Realtime Mobile Bandwidth Prediction using LSTM Neural Network

Lifan Mei[1] Runchen Hu[1] Houwei Cao[2] Yong Liu[1] Zifa Han[3] Feng Li[3] Jin Li[3]

[1] ECE, New York University, NY 11201, USA
{lifan, rh2619, yongliu}@nyu.edu
[2] CS, New York Institute of Technology, NY 10023, USA
{hcao02}@nyit.edu
[3] Huawei Technologies, Nanjing, China
{hanzifa, frank.lifeng, mark.lijin}@huawei.com

Abstract. With the popularity of mobile access Internet and the higher bandwidth demand of mobile applications, user Quality of Experience (QoE) is particularly important. For bandwidth and delay sensitive applications, such as Video on Demand (VoD), Realtime Video Call, Games, etc., if the future bandwidth can be estimated in advance, it will greatly improve the user QoE. In this paper, we study realtime mobile bandwidth prediction in various mobile networking scenarios, such as subway and bus rides along different routes. The main method used is Long Short Term Memory (LSTM) recurrent neural network. In specific scenarios, LSTM achieves significant accuracy improvements over the state-of-the-art prediction algorithms, such as Recursive Least Squares (RLS). We further analyze the bandwidth patterns in different mobility scenarios using Multi-Scale Entropy (MSE) and discuss its connections to the achieved accuracy.

Keywords: Bandwidth Prediction · Long Short Term Memory · Multi-Scale Entropy · Bandwidth Measurement.

1 Introduction

We have witnessed the tremendous growth of mobile traffic in the recent years. Users are increasingly spending more time on mobile apps and consuming more content on their mobile devices. The growth trend is expected to accelerate in the foreseeable future with the introduction of 5G wireless access and new media-rich applications, such as Virtual Reality and Augmented Reality. However, one main challenge for mobile app developers and content providers is the high volatility of mobile wireless connections. The physical channel quality of a mobile user is constantly affected by interference generated by other users, his/her own mobility, and signal blockages from static and dynamic blockers [10, 9]. The bandwidth available for a mobile session is ultimately determined by the adaptations cross the protocol stack, ranging from adaptive coding and modulation at PHY layer, cellular scheduling at data link layer, hand-overs between base stations, to TCP congestion control, etc. For many mobile apps involving user interactivity and/or multimedia content, e.g., gaming, conferencing and video streaming, it is critical to accurately estimate the available bandwidth in realtime to deliver a high quality of user Quality-of-Experience (QoE). In the example of video streaming,

many recent algorithms on Dynamic Adaptive Streaming over Http (DASH) optimize the video rate selection for incoming video chunks based on the predicted TCP throughput in a future time window of several seconds [12, 16, 6]. To cope with the unavoidable TCP throughput prediction errors, one has to be conservative in video rate selection and resort to long video buffering to absorb the mismatch between the predicted and actual TCP throughput. Both degrade user video streaming QoE. Interactive video conferencing has much tighter delay constraint than streaming. To avoid self-congestion, the available bandwidth on cellular link has to be accurately estimated in realtime, which is used to guide the realtime video coding and transmission strategies [11, 17]. Bandwidth overestimate will lead to long end-to-end video delay or freezing, bandwidth underestimate will lead to unnecessarily poor perceptual video quality. Again, accurate realtime bandwidth prediction is crucial for delivering good conferencing experience, especially in mobile networking scenarios.

In this paper, we study realtime mobile bandwidth prediction using Long Short Term Memory (LSTM) [1] recurrent neural network. Recent advances in Deep Learning have demonstrated that Recurrent Neural Networks (RNN) are powerful tools for sequence modeling and can learn temporal patterns in sequential data. RNNs have been widely used in Natural Language Processing (NLP), speech recognition and time series processing [18, 19]. There are rich structures in realtime mobile network bandwidth evolution, due to user mobility patterns, wireless signal propagation laws, physical blockage models, and the well-defined behaviors of network protocols. This presents abundant opportunities for developing LSTM-based realtime mobile bandwidth estimation. The main idea is to offline train LSTM RNN models that capture the temporal patterns in various mobile networking scenarios. The trained LSTM RNN models will be used online to predict in realtime the network bandwidth within a short future time window. Specifically, we investigate the following research questions:

1. *How much prediction accuracy improvement can LSTM deep learning models bring over the conventional statistical prediction models?*
2. *How predictable is realtime bandwidth at different prediction intervals under different mobility scenarios? Is the LSTM prediction accuracy dependent on specific mobility scenarios?*
3. *Should one train a separate LSTM model for each mobility scenario, or train a universal LSTM model that can be used in different scenarios?*

Towards answering these questions, we made the following contributions:

- We conducted a mobile bandwidth measurement campaign to collect consecutive bandwidth traces in New York City. Our traces cover different transportation methods along different routes at different time of day[4]
- We developed LSTM models for realtime one-second ahead and multi-second ahead bandwidth predictions. Through extensive experiments on our own

