Enhancing Quality of DASH Video with LSTM Throughput Prediction

Arkadiusz Biernacki (arkadiusz.biernacki@polsl.pl)
Silesian University of Technology

Research Article

Keywords: Traffic prediction, Artificial neural networks, Adaptive video

Posted Date: April 13th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1546677/v1

Noname manuscript No.
(will be inserted by the editor)

Enhancing Quality of DASH Video with LSTM Throughput Prediction

Arkadiusz Biernacki

the date of receipt and acceptance should be inserted later

Abstract The quality of streaming video depends on accurate estimation of network throughput, which is challenging in mobile networks due to a highly volatile environment. We used data traces obtained from measurements in 4G and 5G mobile networks, supplied them to an LSTM network and received a prediction of throughput for the next four seconds. Next, we took three dynamic adaptive streaming over HTTP (DASH) algorithms and replaced their default throughput estimation based on moving averages with the LSTM prediction. As the experiment shows, the traffic prediction improved the effectiveness of network capacity utilisation and stability of video playback between 5% and 25% compared to the default estimation. Our approach relies only on past throughput measurements, does not require any modifications to network infrastructure or protocols and is implementable in any DASH player.

Keywords Traffic prediction · Artificial neural networks · Adaptive video

1 Introduction

Streamed multimedia are susceptible to highly variable packet delays, losses and fluctuating network capacity. Therefore, streaming high-quality video to users, especially in mobile networks, remains a challenge. Modern mobile networks offer high maximum capacity; however, due to the nature of wireless transmission, inherent variability in signal strength, interference, noise, and user mobility, their throughput usually has high variability. Content providers split a video file into segments and encode every segment with a different bit rate to cope with this unstable network environment. Such an approach allows a player to switch the media bit rate (and hence the quality) after downloading each segment, thus adapting video quality to the current network conditions. This adaptation is critical to ensure good quality of experience (QoE) for Internet video.

The rate-adaptation algorithm is the main component and a challenge in a DASH system because the client has to estimate, and sometimes even predict, network conditions or the dynamics of the available throughput. Furthermore, the video player has to control the filling level of its local buffer to avoid underflows resulting in playback interruptions. The rate adaptation algorithm might work well when the player does not share a connection with other flows, network resources are stable and do not fluctuate. Since the client measures network throughput using usually a moving

A. Biernacki
Silesian University of Technology, Institute of Computer Science,
Akademicka 16, 44-100 Gliwice, Poland;
E-mail: arkadiusz.biernacki@polsl.pl

average, this estimate differs from real throughput. This mismatch results in undesirable behaviour of streaming algorithms which can be both too conservative and too aggressive. A solution to this problem is to predict the future network throughput. As the video segment usually have several seconds, the prediction is short-term and does not rely on seasonality. Prediction can improve the efficiency of bit rate selection, which translates to lower startup delay and offers better mid-stream bit rate adaptation. Researchers estimate that the applied prediction achieves between 68% - 89% of the optimal quality. The exact numbers are dependent on the playback algorithms employed in a DASH player [1].

In our work, we predict throughput using traces from 4G and 5G networks from static and mobile usage scenarios. In the next step, we replace the default methods of throughput estimation in three DASH algorithms with the obtained predicted results. Finally, we show that the DASH algorithms achieve better performance when equipped with prediction compared with reliance only on an average of network measurements. Our approach does not require any modification to network protocols or the DASH standard and its components. The solution only affects users' players where a more elaborate algorithm based on an LSTM artificial neural network (ANN) replaces the default throughput estimation. With the information provided by the player's measurement module, the algorithm can estimate the expected throughput for the next few seconds. Because an LSTM ANN requires some computation needed to learn and continually update its knowledge, this process increases the usage of memory and CPU of a video player.

Taking the above into account, the contributions of the paper are: the prediction of 5G network throughput at the application level in static and mobile scenarios, the comparison of the prediction quality with the 4G trace, and the application of the prediction made by an LSTM ANN to improve quality of video for three DASH players.

2 Related works

The existing literature on network throughput prediction is extensive and focuses particularly on the general internet traffic, which aggregates flows originating from different kinds of applications: multimedia streaming, web browsing, file downloading or peer-to-peer networking. The latest broad review of prediction methods and a comparison of some of them can be found in [2]. A number of research studies used linear regression, autoregressive integrated moving average (ARIMA), ARIMA with an explanatory variable (ARIMAX), multiple linear regression, support vector regression, GARCH processes and many others techniques derived from statistics. These methods are conventionally called traditional and are considered descriptive in nature, aimed at analyzing datasets with finite, countable, and explainable predictors. The authors of [3] made a comprehensive comparison of the traditional methods using a 3G/HSPA data set.

The newer prediction schemes based on machine learning are sometimes used in accordance or opposition to the traditional methods, and there is an ongoing debate about their accuracy and computational requirements [4]. ANNs, considered as a non-traditional approach, play a role in network traffic prediction mainly due to their universality and accuracy. Historically, the popularity gained simple ANNs with a single hidden layer because versions with multiple layers, called deep ANNs, were harder to train. After researchers overcome the training issues, deep ANNs became popular and gradually replaced single-layered ANNs as a tool for the solution of more complex problems. Because deep ANNs added more complexity and nonliterary to the network traffic model, their prediction accuracy increased. Oliveira's comparative study [5] discusses both approaches in the context of network traffic prediction.

From the variety of deep ANNs, LSTM ANNs are one of the possible choices and are applied readily due to their capability to capture long-range dependences characteristic of aggregated network traffic, i.e. traffic composed of multiple flows. For example, an LSTM ANN was used to identify recurrent patterns of various metrics to predict traffic in cellular networks in [6]. The work

focuses on long-horizon prediction counted in days. In [7], the authors enhanced their LSTM-based prediction with an autocorrelation function which captured the traffic cycles on a daily scale with a prediction span between 5 and 60 minutes. In [8], the authors proposed to remove extreme values by scaling a traffic trace. Their LSTM ANN had four hidden layers with a time window set to 32 units or more. The authors did not provide a prediction horizon, however, the data granularity was more than one minute.

In the case of multimedia applications, the prediction of a single traffic flow is more important and challenging than aggregated traffic [9]. A single flow, especially if transmitted through an unstable mobile environment, can have much variability and sudden changes compared with an aggregated trace smoothed by statistical multiplexing. Hence, the models of the aggregated traffic are usually inapplicable to describe a single flow generated by a DASH system streaming video segments in an ON-OFF fashion which contributes to significant bursts in the transmission. As single flow prediction is a more challenging task, there are fewer works dedicated to this topic and their authors focus mainly on a short-time prediction span. In [10] the authors applied LSTM and Gated Recurrent Units (GRU) ANNs to predict traffic flow obtained from a sensor network for a five minutes time horizon. They achieved better results compared with the ARIMA processes. A comparative study [11] showed that for cellular network traffic, LSTM provides better prediction than a stacked autoencoder. Unfortunately, the authors did not provide information about the traffic traces used in the experiment.

