Video.LSTM	17410	9532	3210	1599
	Non-Video.LSTM	327	96	3211	14
	Average bytes	1840	497	6033	38
50 ms	Long-DASH.LSTM	1021	235	16501	110
	Short-DASH.LSTM	4880	226	18600	418
	Live-Video.LSTM	61354	58996	5908	39002
	Non-Video.LSTM	1072	313	38760	57
	Average bytes	9201	2485	30165	192
100 ms	Long-DASH.LSTM	632	23384	54323	588
	Short-DASH.LSTM	2343	4580	43854	1321
	Live-Video.LSTM	12602	78532	23865	6403
	Non-Video.LSTM	7136	27471	69224	276
	Average bytes	18403	4971	60330	385

average RMSE (excluding the extreme values) as an indication of the expected performance of the model on unseen data on average.

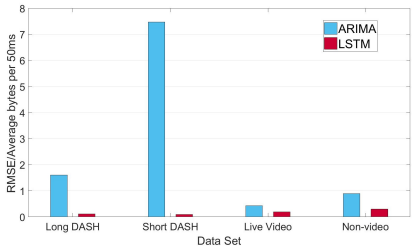
Table II shows LSTM prediction results for prediction intervals of 10ms, 50ms, and 100ms. Again, the row specifies the training data set, and hence the model, the columns stand for the testing data set. For the prediction interval of 10ms, the RMSE results shown are the averages based on 8 test runs. The results show that the LSTM model performs well for the data set of the same type of traffic as the model is trained for, but the predictor performance could be very poor if an LSTM model is applied to data sets that are dominated by different types of traffic than what it was trained for. For the prediction intervals of 50ms and 100ms, the RMSE results are the averages based on 10 test runs. The results confirm the observation that for all prediction intervals the LSTM models perform well on their corresponding test data sets but not on the other test data sets.

For each data set, we normalized the RMSE values, which have been obtained by the model trained on this data set, by the actual average number of bytes per prediction interval, which have been obtained from the ground truth. We present the normalized RMSE in Figure 1. Based on the results we can make the following observations. First, the results obtained using the ARIMA models are rather poor. On the one hand, this is due to the high variability of three of the four time series. On the other hand, this is also due to that the ARIMA model assumes that the time series is stationary, while the considered data sets are not. Secondly, the results using the LSTM network are far superior compared to the ARIMA models. It is important to note, however, that training the LSTM model is associated with computational cost. For instance, it takes about ten minutes to train a simple LSTM model with 200 epochs on a NVIDIA Quadro M2000 GPU (768 cores and 4GB of memory). Evaluations with more epochs could further improve the prediction accuracy, but with higher computational cost.

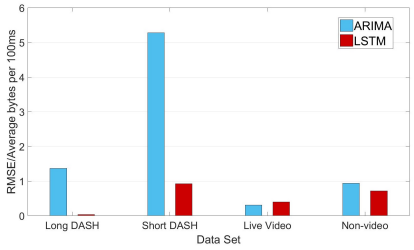
In addition, comparing the results for ARIMA (in Table I)