[4] The collected NYU Metropolitan Mobile Bandwidth Trace Dataset (NYU-METS), is publicly available at https://github.com/NYU-METS/Main

dataset and the HSDPA dataset [8], we demonstrated that LSTM significantly outperforms the existing realtime bandwidth prediction algorithms.
- We systematically evaluated the sensitivity of LSTM models to different mobility scenarios by comparing the performance of *per-scenario, cross-scenario* and *universal* predictions. Using Multi-Scale Entropy (MSE) analysis, we studied the connection between prediction accuracy and bandwidth regularity at different time scales. MSE also provides us with guidelines to explore cross-scenario bandwidth prediction.

The rest of the paper is organized as the following. The related work on realtime bandwidth prediction is reviewed in Section 2. We formally define the realtime bandwidth prediction problem and introduce our LSTM based prediction models in Section 3. The performance of LSTM models is evaluated by public dataset and our own dataset in Section 4. We conduct Multi-Scale Entropy analysis on our collected bandwidth traces and analyze the prediction accuracy in Section 5. The paper is concluded with future work in Section 6.

2 Related Work

Realtime bandwidth prediction has been a challenging problem for the networking community. Simple history-based TCP throughput estimation algorithm was proposed in [13]. Authors of [14] proposed to train a Support Vector Regress (SVR) model [15] to predict TCP throughput based on the measured packet loss rate, packet delay and the size of file to be transmitted. In the context of DASH video streaming, in [12], we adopted prediction algorithm in [13] to guide realtime chunk rate selection, and used a customized SVR model similar to [14] for DASH server selection. Authors of [21] and [16] used the Harmonic Mean of TCP throughput for downloading the previous five chunks as the TCP throughput prediction for downloading the next chunk. In [6], authors developed Hidden Markov Model (HMM) for bandwidth prediction. HMM model is parameterized by history bandwidth, and HMM state transition is used to infer future bandwidth. In the context of video conferencing, in [17], a cellular link is modeled as a single-server queue driven by a doubly-stochastic service process. Bandwidth available for a user is measured by the packet arrival dispersion at the receiver end, and future bandwidth prediction is generated by probabilistic inference based on the single-server queue model. In [11], we used an adaptive filter, Recursive Least Squared (RLS), to make realtime bandwidth prediction. We showed that RLS achieves good prediction accuracy on volatile cellular links. Based on the accurate bandwidth prediction, they proposed a new video conferencing system that can deliver higher video rate and lower video delay than Facetime in side-by-side comparisons.

All the previous predictors are based on the conventional statistical or machine learning models and generate predictions based on short bandwidth history. Different from the conventional models, LSTM deep learning models are more flexible and can be trained by large datasets to better capture the long-term and short-term temporal structures in bandwidth time series. A recent work on Deep Reinforcement Learning (DRL) based DASH [20] takes historical

bandwidth samples as part of the input state vector for DRL to directly generate video chunk rate selection. DRL based DASH achieves better performance and robustness than the traditional DASH. While DRL-DASH implicitly mines the temporal structure in bandwidth, there is no direct/explicit training and validation optimized for bandwidth prediction.

3 LSTM based Realtime Bandwidth Prediction

3.1 Realtime Bandwidth Prediction Problem

Let $x(t)$ be the bandwidth available for a user at time t. Given some bandwidth measurement frequency, one can obtain a discrete-time time series of $\{x(t), t = 1, 2, \cdots .\}$. The realtime bandwidth prediction problem at time t is to estimate the bandwidth available for a user at some future time instant $x(t+\tau)$ given all the observed bandwidth measurements so far, i.e.,

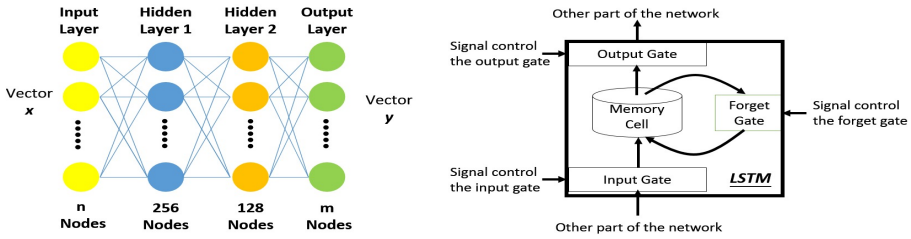
$$\hat{x}(t+\tau) = \mathbf{f}\left(\{x(k), k = 1, 2, \cdots, t\}\right). \tag{1}$$