Except for the historical throughput, an LSTM ANN sometimes operates on additional data to improve the results; for example, the authors of [12] used a complex LSTM ANN with five hidden layers and a multivariate dataset containing information about all downlink and uplink traffic for a given base station. Other works use a meta-learning scheme including a set of predictors, each optimized to predict a particular kind of traffic. The study by He et al. [9] applied this technique to improve traffic scheduling in base stations with a prediction horizon set to 100 ms. The authors of [13] stated that for the prediction of flow throughput the best result achieved Random Forest technique which outperformed a deep ANN. They tested different time windows spanning from 0.03s to 4s on nearly 1 million flows containing traces from miscellaneous networks and applications. Besides flow history data, the study used other attributes related to TCP: window size, window scale or segments retransmissions. Azari et al. [14] compared an LSTM ANN and ARIMA by examining the effect of different parameters of these prediction models. The simulation results proved the superiority of LSTM over ARIMA, particularly when the training time series was long enough. The same technique was used in [15] where the authors concluded that while historical throughput is a principal feature, using it alone can lead to inferior performance compared to that using all the features. Emphasising the importance of combining upper- and lower-layer information, the authors claimed that using the four most important traffic features lead to almost the same prediction performance as when using all of them. Another short-term prediction is proposed in [16] where He et al. evaluated the ARIMA and LSTM for the prediction of both single and multiple flows taking into account video and non-video traffic. Using a prediction span between 10 and 100 ms, they found that LSTM ANNs achieved good results at the cost of more training computations.

Most DASH algorithms use a simple prediction technique and estimate future throughput as an average of past measurements [17]. In the more advanced approach, video players try to optimise playback by employing methods from different domains. Some researchers apply prediction of network throughput and show that it outperforms existing algorithms, which do not apply predictive methods, while others observe that prediction alone is not sufficient and should be combined with other techniques like buffer control [1]. Therefore, developers propose complex playback strategies, where the throughput prediction is only part of them. In [18], the authors focus on inaccuracies in throughput measurements occurring due to idle periods between the chunks causing sub-optimal adaptation decisions. The authors show that a simple prediction scheme based on adaptive filtering can improve the quality of the streaming video. Raca et al. in [19] compared several prediction schemes, among them Support Vector Machines, Random Forest and LSTM ANN and applied

them to a single DASH algorithm. Interestingly, the authors reported that a DASH player supported by the best predictor achieved nearly the same performance as in the case of the ideal prediction. Besides application data, the model relied on network data, reaching as deep as the physical layer, which may be a reason behind the extraordinary high accuracy of the prediction. In [20], we proposed to employ (F)ARIMA processes, shallow ANNs and recursive auto-encoder to enhance the performance of a DASH system.

In this work, we operate on traces gathered from 4G and 5G networks. As yet, the analysis of the traffic generated by the latter network has limited coverage in the literature. Our approach does not require any modifications to the network infrastructure or the TCP stack, as we solely rely on network measurements performed on the application level. Consequently, our work shows the impact of the prediction on video QoE without the support of any other information, which is hard to gather and require profound changes in a DASH system. Contrary to the majority of the works which aim at either short or long time horizons, our prediction span is 4 s, which is suitable for a DASH system.

3 Theoretical background

3.1 Video Streaming Model

A video stream can be modelled as a set of consecutive data chunks $\{c_i\}$ where $i \in \{1, 2, \dots N\}$. Every chunk contains L seconds of video encoded with different bit rates. Hence, the total length of the video stream is $N \times L$ seconds. A video player can choose to download the video segment c_i with the bit rate $q_i \in \{Q_{\min}, \dots, Q_{\max}\}$ [bit/s], where $Q_{\min}$ and $Q_{\max}$ are the bounds of the set of all available bit rate levels. The amount of data in segment c_i is then $L \times q_i$. The higher the bit rate is selected by a video player, the higher is also video quality perceived by a user.

The video segments are downloaded into a player buffer, where they wait for the playback. At the time t_i, a video player starts to download the video chunk c_i. The downloading time depends on the selected bit rate q_i as well as average network throughput r_i. At time t_{i+d} the chunk c_i is completely downloaded and at time t_{i+1}, where $t_{i+d} \leq t_{i+1}$, the video player starts to download the next chunk c_{i+1}. We denote by $b_i \in [0, B_{\text{full}}]$ the length of video, counted in time units, stored in the player buffer at the time t_i. The buffer size B_{full} depends on a particular video player and user's storage limitations. If we denote by r_i [bit/s] average network throughput at time $\Delta t_i = t_{i+d} - t_i$, then we have:

$$t_{i+d} = t_i + \frac{Lq_i}{r_i},$$

where

$$r_i = \frac{1}{\Delta t_i} \int_{t_i}^{t_{i+d}} r \, dt. \tag{1}$$

While the video segments are being downloaded and played, the amount of data in the player's buffer changes. After the segment c_i is downloaded, the amount of data increases by L seconds. Meanwhile, the buffer occupancy decreases as the user watches the video. Hence, the evolution of the buffer is described as:

$$b_{i+1} = \max[(b_i - t_{i+1} + t_i), 0] + L.$$

If $b_i < t_{i+1} - t_i$, the buffer becomes empty while the video player is still downloading the chunk c_i what freezes the video.

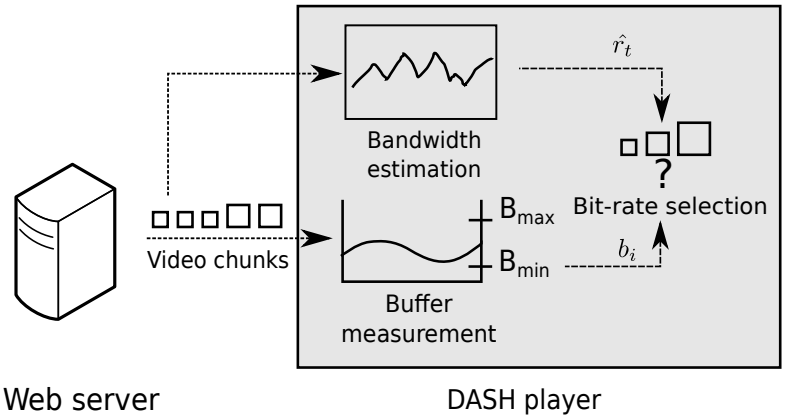


Fig. 1: Adaptation strategy for video. The algorithm selects the quality of the video taking into account the level of player buffer and the prediction of network throughput obtained from its past measurements.

3.2 Adaptation strategies

When at the time t_{i+1}, a video player selects the bit rate q_{i+1}, it has only knowledge of the past throughput $\{r_t, t \leq t_i\}$, while the future one $\{r_t, t > t_i\}$ is not known. However, the video player may use a predictor defined as $\{\hat{r_t}, t > t_i\}$ to get insight into the hypothetical future throughput. Furthermore, also an inspection of the buffer level may provide supportive information to the player. Taking into account the future throughput estimation and occupancy of the buffer, the player selects the quality of the next segment c_{i+1}, see Fig. 1, as follows:

$$q_{i+1} = f(b_i, \{\hat{r_t}, t > t_i\}).$$

In our work, the selection of the video bit rate is based on three exemplary approaches. The first approach employs a simplified version of the algorithm implemented in MSS and is extensively described in [21]. The second approach is based on implementation of ExoPlayer, which is a media player platform for Android developed by Google [22]. The third solution is described in [23] and we will refer to it as *Tian2016*.