(a) Prediction interval of 10ms



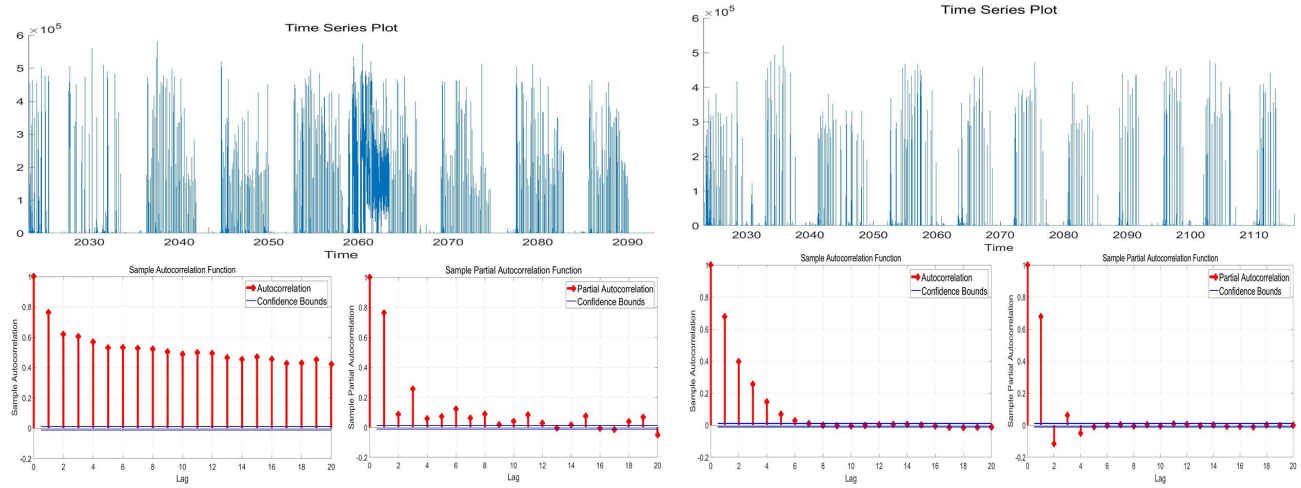
(b) Prediction interval of 50ms



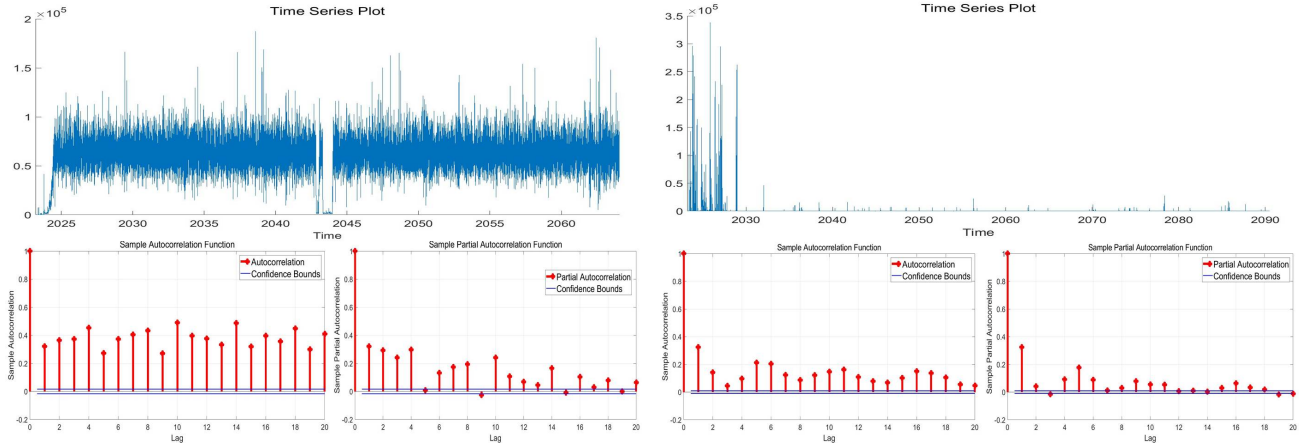
(c) Prediction interval of 100ms

Figure 1. Normalized RMSE for data sets of different prediction intervals.

and for LSTM (in Table II), though it may happen that in a few cases a model trained on one data set performs well on other data sets, in general there does not seem to exist a one-size-fit-all predictor for the data sets dominated by different types of traffic. As a final general remark, although bursts following a regular pattern (e.g., in time) would be possible to predict by simple predictors (e.g., ARIMA), our results show



(a) Long DASH (b) Short DASH



(c) Live Video (d) Non-video

Figure 2. Time series, ACF and PACF of four homogeneous data sets with $\tau = 100ms$.

that in practice the traffic is not regular enough for these simple predictors to predict traffic bursts.

V. Conclusions

The results for traffic prediction show that the traditional ARIMA model does not perform well due to the non-linear and non-stationary characteristics of the data traffic. The ML-based approaches such as LSTM could significantly outperform the ARIMA models, at the cost of significant training time and, somewhat increased prediction complexity. In an on-line setting the predictor to be used may have to be adapted based on the performance of the available predictors. In future work we plan to explore a bigger variety of predictors, and the possibility of choosing predictors dynamically for traffic of mixed data types. Furthermore, we will investigate whether retraining predictors online is a feasible solution.

References

[1] M. Finsterbusch, C. Richter, J.-A. M. E. Rocha, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Communications Surveys & Tutorials*, pp. 1135–1156, 2014.

[2] J. Garcia, T. Korhonen, R. Andersson, and F. Vastlund, "Towards video flow classification at a million encrypted flows per second," in *Proc. of IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, 2018, pp. 818–824.

[3] L. Peng, B. Yang, Y. Chen, and Z. Chen, "Effectiveness of statistical features for early stage internet traffic identification," *International Journal of Parallel Programming*, pp. 1–17, 2015.

[4] K. J, Hajjar, and D.-V. J. A. A, "A multilevel taxonomy and requirements for an optimal traffic-classification model," *International Journal of Network Management*, pp. 101–120, 2014.

[5] P. Velan, M. Cermak, P. Celeda, and M. Drasar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, pp. 355–374, 2015.

[6] MathWorks, "Time econometric modeler," 2019, https://se.mathworks.com/help/econ/econometric-modeler-overview.html.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation, MIT Press*, 1997.

[8] Z. Zhao, W. Chen, X. Wu, P. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," in *Proc. of IET Intelligent Transport Systems*, 2017.

[9] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, pp. 654–669, 2018.

[10] Wireshark, https://www.wireshark.org/.

[11] T. Anderson, "The statistical analysis of time series," *J. Wiley&Sons*, 1971.