There are many ways to build the estimation function $\mathbf{f}(\cdot)$, ranging from simple history-repeat, i.e., $\hat{x}(t+\tau) = x(t)$, Exponential Weighted Moving Average (EWMA), $\hat{x}(t+1) = (1-\alpha)\hat{x}(t) + \alpha x(t)$, Harmonic Mean, $\hat{x}(t+\tau) = h/\sum_{k=0}^{h-1} 1/x(t-k)$, etc., to more sophisticated signal processing approaches, such as Kalman filter [7] and Recursive Least Squares (RLS) [3]. In [11], we used RLS for realtime bandwidth prediction. By assuming $\hat{x}(t+1) = \sum_{k=0}^{h-1} \omega(k)x(t-k)$, RLS recursively finds the coefficients ω that minimizes a weighted linear least squares cost function. In the bandwidth prediction part of [11], it was shown that RLS achieves better accuracy than other averaging and signal processing algorithms, such as Least Mean Square and EWMA etc.

3.2 LSTM-based Prediction Model

While all those methods use history measurements to generate bandwidth prediction, they did not fully explore the temporal patterns in realtime bandwidth evolution for more accurate prediction. Meanwhile, LSTM network has recently emerged as a powerful tool for exploring temporal structures in sequential data. As illustrated in Figure 1a, a LSTM network consists of layers of LSTM units. As illustrated in Figure 1b, a common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "memorizing" values over arbitrary time intervals; hence the word "memory" in LSTM. Each of the three gates can be thought of as a "conventional" artificial neuron, as in a multi-layer (or feed-forward) neural network: they compute an activation (using an activation function) of a weighted sum. Intuitively, they can be considered as regulators of the flow of values going through the connections between the LSTM units; hence the denotation "gate". There are connections between these gates and the cell. Detailed LSTM reviews can be found in [1, 2].

The input to our LSTM bandwidth prediction network is the recent bandwidth measurements, i.e, $\mathbf{x} = [x(t), x(t-1), \cdots, x(t-n+1)] \in R^n$, the output is the predicted bandwidth in a future time window $\mathbf{y} = [\hat{x}(t+1), x(t+2), \cdots, x(t+$



(a) LSTM Network Architecture (b) Internal Structure of LSTM Unit

Fig. 1: LSTM Network for Realtime Bandwidth Prediction

$m)] \in R^m$. Note that since LSTM network adaptively keeps "memory", the bandwidth prediction for time window $(t, t+m]$ is not only directly determined by the recent bandwidth history in $(t-n, t]$, but also indirectly affected by bandwidth history before $t-n$ through the memory cells. This gives LSTM more flexibility in capturing long-term bandwidth evolution trends than the traditional signal processing and averaging approaches working on a moving history window. Following the architecture in Figure 1a, we build a LSTM network with one input Layer, one output layer and two hidden layers, each with 256 and 128 LSTM units respectively[5]. Given the LSTM architecture, the mapping from input $\mathbf{x}$ to output $\mathbf{y}$ is parameterized by all the parameters in the LSTM network, denoted as θ, which are obtained by minimizing the loss function in training.

Since we study realtime bandwidth prediction for a range of mobile networking scenarios, one option is to train a separate LSTM network for each scenario, that is using bandwidth data collected from scenario i to train a LSTM network with parameters $\theta^{(i)}$, and then use it to predict bandwidth for scenario i, i.e.,

$$\textbf{per-scenario:} \qquad \hat{\mathbf{y}}^{(\mathbf{i})} = \mathbf{LSTM}\left(\mathbf{x}^{(\mathbf{i})}, \theta^{(\mathbf{i})}\right), \quad \forall i. \tag{2}$$

Another option is to train one universal LSTM network with parameters $\theta^{(0)}$ using all data collected from all scenarios, and hope the trained universal LSTM model can be used to predict bandwidth in all scenarios, i.e.,

$$\textbf{universal:} \qquad \hat{\mathbf{y}}^{(\mathbf{i})} = \mathbf{LSTM}\left(\mathbf{x}^{(\mathbf{i})}, \theta^{(\mathbf{0})}\right), \quad \forall i. \tag{3}$$

The third option is to train a LSTM network using data from scenario i, then use it to predict bandwidth in scenario j.

$$\textbf{cross-scenario:} \qquad \hat{\mathbf{y}}^{(\mathbf{j})} = \mathbf{LSTM}\left(\mathbf{x}^{(\mathbf{j})}, \theta^{(\mathbf{i})}\right), \quad i \neq j. \tag{4}$$

[5] We also tried a LSTM network with 256 and 256 nodes, and a LSTM network with 128 and 128 nodes. The performance difference is not significant. The results presented in this paper is based on the 256 + 128 LSTM network.

To generate training samples, we use a sliding-window based approach. For example, to predict the bandwidth in the next second ($m = 1$) based on the bandwidth measurements in the previous five seconds ($n = 5$), in the training, we use every consecutive six bandwidth measurements as one training data point. The first five seconds bandwidth form the input vector, and the sixth second bandwidth is the output label. Likewise, for the general multiple seconds prediction, i.e., predicting the future bandwidth for the next m seconds based on the previous n seconds bandwidth, we use every consecutive $n + m$ bandwidth measurements as one data point. The first n measurements form the input vector, and the last m measurements form an output label vector.