The first algorithm selects the bit rate of a video chunk as follows

$$q_{i+1} = \begin{cases} q_i^+ & b_i > B_{\max} \quad (2a) \\ q_i^- & b_i < B_{\min} \quad (2b) \\ q_i^- & b_i \in [B_{\min}, B_{\max}] \wedge (q_i > \hat{r}_i) \quad (2c) \\ q_i & b_i \in [B_{\min}, B_{\max}] \wedge \hat{r}_i - d < q_i < \hat{r}_i \quad (2d) \\ q_i^+ & b_i \in [B_{\min}, B_{\max}] \wedge (q_i < \hat{r}_i - d) \quad (2e) \end{cases}$$

The notation q_i^+ and q_i^- denotes an increase or a decrease of the video bit rate q_i by one level. The basic idea of the buffer-based part of the algorithm is to select video bit rate based on the amount of data available in a player's buffer. Thus, when the buffer reaches $B_{\max}$, the system is allowed to increase the bit rate (2a). Similarly, when the playback drains the cached data, the selected quality is reduced if the buffer shrinks below the threshold $B_{\min}$ (2b). When the buffer level is between the thresholds $B_{\min}$ and $B_{\max}$, the player uses throughput estimation $\hat{r}_i$ to select the quality level. If the estimated throughput $\hat{r}_i$ is smaller than the current video bit rate q_i, the algorithm decreases the video quality (2c). If the estimated throughput is not sufficiently high, the bit rate remains

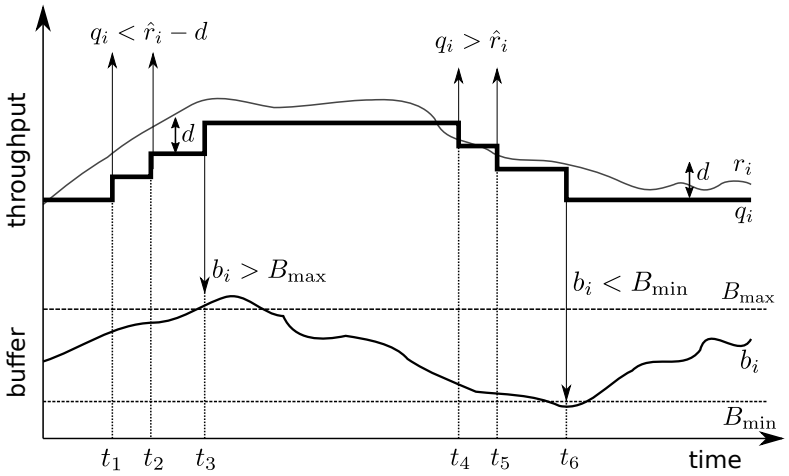


Fig. 2: An example operation of a simplified version of the MSS algorithm. At times t_1 and t_2, network throughput is high enough to increase the video bit rate (2e). At t_3 the buffer level, which is above the threshold, allows increasing the video bit rate again (2a). At times t_4 and t_5, network throughput drops what causes the video bit rate to decrease (2c). At time t_6, the buffer level drops below the threshold, and the video bit rate is reduced again (2b). After t_6, network throughput is not high enough to increase the video-bit rate (2d).

unchanged (2d). If the estimated bit rate surpasses the throughput at least by d bits/s, the video quality is increased (2e). The parameter d marks a region of network throughput for which there is no need to switch video quality to a higher level. As a result, the parameter plays a stabilising role and prevents switching the quality levels too frequently, which could harm the overall video quality perceived by users. The constants $Q_{\min}$ and $Q_{\max}$ define a range of available quality levels. Fig. 2 illustrates the above-described idea, which we implemented in the open-source software described in [24]. Following [25], we set the player's buffer size B_{full} to 35 s, and the thresholds $B_{\min}$ and $B_{\max}$ to 40% and 80% of the buffer size.

The second algorithm is a part of the ExoPlayer library. Its input parameters are a buffer size and throughput estimation. When deciding about the quality of the next segment, the algorithm calculates the best representation that fits in the currently available bandwidth according to the formula:

$$q_{i+1} = \begin{cases} \max(Q) & q_{i+1} \leq \lambda \hat{r_i} \quad (3a) \\ q_i & \hat{r_i} > r_{i-1} \wedge b_i < B_{\min} \quad (3b) \\ q_i & \hat{r_i} < r_{i-1} \wedge b_i > B_{\max} \quad (3c) \end{cases}$$

The algorithm scans the available qualities Q and selects the representation with the highest average bit rate lower than or equal to the estimated throughput r modified by a factor λ, as presented in (3a). The algorithm will switch to the best achievable video bit rate if it differs from the current bit rate, excluding two scenarios: network throughput increases but the buffer level remains low (3b) or network throughput decreases but the buffer level remains high (3c). For ExoPlayer, the default values for $B_{\min}$ and $B_{\max}$ are 10 s and 25 s.

The third approach, which we will refer to as *Tian2016*, tries to avoid video rate fluctuations triggered by short-term TCP throughput variations and estimation errors. Furthermore, it increases video rate smoothly when the available network bandwidth is consistently higher than the current video bit rate and decreases video bit rate more aggressively when network throughput drops. The solution is based on the proportional-integral-derivative (PID) controller – a control loop with a

feedback mechanism. The mechanism adjusts a control output based on the difference between a set point and a measured process variable. The value of the controller output is transferred to the system input. The authors claim that the proposed approach is highly responsive to congestion level shifts and can maintain a stable video rate during short-term bandwidth variations occurring in mobile networks. We took the parameters for the algorithm from [23] from sections V and VI-D.

Different systems use different video segment lengths L which usually range from 2 s to 10 s; for example, the MSS algorithm uses 2-5 seconds [25]. In our work, we set the duration of the video segment to 4 s.

The presented adaptive strategies are examples of many potential strategies and we use them solely for an evaluation purpose. In-depth reviews of DASH algorithms can be found in [26][27].

3.3 Quality measures

The main goal of DASH systems is to improve the QoE, which depends on the bit rate of video chunks. The choice of the bit rate involves a few potentially conflicting issues. Firstly, a user expects that a video player will provide the highest possible bit rate available for the given constraint of network throughput. Secondly, the player should keep the playback as smooth as possible by avoiding frequent bit rate switches. Thirdly, the video player should avoid its buffer starvation to minimize the time which the video freezes on a user's screen. Having identified the three main components of QoE, namely bit rate, bit rated variation and rebuffering time, we combine them into a single measure, where the QoE is calculated as a weighted sum of the parameters of each video segment in the range from 1 to N:

$$\mathrm{QoE} = \sum_{i=1}^{N} q_i - \lambda \sum_{i=1}^{N-1} |q_{i+1} - q_i| - \mu \sum_{i=1}^{N} (K(b_i)),$$

where λ and μ are positive weights corresponding to video quality variations and rebuffering time. To count the buffer's starvation periods, we introduce the penalty function $K(b_i) = 1, b_i = 0$ and $K(b_i) = 0, b_i > 0$.

Video stalls disturb users more than bit rate switches do; thus, the value of μ is usually much higher than λ. A small λ indicates that bit rate switches are less nuisance for a user, while a large μ implies that the user is more concerned about rebuffering.

To obtain time- and video-independent measures with possibilities of comparison with different algorithms, we adapt from [28] a normalised QoE model

$$\mathrm{nQoE} = \frac{\mathrm{QoE}}{\mathrm{QoE\_ref}}, \tag{4}$$

where QoE_ref is the QoE of another benchmark algorithm.

4 Prediction of network throughput

4.1 The prediction problem

In our approach to network traffic prediction, we have measurements of the previous throughput $\{r_t, r_{t-1}, \ldots, r_{t-n}\}$, and we wish to predict the value of r_{t+m}, $m > 0$. For this purpose, we apply the predictor $\hat{r}_{t+m}$ which employs the observations of the past throughput measurements

$$\hat{r}_{t+m} = \phi(r_t, r_{t-1}, \ldots, r_{t-n}). \tag{5}$$

The problem is to choose ϕ which minimizes the difference between $\hat{r}_{t+m}$ and r_{t+m}.

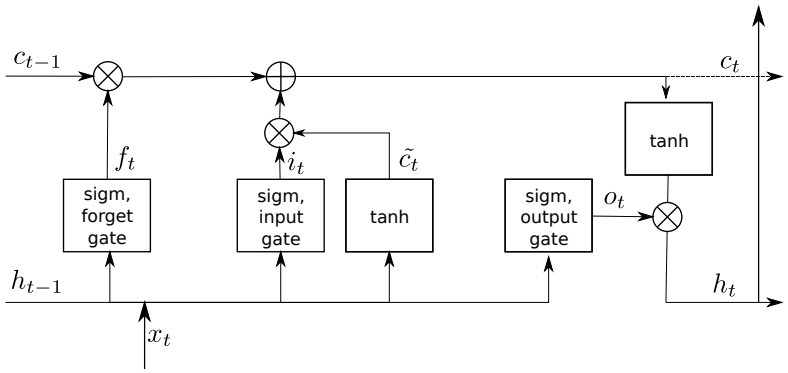


Fig. 3: Structure of LSTM Memory Block: x_t is the vector of input signals at the time t, C_{t-1} and C_t represent cell states at the previous $t-1$ and actual t times, h_{t-1} and h_t are output signals of the cell at $t-1$ and t times, b represents the bias. The symbol $\times$ represents the multiplication and $+$ summation of signals.

Typically, a DASH algorithm predicts network throughput for the nearest future r_{i+1} using an average of past throughput measurements [17]

$$\hat{r}_{t+1} = \frac{1}{n}\sum_{k=1}^{n} r_{t+1-k}. \tag{6}$$

According to [27], DASH players apply the estimator (6) or its variation based on weighted, exponential or harmonic moving average. In our work, we improve the estimation by replacing the simple computation (6) with an LSTM ANN.

To measure the accuracy of the prediction algorithm, we chose the Normalised Root Mean Square Error (NRMSE)

$$NRMSE = \frac{1}{\overline{r}}\sqrt{\frac{\sum_{t=1}^{N}(r_t - \hat{r_t})^2}{N}}, \tag{7}$$

The NRMSE represents the squared difference between predictions $\hat{r}_t$ and observed values r_t divided by the number of observations N. An average of observed values $\overline{r}$ normalises the calculation result. The lower is the NRMSE value, the better the model describes empirical data. The normalisation allows us to compare predictions among traces with different throughput levels.

4.2 Long short-term memory networks

An LTSM is a recurrent ANN which can learn long-range dependences in data. LSTM ANNs respond to the short-term memory problem of recurrent ANNs. Namely, a recurrent ANN has difficulties retaining data from earlier computations because it misses information about long-memory dependences. An LSTM ANN handles this problem by allowing its memory cells to select when to forget certain information, thus achieving the optimal time lags for time series problems. For this purpose, a network uses a memory block which contains cells with self-connections, memorizing the temporary state and three adaptive, multiplicative gates: the input, the output and the forget to control information flow in the block – see Fig. 3. The gates can learn to open and close, enabling LSTM memory cells to store data over long periods, which is desirable in time series prediction with long temporal dependency. Multiple memory blocks form a hidden layer. An LSTM ANN consists of an input layer, at least one hidden layer and an output layer.

A cell operates in two succeeding time points: the actual t and the previous $t-1$ representing the state of the cell at the previous point of time. A cell operation, denoted by CL, is a mathematical function that takes three inputs and generates two outputs:

$$(h_t, c_t) = \mathrm{CL}(h_{t-1}, c_{t-1}, x_t),$$

where x_t is the value of the time series at the time t, c_{t-1} is a memory signal of the cell from the previous time $t-1$, and h_{t-1} is an output signal of the cell at the preceding time $t-1$. At the same time, the cell generates two output signals: C_t – memory cell state at the time t and h_t – output signal of the cell at the time t. The hidden layer cells pass the signal to the output layer.

At every time step, the element x_t of a time sequence enters the cell. Both output signals h_t and c_t leave the cell at time t and return to that same cell at a time $t+1$. The cell can multiply or add signals, what is denoted by the symbols $\times$ and $+$. Cell's gates also use two functions: *sigm* and *tanh*. The first represents the sigmoid function which outputs numbers between zero and one ($\mathrm{sigm} : R \to [0, 1]$), describing how much of each signal to let through. The value of zero transfers nothing, while a value of one transfers everything. The second function represents the hyperbolic tangent $\tanh : R \to [-1, 1]$) and denotes which fraction of the candidate value $\tilde{c_t}$ updates the cell's state c_{t-1}.

An entering signal simultaneously reaches all three cell's gates. The *forget gate* decides which part of the memory from the previous state transfers to the current state. The gate multiplies the signal and projects it into a unit interval using a sigmoid function

$$f_t = \mathrm{sigm}(W_f[h_{t-1}, x_t] + b_f).$$

The remaining two gates determine which new information to store in the cell. Firstly, the *input gate* decides which values to update, transforming them into a unit interval. Next, the *tanh* layer creates a new candidate value $\tilde{c_t}$ and adds it later to the cell state.

$$\begin{aligned} i_t &= \mathrm{sigm}(W_i[h_{t-1}, x_t] + b_i), \\ \tilde{c_t} &= \tanh(W_C[h_{t-1}, x_t] + b_C). \end{aligned}$$

Finally, the new state c_t replaces the old one c_{t-1}: the old state c_{t-1} multiplies the *forget gate* value f_t, and the candidate result $\tilde{c_t}$ multiplies the *input gate* value i_t

$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c_t}. \tag{8}$$

Equation (8) controls the amount of information c_t which will form the state of the cell at the time t.

In the cell's right-side, a multiplicative operation generates the output signal

$$h_t = o_t \times \tanh(c_t),$$

which is the product of the actual memory signal c_t computed in (8) and the signal

$$o_t = \rho(W_o[h_{t-1}, x_t] + b_o),$$

generated in the *output gate*. The output signal h_t describes how much information the cell transmits to other cells at the next time point.

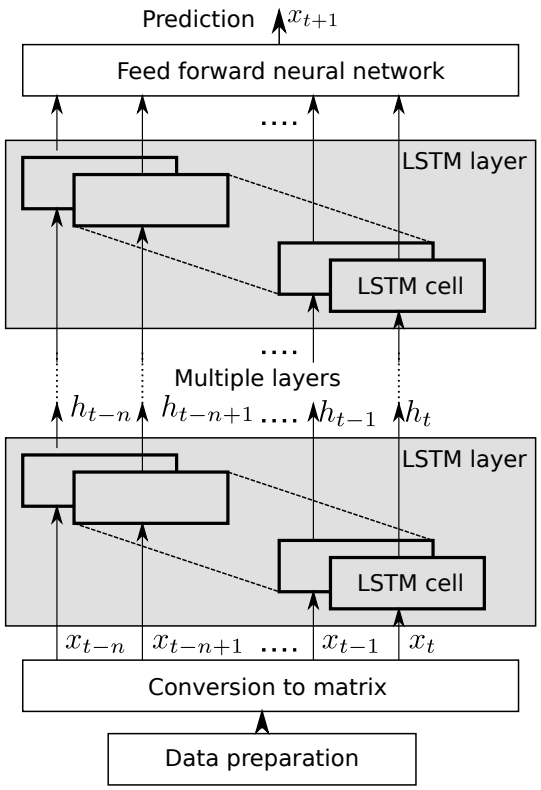


Fig. 4: Architecture of the LSTM ANN used for network throughput prediction.

Table 1: Network capacity traces used in the experiments.