4 Data Collection and Performance Evaluation

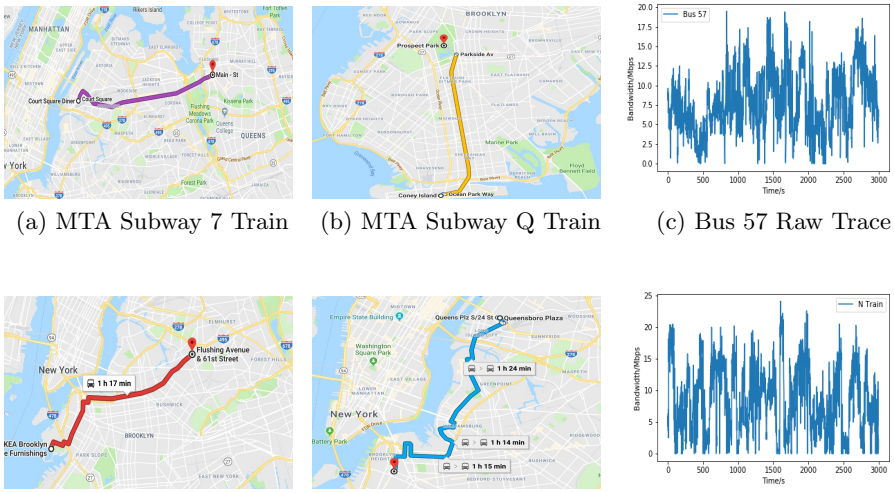
4.1 Datasets

It is critical to train and test LSTM models using large representative bandwidth datasets. We first used the HSDPA [8] dataset from the University of Oslo. It consists of cellular bandwidth traces collected on different transportation methods, including Train, Tram, Ferry, Car, Bus and Metro. For each trace, it recorded the bandwidth and location every 1,000 milliseconds, and the duration for each trace ranges from 500 to 1,000 seconds. However, we later found that the bandwidth traces are too short for MSE analysis. We also collected long bandwidth traces in New York City MTA bus and subway by ourselves. Figure 2 shows some sample routes for our bandwidth collection, including Subway 7 Train, Subway Q Train, Bus B57 and B62. On each route, we conducted multiple experiments at different time of day. For each experiment, we connect a LTE mobile phone with unlimited data plan to a remote server in our lab. We run $iPerf$ and record TCP throughput every $1,000$ millisecond. All the bandwidth samples are logged on the server side. The duration of each trace ranges from $10,000$ to $20,000$ seconds. It took us four months to complete the first batch of data. We are continuing this measurement campaign and keep adding new traces to our NYU-METS Dataset for future research.

4.2 Next-Second Prediction

For the next-second prediction, the dimension of LSTM output is $m = 1$, and we pick LSTM input dimension of $n = 5$ for evaluation. Figure 3 visually compares the predicted values from Harmonic Mean, RLS and LSTM with the ground truth for a trace collected on NYC Subway 7 Train. For LSTM training, we use Adam optimizer [22] with default parameters (including learning rate, beta, etc) in training. 80% of the trace is used for training, the rest 20% is used for testing. We manually adjust dropout and epoch based on the performance of model.

We use the *Root Mean Square Error (RMSE)* and *Mean Absolute Error (MAE)* between the predicted bandwidth and the ground truth as the main accuracy measures. The complete prediction result of the three algorithms on our NYU-METS Dataset is reported in Table 1. (LSTM runs in the *per-scenario* mode). The unit is *Mbps*. LSTM has the lowest RMSE and MAE cross all mobility scenarios. The average accuracy improvement of LSTM over RLS and



(a) MTA Subway 7 Train (b) MTA Subway Q Train (c) Bus 57 Raw Trace

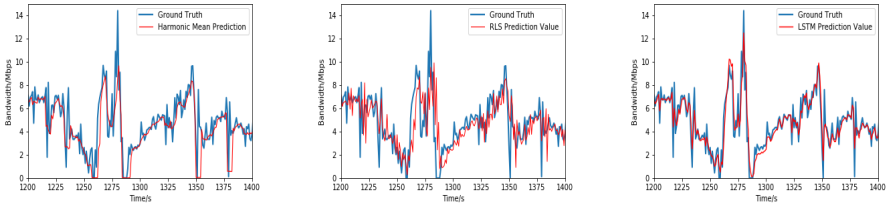
(d) MTA Bus 57 (e) MTA Bus 62 (f) Q Train Raw Trace