Trace	Activity	Length [min]	Average [MB/s]	Var. range [MB/s]
S4G	Static 4G	254	5.2	2.6-9.6
M5G	Mobile 4G	290	2.7	0.3-6.1
S5G	Static 5G	260	8.3	2.7-25
M5G	Mobile 5G	459	3.5	0.4-11

4.3 Traffic characteristics

The traffic traces were collected in Ireland by a large telecommunication operator. They include static and mobility scenarios and total roughly 21 hours. We divided them into four sets: 5G, 4G, static and mobile, as listed in Table 1. A more detailed description of the traffic generation, capture methodology and properties of the traces can be found in [29] for 4G and in [30] for the 5G traffic.

Table 1 shows that the average rates of the 5G traces are about 50% higher than 4G traces for the static scenario. Furthermore, the 5G traces achieve nearly three times higher variation rates, reaching about 25 MB/s, compared to 4G traces which obtain 9.6 MB/s. In the mobile scenario, the differences between 5G and 4G are lower – the average throughput of 5G traces is only 27% higher than the 4G traces. The authors of [29] explain these differences by the lack of 5G base stations across the driving routes where the traffic was captured. Despite this restriction, the upper limit for the variation range is still almost two times higher for 5G than 4G.

The average amount of traffic registered within a specified time is represented in our work as the average throughput r_i (1). As stated in section 3.1, the length of a video segment is L. Assuming that the algorithm downloads video segments at the rate $q_i \approx \tilde{q}_i$, the algorithm will measure the network throughput roughly every L seconds. Hence,

$$\Delta T \approx L = 4 \text{ s}.$$

4.4 Traffic prediction with an LSTM ANN

For the time series prediction, an LSTM ANN uses a sliding window containing n most recent observations, as defined in (5). From every set of the traces listed in Table 1, we extracted ten overlapping sub-traces, each 30-minutes length. The models were trained on the first 10 minutes of each sub-trace and tested on its remaining part. We concatenated the obtained predictions, what allowed us to input their values into a video player and have uninterrupted playback.

Regarding the one-step prediction, we used a many-to-one architecture, which means that the network observes mobile traffic for a fixed number of timeslots until t and then try to predict the traffic at the next time slot t+1. In the prediction process, firstly, we normalise the consecutive traffic flow data $\{r_t, r_{t-1}, \ldots, r_{t-n}\}$, i.e. remove trends and differentiate it obtaining $\{x_t, x_{t-1}, \ldots, x_{t-n}\}$, which is then delivered to the LSTM ANN. Our LSTM consists of an input, two hidden and output layers, each characterized by the weight matrices and vectors. The outputs of the top LSTM layer are fed to a conventional feed-forward ANN, which maps the intermediate LSTM outputs to a single value. In the last step, we denormalise the output values from the feed-forward ANN obtaining the prediction of network throughput $\hat{x}_{t+1}$ at $t+1$. This process and the structure of the LSTM ANN used for predicting traffic flow is presented in Fig. 4.

An LSTM ANN minimizes the NRMSE defined in (7) through training which iteratively changes the weights and biases of the network to decrease the differences between the model output and observed values. For this purpose, we chose *Backpropagation Through Time* algorithm.

4.5 Model validation

The prediction is more accurate for traces generated by static terminals compared with the mobile ones, see Fig. 5. Furthermore, the traffic from 4G systems obtained a lower NRMSE than the 5G traces. These differences may result from the positive correlation between the prediction errors and variance of the traces. The traces generated by mobile scenarios have higher variation than the static ones, see Table 1. Additionally, the 5G traces, due to higher network bandwidth, achieved higher peak values. The variation of the results is roughly on the same level, except for the 5G mobile scenario where the NRMSE have a higher range. Compared to a simple moving average, the LSTM predations achieved lower NRMSE. The difference is particularly notable for the mobile and 5G scenarios.

In Fig. 6, we presented a fragment of the M5G trace to visualise the traffic characteristics and the prediction accuracy. The trace has an irregular structure without clearly identifiable patterns. The traffic frequently and abruptly changes its volume, producing oscillations between 2.5 and 5 MB/s. The prediction generally follows the traffic pattern relatively well; however, sometimes, it overestimates the scope of the oscillations, especially in the fragments where the traffic fluctuations have a higher frequency.

The prediction errors for the exemplary M5G trace have a normal distribution with a zero mean, as shown in Fig. 7a. The errors are between -1.0 MB/s and 1.0 MB/s, and most do not exceed 0.5 MB/s. Compared with the case of the simple moving average shown in Fig. 7b, its histogram is broader, and, consequently, the errors have higher values than for the LSTM prediction.

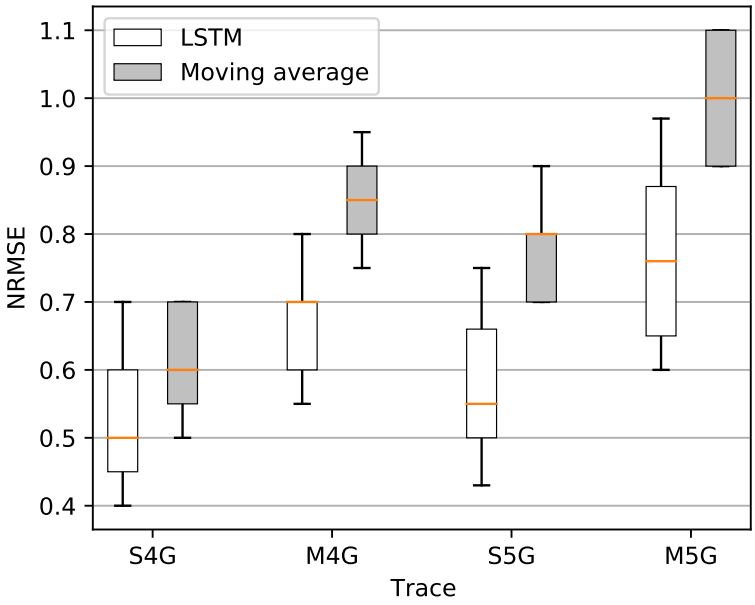


Fig. 5: NRMSE of the analysed prediction models for different traces. Every box on the plot represents predictions made on ten overlapping sub-traces, each 30-minutes length, which were extracted from traces listed in Table 1.

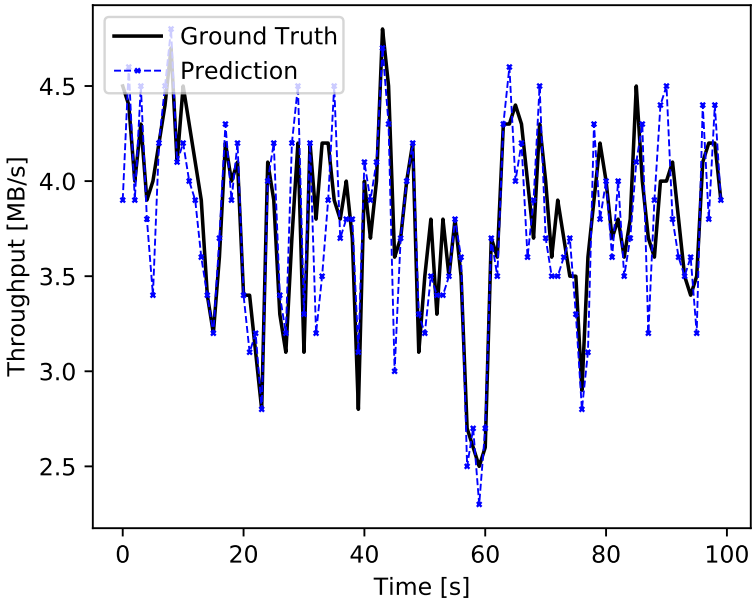
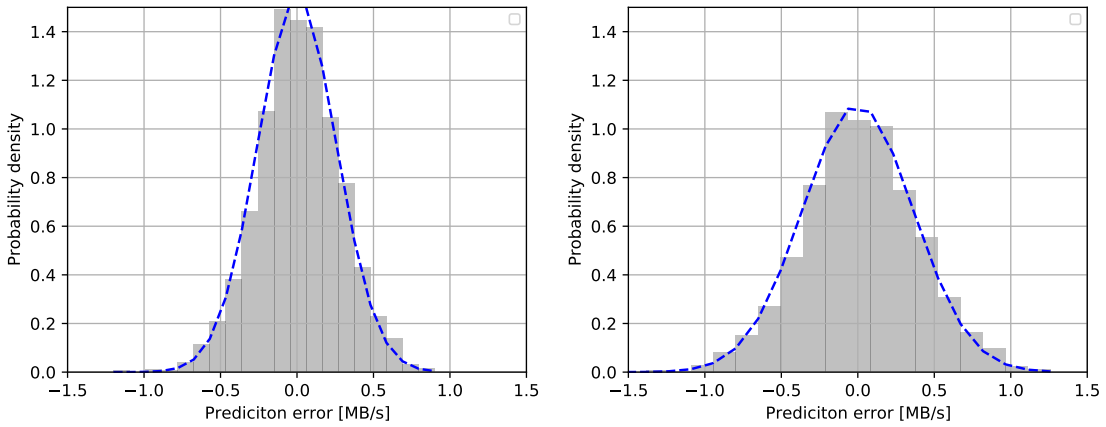


Fig. 6: One-step prediction for the M5G trace. The prediction tends to overestimate the traffic oscillations, but the LSTM ANN achieves better results than a naive predictor based on a simple moving average.



(a) (b)

Fig. 7: Distribution of prediction errors generated by a a) LSTM and b) simple moving average for the one-step prediction for M5G trace. The narrower prediction error interval for the LSTM ANN indicate fewer errors and better prediction accuracy.

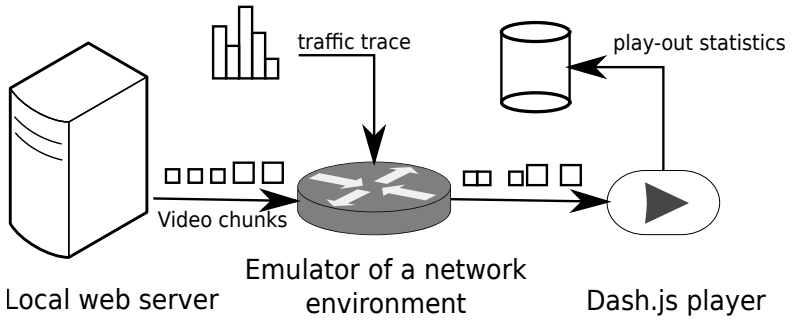


Fig. 8: Experiment set-up includes a web server, *netem* module and a *dash.js* video player. The connection between the emulated network and the server has fixed parameters. The *netem* shapes the traffic between the emulated network and the DASH player providing variable transmission time of video segments.

4.6 Playback algorithms supported by prediction

To assess the influence of the prediction on the quality of the video, we prepared a local media streaming system consisting of three Linux machines, as illustrated in Fig. 8. The first machine ran a web server, the second one ran a traffic control facility *netem*, and the last one ran a dash.js player [24] in a Google Chrome browser. The captured traffic trace, which we examined in section 4.3, was used as a template for the capacity shaper implemented in the *netem* module. In addition to the capacity throttling, the *netem* module also adds a uniformly distributed delay from the interval [5 ms, 145 ms] and uniformly distributed packet loss from the interval [0.05%, 0.15%] to emulate the instability of the connections. We placed six video files acquired from [31] at the web server and encoded them with 18 bit rates ranging from 0.1 to 20 MB/s. The server streamed the video to the dash.js media player. The player had an open-source code; hence it was possible to integrate a variety of adaptation logic, making it attractive for performance comparison of different DASH algorithms and their parameters.

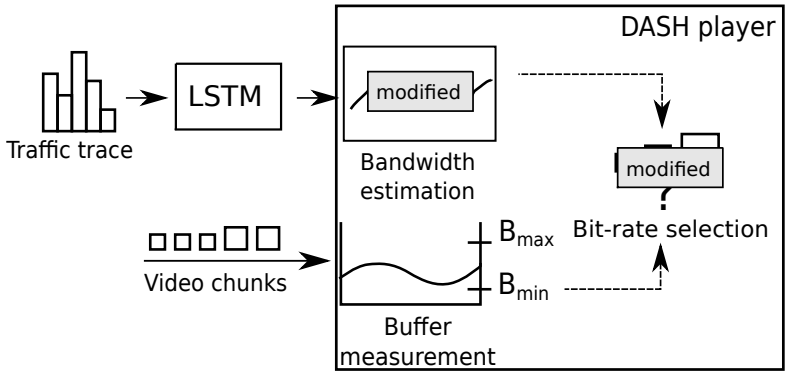


Fig. 9: The architecture of dash.js reference player. The modified estimation module acquires information about network throughput from the LSTM ANN. The MSS, ExoPlayer and Tian algorithms replaced the player's default playback logic.

The predicted throughput may integrate into the adaptation logic in two different ways. Firstly, it is possible to incorporate a prediction algorithm directly into a DASH player and replace its default throughput estimation algorithm. This approach requires translating the model into a native DASH player's code or wrapping the model into a package, which the player calls as an external dependency. In summary, it requires deep integration into the player's implementation, what is challenging. In the second case, the prediction overwrites the measured throughput samples. The player acquires the information about throughput not directly from a network but an output of the prediction model. This solution is easier to implement as it requires only minimal amendment of the player's code; hence, we replaced the default throughput estimation with the predicted values, see Fig. 9. As we mentioned in section 3.2, we also overwrote the default playback algorithm implemented in the plug-in using the simplified versions of MSS, ExoPlayer and Tian algorithms.

Fig. 10 shows a fragment of the performance of the MSS algorithm in its original version and the modified version with support of the prediction. Both versions operate on the M5G trace from the set presented in Table 1. Compared with the original version, the prediction supported algorithm switches its bit rates less often and more quickly recovers from bandwidth drops; nevertheless, its average bit rate is similar. Consequently, the less frequent switching of bit rates in the prediction-based version does not come at the cost of a lower bit rate.

Fig 11 compares nQoE (4) between the LSTM prediction and simple moving average. The nQoE is higher from about 5% to over 20%, depending on a DASH player and streaming scenario. In the case of MSS and ExoPlayer, the average improvement is mostly between 10% and 15%. For static traces generated by the 4G system, this value is closer to 10%; for mobile scenarios, the improvement is about 17%. The mobile use case obtain on average better results, but the higher nQoE variation indicates that the refinement is not uniform but depends on network conditions. Throughput in mobile scenarios, especially involving 5G network, have higher variability (see Table 1); therefore, some parts of the traces may be harder to predict. The poorer performance of the default prediction based on moving average can also contribute to better relative improvement. For Tian, the resulting pattern is similar, but the gain is a few percent lower than in the other two algorithms. Probably the more complex algorithm based on a PID controller and less reliance on direct throughput estimation reduces the benefits from the prediction.