Fig. 2: New York City Self-measured Bandwidth

Table 1: Evaluation Results on NYU-METS Traces

	7A Train	7B Train	Bus 57	Bus 62	N Train
Testset Average	6.39	4.76	10.04	2.55	8.98
RLS RMSE	2.57	2.19	2.59	0.87	3.04
RLS MAE	1.69	1.49	1.72	0.66	2.11
Harmonic RMSE	2.98	2.60	2.79	0.94	3.36
Harmonic MAE	1.86	1.68	1.78	0.70	2.26
LSTM RMSE	**2.26**	**2.05**	**2.32**	**0.72**	**2.81**
LSTM MAE	**1.49**	**1.41**	**1.54**	**0.55**	**1.90**
RLS RMSE Error Ratio	40.3%	46.0%	25.8%	34.2%	33.8%
RLS MAE Error Ratio	26.5%	31.3%	17.1%	26.1%	23.5%
HAR RMSE Error Ratio	46.6%	54.6%	27.8%	37.0%	37.4%
HAR MAE Error Ratio	29.1%	35.4%	17.7%	27.4%	25.2%
LSTM RMSE Error Ratio	35.3%	43.1%	23.1%	28.2%	31.3%
LSTM MAE Error Ratio	23.3%	29.6%	15.3%	21.4%	21.2%
Relative RMSE Improvement over RLS	**14.0%**	**6.7%**	**11.8%**	**21.2%**	**8.2%**
Relative MAE Improvement over RLS	**13.6%**	**5.9%**	**11.9%**	**21.6%**	**11.0%**
Relative RMSE Improvement over Harmonic	**31.8%**	**26.7%**	**20.4%**	**31.1%**	**19.5%**
Relative MAE Improvement over Harmonic	**24.9%**	**19.7%**	**15.8%**	**27.7%**	**18.9%**

Harmonic Mean in RMSE are 12.4% and 25.9% respectively, for MAE, these are 12.8% and 21.4% respectively. Since Harmonic Mean performs much worse than the other two, in the following, we only compare LSTM with RLS.



(a) Harmonic Mean (b) RLS (c) LSTM

Fig. 3: Harmonic Mean, RLS and LSTM Predictions on Subway 7 Train

Table 2 compares the accuracy of per-scenario LSTM with RLS on the HSDPA dataset. The unit for the numbers is *kbps*. LSTM still outperforms RLS in all mobility scenarios. The Relative Improvement of LSTM over RLS are around 14.1% and 13.9% for RMSE and MAE respectively. For HSDPA dataset, we also trained a *universal* LSTM model by using all traces from different transportation scenarios, including Bus, Tram, Train, Metro and Car, then test its accuracy on individual transportation scenarios. However, it is performance is inferior to the corresponding per-scenario models. For some scenarios, its performance is even worse than RLS. Due to the space limit, we don't report the detailed statistics here. We defer the discussion on *cross-scenario* prediction to the next section, and defer universal prediction to future investigation.

Table 2: HSDPA Traces Evaluation Result of LSTM and RLS

	Ferry	FerryB	Tram	TramB	Metro	MetroB
Testset Average	248.4	217.6	118.8	133.4	96.0	119.7
RLS RMSE	71.3	88.9	35.3	35.6	34.2	35.5
RLS MAE	53.1	58.5	25.5	26.6	25.8	26.9
LSTM RMSE	**60.8**	**80.4**	**31.5**	**30.2**	**29.2**	**32.5**
LSTM MAE	**45.6**	**50.1**	**23.3**	**22.3**	**23.2**	**24.3**
RLS RMSE Error Ratio	28.7%	40.9%	29.8%	26.7%	35.7%	29.7%
RLS MAE Error Ratio	21.4%	19.7%	21.5%	19.9%	26.7%	22.5%
LSTM RMSE Error Ratio	24.5%	37.0%	26.6%	22.6%	30.4%	27.1%
LSTM MAE Error Ratio	18.4%	16.8%	19.6%	16.7%	24.3%	20.3%
Relative RMSE Improvement	**17.3%**	**10.6%**	**12.2%**	**17.8%**	**17.4%**	**9.3%**
Relative MAE Improvement	**16.5%**	**16.9%**	**9.5%**	**19.3%**	**11.0%**	**10.6%**

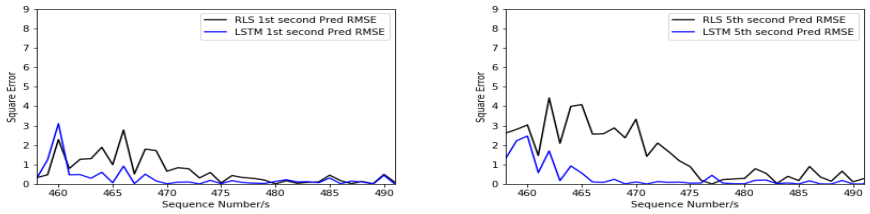
4.3 Multi-Second Prediction

We now study the prediction accuracy for longer time intervals. For LSTM model, we fix the input vector dimension to be $n = 10$, and vary the output vector dimension m from 2 to 5. In other words, LSTM network takes as input the bandwidth vector in the previous ten seconds to predict bandwidth for up to five seconds ahead. For each combination of n and m, we train a different LSTM model, denoted as $LSTM(n, m)$. Note that, at time t, a $LSTM(n, m)$ model can generate bandwidth predictions for $t + i$, $1 \leq i \leq m$. To make RLS

generate prediction i seconds ahead, we simply update RLS parameters by using bandwidth of i seconds ahead, instead of the next second, as the targeted output.