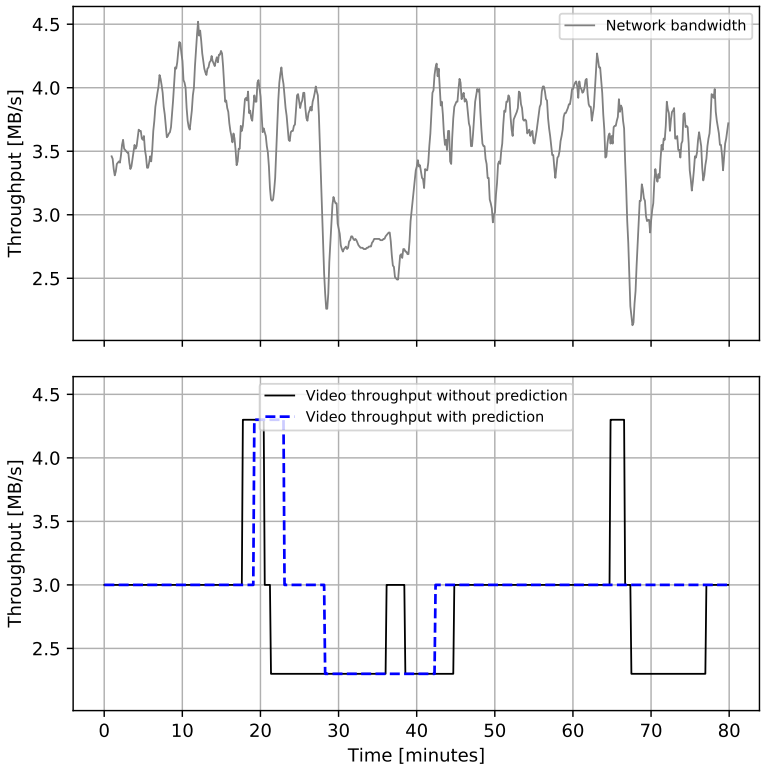


Fig. 10: Performance of MSS algorithm without and with support of traffic prediction. Throughput prediction eliminates some of the video bit rate drops observed at the beginning, end and about 20th minute of the trace. However, as the prediction sometimes overestimates the throughput, the bit rated unnecessary switches to a higher level at about 20-the minute of the trace.

4.7 Discussion of the results

Our analysis shows that an LSTM ANN obtains better prediction accuracy compared with a naive predictor based on a moving average. The prediction achieved the best results for traffic registered by fixed terminals. The traffic acquired from mobile equipment, especially that operating in a 5G network, had a more irregular structure and was harder to predict for both the LSTM ANN and a moving average. However, for mobile traces, the accuracy of the prediction made by the moving average deteriorates even faster than the accuracy of an LTSM ANN. Although we used an NRMSE (7), comparison with other research is not easy, as their NRMSE values differ significantly across different works. For instance, in [9], the NRMSE ranges between about 0.1 and 5.5 depending on an LSTM ANN and a data set used for the prediction, whereas in [12], the NRMSE is below 0.1 for all experiments. It is also unclear how the authors normalised their errors as they did not specify explicit mathematical formulas in their works. In other mentioned works in section 2, e.g. [6][7][8][10], the authors provided non-normalised and non-comparative mean square error measures of prediction errors. Sometimes this measure is normalised like in [32] (normalised mean absolute error) or [13] (relative MSE) but is non-comparative with NRMSE without additional data.

Exemplary DASH algorithms supported by throughput prediction achieved better performance than their versions with default throughput estimators. Most gained the MSS and ExoPlayer playback strategies, while the results for Tian were lower. An explanation is the degree of complexity

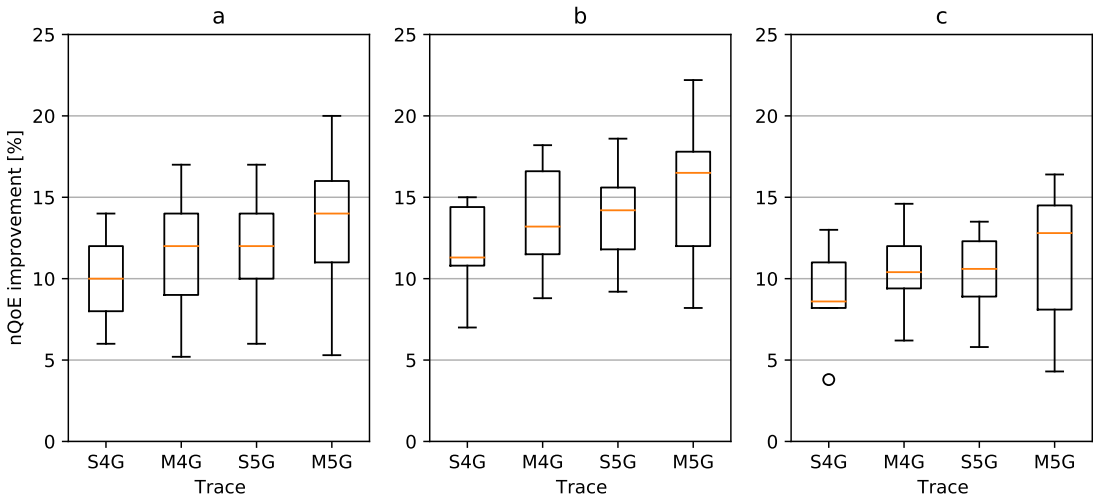


Fig. 11: The relative improvement of the nQoE metric across different HAS algorithms and network traces with respect to the no-prediction: a) MSS, b) ExoPlayer, and c) Tian. The nQoE metrics are normalised to the no-prediction case set to 0% improvement. The scenarios with traces from the mobile environment obtain, on average, better results, but their variation is also higher.

of the examined approaches. The first two algorithms rely heavily on rate estimation and use for this purpose moving averages, while the buffer measurements play a supportive role. In the case of Tian, the algorithm uses primarily the buffered video time as a feedback signal, while the expected throughput is estimated using support vector regression. This shift of focus results in a lower impact of prediction on the algorithm performance.

To our best knowledge, there is only a single work with which we can directly compare the results because every research uses its own set of measures. In our work, the quality improvement is between 5% and 25%, while for the perfect throughput prediction Raca et al [33] reached 55% using similar nQoE (4) measure. The authors built a perfect estimator by looking explicitly into the future values of time series and applied it to three algorithms: ExoPlayer, Elastic [34] and Arbiter+ [35]. Similarly to our case, the simpler algorithm ExoPlayer profited the most from the prediction. The prediction gave a less impressive nQoE increase for the two other, more complex algorithms.

5 Conclusions

In this work, we improved the performance of adaptive streaming algorithms employing an LSTM ANN which predicted network throughput. As a result, end-users experience higher video quality without longer buffering times or a higher number of bit rate switches. The quality improvement is especially significant in 5G networks; however, the more advanced the playback algorithm is, the less it gains from throughput prediction. The best results achieved relatively simple strategies based on throughput and buffer measurement. The improvement is not uniform across the usage scenario and depends on network conditions. The prediction gain was smaller for a more complex playback algorithm supported by the proportional-integral-derivative controller.

Our approach does not require modifications of the network infrastructure or the TCP stack. It can be integrated with the existing playback algorithms and video players because the throughput estimation is separated from other modules of a video player. The proposed technique can be used by developers of the video players, which are free to employ their adaptation strategies to enhance

video playback. The prediction model relies only on past throughput measurement and works without additional network information, which can be hard to acquire. Therefore, the prediction accuracy and the quality improvement are less impressive than in some cited works, which support their prediction with additional data.