Table 3: Prediction RMSE on RLS and LSTM

	1st sec	2nd sec	3rd sec	4th sec	5th sec
RLS	2.57	2.88	3.16	3.53	3.76
$LSTM(10,1)$	**2.26**	-	-	-	-
$LSTM(10,2)$	2.27	**2.66**	-	-	-
$LSTM(10,3)$	2.29	2.68	**2.96**	-	-
$LSTM(10,4)$	2.33	2.69	2.97	**3.21**	-
$LSTM(10,5)$	2.40	2.71	2.98	3.22	**3.40**
Improvement over RLS	**13.7%**	**8.2%**	**6.8%**	**9.9%**	**10.6%**



(a) One Second Prediction (b) Five Second Prediction

Fig. 4: RLS vs LSTM Multi-Second Prediction

Table 3 compares the prediction accuracy of different LSTM models and RLS on the NYC Subway 7 Train trace. The RMSE value unit is $Mbps$. Not coincidentally, all LSTM models outperform RLS at all prediction intervals. In the representative results, the best prediction accuracy for interval i is achieved by $LSTM(10,i)$, marked in bold fonts. Theoretically, $LSTM(n,m)$ model is trained to minimize the prediction errors for all intervals from 1 to m. Consequently, the prediction error at interval $m_1 < m$ will be larger than those of $LSTM(n,m_2)$ models ($m_1 \leq m_2 < m$). Figure 4a and Figure 4b illustrate sample prediction error evolution of RLS and LSTM for one second and five second intervals. Y-axis is the square error between prediction value and ground truth. It is visually clear that LSTM RMSE is lower than RLS most of the time. The accuracy improvement of LSTM is more prominant for the five second prediction interval. Figure 5 compares the average RMSE for all LSTM models with RLS at different prediction intervals. Both RMSEs increase as the prediction interval increases. The slope for LSTM increase is 0.270, while that for RLS is 0.302. This suggests that not only LSTM is more accurate than RLS at individual prediction intervals, LSTM's accuracy decays slower than RLS as the interval increases.

4.4 Computation Overhead

To validate the feasibility of offline training and online prediction, we report the computation overhead of our LSTM models. Our CPU Configuration is: 4th

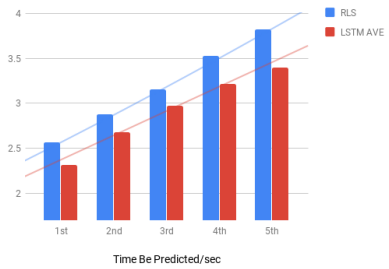


Fig. 5: Impact of Prediction Interval on LSTM and RLS

Gen Intel Core i5-4210U (1.70GHz 1600MHz 3MB). Neural Network Structure: Hidden Layer 1 & 2 have 256 and 128 nodes respectively. The training and running overhead detail is presented in Table 4. Even though the offline training time is long, once the training is done, the trained model can be used for realtime prediction. As shown in Table 4b, the online prediction consumption is so small. It takes less than six seconds to predict 12,500 five seconds bandwidth vector in the $LSTM(10, 5)$ model. Once the model is trained offline, it can be used to generate realtime prediction on any reasonably configured mobile phone.

Table 4: Computation Consumption

(a) Offline Training

Trainsize	Batchsize = 2	Batchsize = 4
13,000	120s/epoch	62s/epoch
10,000	95s/epoch	50s/epoch
5,000	49s/epoch	26s/epoch
3,000	35s/epoch	14s/epoch
1,000	11s/epoch	6s/epoch

(b) Online Running Consumption

Prediction Size	$LSTM(10, 1)$	$LSTM(10, 5)$
12,500	4,953ms	5,706ms
5,000	2,396ms	2,590ms
2,500	1,191ms	1,239ms
500	266ms	386ms
50	38ms	53ms

5 Multi-Scale Entropy Analysis

5.1 Prediction Accuracy Analysis using Multi-Scale Entropy

The predictability of a time series is determined by its complexity and the temporal correlation at different time scales. The traditional entropy measure can be used to quantify the randomness of a signal: the higher the entropy, the more random thus less predictable. However, the traditional entropy measure cannot model the signal complexity and temporal correlation at different time scales. Recently, *Multi-Scale Entropy (MSE)* [5] has been proposed to measure the complexity of physical and physiologic time series. Given a discrete time series $\{x(i), 1 \leq i \leq N\}$, a coarse-grained time series $\{y^{(s)}(j)\}$ can be constructed at scale factor of $s \geq 1$:

$$y^{(s)}(j) \triangleq \frac{1}{s} \sum_{i=(j-1)s+1}^{js} x(i), 1 \leq j \leq N/s.$$

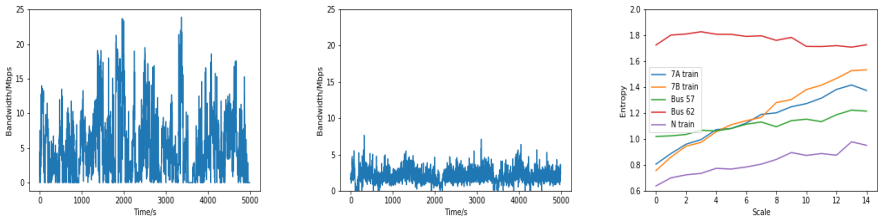
Then the entropy measure of $\mathbf{x}$ at time scale s can be calculated as the entropy of $\mathbf{y}^{(s)}$:

$$H^{(s)}(\mathbf{x}) \triangleq H(\mathbf{y^{(s)}}) = -E[\log p(\mathbf{y^{(s)}})], \tag{5}$$

where $p(\mathbf{y^{(s)}})$ is the probability density of the constructed signal at scale s. By varying s, one can examine the complexity/regularity of $\mathbf{x}$ at different time scales. The Multi-Scale Entropy curve $H^{(s)}(\mathbf{x})$ also reveals the temporal correlation structures of the time series [5].

We apply MSE to study the predictability of network bandwidth under different mobile networking scenarios. MSE can represent the regularity patterns of each scenario. Given a set of scales $\mathcal{S} = [s_0, s_1, \cdots, s_m]$, we generate a MSE vector for scenario i as $MSE_i \triangleq [H^{(s)}(\mathbf{x_i}), s \in \mathcal{S}]$, where $\mathbf{x_i}$ is bandwidth trace from scenario i. MSE_i can be used to analyze the per-scenario prediction accuracy for scenario i, as defined in (2). Additionally, by comparing MSE_i and MSE_j, we can also study the feasibility of cross-scenario prediction between scenarios i and j, as defined in (4). More specifically, we measure the MSE similarity between scenarios i and j as the weighted sum of the correlation coefficient and Euclidean distance between MSE_i and MSE_j. We will demonstrate the connection between MSE and prediction accuracy of both per-scenario and *cross-scenario* predictions next.

5.2 MSE Analysis of NYC MTA Traces



(a) 7A Train Bandwidth (b) Bus 62 Bandwidth (c) MSE Analysis

Fig. 6: Multi-Scale Entropy of Different Mobility Scenarios

We apply MSE analysis to bandwidth from every scenario in New York City MTA traces. Figure 6a and 6b plot the raw bandwidth traces for two sample traces. They present different variability at different scenarios. Figure 6c shows the results of Multi-Scale Entropy for five sample traces. The scale is from 1 to 15. According to the [5], to make the MSE analysis valid, the sequence should be at least 1,000 points at each scale. From the result of Figure 6c, we find that same routes share similar MSE patterns. For example, 7A Train and 7B Train traces were both collected from 7 train but on different days. From the curves of Bus 57 and Bus 62, we find that even though the transportation methods are the same, due to different routes, the MSE patterns can be very different. Table 5a shows the *cross-scenario* prediction accuracy in RMSE. Each row is

for a model trained using data from some mobility scenario, each column is the prediction accuracy for the testset from some mobility scenario. For example, Row 3 & Column 1 shows that the LSTM model trained by Bus 57 data can achieve RMSE of 2.276 when predicting bandwidth for 7A Train testset.

Table 5: Multi-Scale Entropy Analysis

(a) Cross-scenario Prediction RMSE

	7A	7B	B57	B62	N
7A Model	**2.257**	2.060	2.475	0.746	2.837
7B Model	2.267	**2.052**	2.369	0.749	2.817
B57 Model	2.276	2.096	**2.320**	0.754	2.830
B62 Model	2.762	2.205	3.278	**0.719**	3.423
N Model	2.259	2.091	2.382	0.770	**2.808**

(b) MSE Similarity

	7A	7B	B57	B62	N
7A	-	1.223	1.176	1.021	1.106
7B	1.221	-	1.121	1.044	1.080
B57	1.166	1.118	-	1.029	1.123
B62	0.630	0.696	0.665	-	0.690
N	0.984	0.964	1.038	0.907	-

Table 5b shows the MSE similarity between different mobility scenarios. Table 6 reports for each scenario i the correlation between its MSE similarity with other scenarios and the accuracy of *cross-scenario* prediction using models trained for other scenarios. Close to -1 correlations suggest that higher MSE similarity leads to higher accuracy (lower RMSE). Multi-Scale Entropy analysis provides a good measure to explore the possibility of *cross-scenario* prediction, which can be very beneficial for mobility scenarios with limited available data for training Deep learning models.