We identified several potential paths for further investigation. Firstly, the traffic models can be improved as some authors report that the best results are usually obtained by hybrid techniques. It will be interesting to investigate how to leverage the additional information to increase the prediction accuracy as more elaborate approaches may better adjust to network data and provide better input for adaptive algorithms. Secondly, there is room to extend our prediction beyond the current single video chunk horizon. Thirdly, one can investigate the optimal length of the training, test sets and a time horizon that provides the best prediction. Finally, we would like to move from the offline analysis to an online prediction made by a mobile device, which measures the throughput in real-time and provides information about battery, memory and processor usage.

References

1. Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K. Sinha. Can Accurate Predictions Improve Video Streaming in Cellular Networks? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 57–62. ACM, 2015.
2. Christos Katris and Sophia Daskalaki. Comparing forecasting approaches for Internet traffic. *Expert Systems with Applications*, 21(42):8172–8183, 2015.
3. Yan Liu and Jack YB Lee. An Empirical Study of Throughput Prediction in Mobile Data Networks. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.
4. Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3):e0194889, 2018. Publisher: Public Library of Science San Francisco, CA USA.
5. Tiago Prado Oliveira, Jamil Salem Barbar, and Alexsandro Santos Soares. Computer network traffic prediction: a comparison between traditional and deep learning neural networks. *International Journal of Big Data Intelligence*, 3(1):28–37, 2016. Publisher: Inderscience Publishers (IEL).
6. Anestis Dalgkitsis, Malamati Louta, and George T Karetsos. Traffic forecasting in cellular networks using the lstm rnn. In *Proceedings of the 22nd Pan-Hellenic Conference on Informatics*, pages 28–33, 2018.
7. Qinzheng Zhuo, Qianmu Li, Han Yan, and Yong Qi. Long short-term memory neural network for network traffic prediction. In *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 1–6, Nanjing, November 2017. IEEE.
8. Fawaz Waselallah Alsaade and Mosleh Hmoud Al-Adhaileh. Cellular Traffic Prediction Based on an Intelligent Model. *Mobile Information Systems*, 2021:1–15, July 2021.
9. Qing He, Arash Moayyedi, Gyorgy Dan, Georgios P. Koudouridis, and Per Tengkvist. A Meta-Learning Scheme for Adaptive Short-Term Network Traffic Prediction. *IEEE Journal on Selected Areas in Communications*, 38(10):2271–2283, October 2020.
10. Rui Fu, Zuo Zhang, and Li Li. Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328, Wuhan, Hubei Province, China, November 2016. IEEE.
11. Shulin Cao and Wei Liu. LSTM Network Based Traffic Flow Prediction for Cellular Networks. In Houbing Song and Dingde Jiang, editors, *Simulation Tools and Techniques*, volume 295, pages 643–653. Springer International Publishing, Cham, 2019. Series Title: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.
12. Hoang Duy Trinh, Lorenza Giupponi, and Paolo Dini. Mobile traffic prediction from raw data using LSTM networks. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1827–1832. IEEE, 2018.
13. Maxime Labonne, Jorge López, Claude Poletti, and Jean-Baptiste Munier. Short-Term Flow-Based Bandwidth Forecasting using Machine Learning. *arXiv:2011.14421 [cs, eess]*, December 2020. arXiv: 2011.14421.
14. Amin Azari, Panagiotis Papapetrou, Stojan Denic, and Gunnar Peters. Cellular traffic prediction and classification: A comparative evaluation of LSTM and ARIMA. In *International Conference on Discovery Science*, pages 129–144. Springer, 2019.
15. Chaoqun Yue, Ruofan Jin, Kyoungwon Suh, Yanyuan Qin, Bing Wang, and Wei Wei. LinkForecast: Cellular Link Bandwidth Prediction in LTE Networks. *IEEE Transactions on Mobile Computing*, 17(7):1582–1594, July 2018.

16. Qing He, Georgios P Koudouridis, and György Dán. A comparison of machine and statistical time series learning for encrypted traffic prediction. In *2020 International Conference on Computing, Networking and Communications (ICNC)*, pages 714–718. IEEE, 2020.
17. Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen, and David Oran. Probe and adapt: Rate adaptation for http video streaming at scale. *Selected Areas in Communications, IEEE Journal on*, 32(4):719–733, 2014.
18. Abdelhak Bentaleb, Christian Timmerer, Ali C. Begen, and Roger Zimmermann. Performance Analysis of ACTE: A Bandwidth Prediction Method for Low-latency Chunked Streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 16(2s):1–24, July 2020.
19. Darijo Raca, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, and Vijay Gopalakrishnan. On Leveraging Machine and Deep Learning for Throughput Prediction in Cellular Networks: Design, Performance, and Challenges. *IEEE Communications Magazine*, 58(3):11–17, March 2020.
20. Arkadiusz Biernacki. Traffic prediction methods for quality improvement of adaptive video. *Multimedia Systems*, 24(5):531–547, 2018.
21. A. Zambelli. IIS smooth streaming technical overview. *Microsoft Corporation*, 2009.
22. ExoPlayer.
23. Guibin Tian and Yong Liu. Towards Agile and Smooth Video Adaptation in HTTP Adaptive Streaming. *IEEE/ACM Transactions on Networking*, 24(4):2386–2399, 2016.
24. DASH Industry Forum (DASH-IF). JavaScript Reference Client. Retrieved from https://reference.dashif.org/dash.js/, 2021.
25. Jeroen Famaey, Steven Latré, Niels Bouten, Wim Van de Meerssche, Bart De Vleeschauwer, Werner Van Leekwijck, and Filip De Turck. On the merits of SVC-based HTTP adaptive streaming. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 419–426. IEEE, 2013.
26. Jonathan Kua, Grenville Armitage, and Philip Branch. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP. *IEEE Communications Surveys & Tutorials*, 19(3):1842–1866, 2017.
27. Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP. *IEEE Communications Surveys & Tutorials*, 21(1):562–585, 2019.
28. Román Belda, Ismael de Fez, Pau Arce, and Juan Carlos Guerri. Look ahead to improve QoE in DASH streaming. *Multimedia Tools and Applications*, 79(33-34):25143–25170, September 2020.
29. Darijo Raca, Jason J. Quinlan, Ahmed H. Zahran, and Cormac J. Sreenan. Beyond throughput: a 4G LTE dataset with channel and context metrics. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 460–465, Amsterdam Netherlands, June 2018. ACM.
30. Darijo Raca, Dylan Leahy, Cormac J. Sreenan, and Jason J. Quinlan. Beyond throughput, the next generation: a 5G dataset with channel and context metrics. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 303–308, Istanbul Turkey, May 2020. ACM.
31. Anatoliy Zabrovskiy, Christian Feldmann, and Christian Timmerer. Multi-codec DASH dataset. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 438–443, Amsterdam Netherlands, June 2018. ACM.
32. Meejoung Kim. Network traffic prediction based on INGARCH model. *Wireless Networks*, 26(8):6189–6202, November 2020.
33. Darijo Raca, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, Vijay Gopalakrishnan, Balagangadhar Bathula, and Matteo Varvello. Incorporating Prediction into Adaptive Streaming Algorithms: A QoE Perspective. In *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 49–54, Amsterdam Netherlands, June 2018. ACM.
34. Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. ELASTIC: A client-side controller for dynamic adaptive streaming over HTTP (DASH). In *2013 20th International Packet Video Workshop*, pages 1–8. IEEE, 2013.
35. Ahmed Hamdy Zahran, Darijo Raca, and Cormac J Sreenan. Arbiter+: Adaptive rate-based intelligent http streaming algorithm for mobile networks. *IEEE Transactions on Mobile Computing*, 17(12):2716–2728, 2018. Publisher: IEEE.