Table 6: Correlation between MSE Similarity and Cross-scenario Prediction Accuracy

	7A Train	7B Train	Bus 57	Bus 62	N Train
Correlation Value	-0.916	-0.943	-0.945	-0.937	-0.994

6 Conclusion

In this paper, we studied realtime mobile bandwidth prediction. We developed LSTM recurrent neural network models to capture the rich temporal structures in mobile bandwidth traces for accurate prediction. In both next-second and multi-second predictions, LSTM outperforms other state-of-the-art prediction algorithms, such as RLS and Harmonic Mean. Using Multi-Scale Entropy analysis, we investigated the connection between MSE and *cross-scenario* prediction accuracy. Going forward, we will continue our mobile bandwidth measurement campaign. For online bandwidth prediction, we will study how to dynamically select LSTM models trained offline to match the current mobility scenario through adaptive model fusion. We will also study the feasibility of using extra information, e.g. GPS, speed/acceleration sensor readings, to assist mobility scenario identification and model selection. We will also develop LSTM models for the emerging 5G mobile networks. Finally, we will explore data fusion of LSTM models and other prediction models to further improve the prediction accuracy.

References

1. Gers, Felix A., Jrgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." (1999): 850-855.
2. Sundermeyer, Martin, Ralf Schlter, and Hermann Ney. "LSTM neural networks for language modeling." In Thirteenth annual conference of the international speech communication association. 2012.
3. Haykin, Simon S. Adaptive filter theory. Pearson Education India, 2008.
4. Costa, Madalena D., Chung-Kang Peng, and Ary L. Goldberger. "Multiscale analysis of heart rate dynamics: entropy and time irreversibility measures." Cardiovascular Engineering 8.2 (2008): 88-93.
5. Costa, Madalena, Ary L. Goldberger, and C-K. Peng. "Multiscale entropy analysis of complex physiologic time series." Physical review letters 89.6 (2002): 068102.
6. Sun, Yi, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. "CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction." In Proceedings of the 2016 ACM SIGCOMM Conference, pp. 272-285. ACM, 2016.
7. Brown, Robert Grover, and Patrick YC Hwang. Introduction to random signals and applied Kalman filtering. Vol. 3. New York: Wiley, 1992.
8. HSDPA, http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/
9. Bai, Tianyang, Rahul Vaze, and Robert W. Heath. "Analysis of blockage effects on urban cellular networks." IEEE Transactions on Wireless Communications 13, no. 9 (2014): 5070-5083.
10. Bai, Tianyang, Rahul Vaze, and Robert W. Heath. "Using random shape theory to model blockage in random cellular networks." In Signal Processing and Communications (SPCOM), 2012 International Conference on, pp. 1-5. IEEE, 2012.
11. Kurdoglu, Eymen, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu. "Real-time bandwidth prediction and rate adaptation for video calls over cellular networks." In Proceedings of the 7th International Conference on Multimedia Systems, p. 12. ACM, 2016.
12. Tian, Guibin, and Yong Liu. "Towards agile and smooth video adaptation in dynamic HTTP streaming." In Proceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 109-120. ACM, 2012.
13. Qi He, Constantinos Dovrolis, Mostafa H. Ammar, "On the predictability of large transfer tcp throughput, in Proceedings of ACM SIGCOMM, 2005.
14. Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu, "A machine learning approach to tcp throughput prediction", in ACM SIGMETRICS, 2007.
15. Alex J. Smola and Bernhard Schlkopf, A tutorial on support vector regression, Statistics and Computing, vol. 14, no. 3, pp. 199222, Aug. 2004.
16. Yin, Xiaoqi, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. "A control-theoretic approach for dynamic adaptive video streaming over HTTP." In ACM SIGCOMM Computer Communication Review, vol. 45, no. 4, pp. 325-338. ACM, 2015.
17. Winstein, Keith, Anirudh Sivaraman, and Hari Balakrishnan. "Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks." In NSDI, vol. 1, no. 1, pp. 2-3. 2013.
18. Cho, Kyunghyun, Bart Van Merrinboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

19. Hinton, Geoffrey, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." IEEE Signal processing magazine 29, no. 6 (2012): 82-97.
20. Mao, Hongzi, Ravi Netravali, and Mohammad Alizadeh. "Neural adaptive video streaming with pensieve." In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pp. 197-210. ACM, 2017.
21. Jiang, Junchen, Vyas Sekar, and Hui Zhang. "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive." IEEE/ACM Transactions on Networking (TON) 22, no. 1 (2014): 326-340.
22. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).