Computer Networks 182 (2020) 107515

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Realtime mobile bandwidth prediction using LSTM neural network and Bayesian fusion

Lifan Mei [a,*], Runchen Hu [a], Houwei Cao [b], Yong Liu [a], Zifan Han [c], Feng Li [c], Jin Li [c]

[a] NYU WIRELESS, Department of Electrical and Computer Engineering, New York University, Brooklyn, NY 11201, USA
[b] Department of Computer Science, New York Institute of Technology, New York, NY 10023, USA
[c] Huawei Technologies, Nanjing, China

ARTICLE INFO

Keywords:
Bandwidth prediction
Long Short Term Memory
Multi-Scale Entropy
Bandwidth measurement
Model switching
Model Fusion
Bayes Theorem

ABSTRACT

With the increasing popularity of mobile Internet and the higher bandwidth requirement of mobile applications, user Quality of Experience (QoE) is particularly important. For applications requiring high bandwidth and low delay, such as video streaming, video conferencing, and online gaming, etc., if the future bandwidth can be estimated in advance, applications can leverage the estimation to adjust their data transmission strategies and significantly improve the user QoE. In this paper, we focus on accurate bandwidth prediction to improve user QoE. Specifically, We study realtime mobile bandwidth prediction in various mobile networking scenarios, such as subway and bus rides along different routes. The primary method used is Long Short Term Memory (LSTM) recurrent neural network. In individual scenarios, LSTM significantly improves the prediction accuracy of state-of-the-art prediction algorithms, such as Recursive Least Squares (RLS) by 12% in Root Mean Square Error (RMSE) and by 17% in Mean Average Error (MAE). We further developed Multi-Scale Entropy (MSE) to analyze the bandwidth patterns in different mobility scenarios and discuss its connection to the achieved accuracy. For practical applications, we developed Model Switching and Bayes Model Fusion to use pre-trained LSTM models for online realtime bandwidth prediction.

1. Introduction

We have witnessed the tremendous growth of mobile Internet traffic in the recent years. Users are increasingly spending more time on mobile apps and consuming more content on their mobile devices. The mobile traffic growth 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. Providing better Quality of Experience (QoE) to mobile app users is of great significance. 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 [1,2]. 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 upcoming video chunks based on the predicted TCP throughput in a future time window of several seconds [3–5]. If TCP throughput in the next few seconds can be accurately predicted, the selected video rate can maximize the delivery video quality and avoid video freeze. Interactive video conferencing has much tighter delay constraint than streaming. To avoid self-congestion, the available bandwidth on cellular links has to be accurately estimated in realtime, which is used to guide the realtime video coding and transmission strategies [6,7]. Bandwidth overestimate will lead to long end-to-end video delay or freezing, and bandwidth underestimate will lead to unnecessarily poor perceptual video quality. Again, accurate realtime bandwidth prediction is crucial for delivering a good conferencing experience, especially in mobile networking scenarios.

In this paper, we study realtime mobile bandwidth prediction using Long Short Term Memory (LSTM) [8] recurrent neural network and Bayes fusion. 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

* Corresponding author.
E-mail addresses: lifan@nyu.edu (L. Mei), hcao02@nyit.edu (H. Cao), yongliu@nyu.edu (Y. Liu).

https://doi.org/10.1016/j.comnet.2020.107515
Received 2 March 2020; Received in revised form 31 July 2020; Accepted 24 August 2020
Available online 28 August 2020

used in Natural Language Processing (NLP), speech recognition, and time series processing [9,10]. 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?*
4. *How should one switch between different pre-trained models when user mobility pattern changes? Is it possible to fuse the predictions from multiple models to adapt to mobility pattern changes?*

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 times of the day.[1] The traces are described in detail in Section 4.
- We developed LSTM models for realtime one-second ahead and multi-second ahead bandwidth predictions. Through extensive experiments on our own dataset and the HSDPA dataset [11], we demonstrated that LSTM significantly outperforms the existing realtime bandwidth prediction algorithms. Our LSTM models and their performance are presented in Sections 3 and 4.
- We systematically evaluated the sensitivity of LSTM models to different mobility scenarios by comparing the accuracy 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 analysis is presented in Section 5.
- We designed *model switching*, which selects the best-performing LSTM model in recent history, and can converge to the model that fits the most to the current mobility scenario. We also designed *model fusion*, which generates predictions through Bayesian fusion of outputs of multiple models, and can make smooth transition from one mobility scenario to another. Model switching and fusion are presented in Section 6.

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 a 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. In Section 6, we present Model Switching and Model Fusion to address mobility pattern changes. The paper is concluded with future work in Section 7.

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

2. Related work

Realtime bandwidth prediction has been a challenging problem for the networking community. A simple history-based TCP throughput estimation algorithm was proposed in [12]. Authors of [13] proposed to train a Support Vector Regress (SVR) model [14] to predict TCP throughput based on the measured packet loss rate, packet delay, and the size of the file to be transmitted. In the context of DASH video streaming, in [3], we adopted the prediction algorithm in [12] to guide realtime chunk rate selection, and used a customized SVR model similar to [13] for DASH server selection. Authors of [15] and [4] used the Harmonic Mean of TCP throughput for downloading the previous five chunks as the TCP throughput prediction for downloading the next chunk. In [5], 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. For mobile bandwidth measurement, [16,17], and [18] provide detailed guidance on the methodology and precautions.

Constantly providing a high level of QoS in 4G/5G mobile networks is challenging, e.g. [19] and [20]. In the context of video conferencing, in [7], a cellular link is modeled as a single-server queue driven by a doubly-stochastic service process. The 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 [6], we used an adaptive filter, namely Recursive Least Squared (RLS), to make realtime bandwidth prediction. We showed that RLS achieves good prediction accuracy on volatile cellular links. Authors of [21] used Random Forest to make realtime bandwidth predictions in LTE-A mobile networks.

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. Deep Neural Networks (DNNs) have recently been used to solve networking problems. Authors of [22] use DNN to make offloading decision for edge computing. Authors of [23] use DNN to forecast network traffic matrix. A recent work on Deep Reinforcement Learning (DRL) based DASH [24] takes historical bandwidth samples as part of its input state vector for DRL to directly generate video chunk rate selection. 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

In this section, we formulate the realtime bandwidth prediction problem, and introduce our LSTM prediction models. The key parameters and notations are listed in Table 1.

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 series of $\{x(t), t = 1, 2, \dots .\}$. 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:

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

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 Method, $\hat{x}(t+\tau) = h/\sum_{k=0}^{h-1} 1/x(t-k)$, etc., to more sophisticated signal processing approaches, such as Kalman filter [25] and Recursive Least Squares (RLS) [26]. In [6], 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

Table 1
Key parameters and notations.

Symbol	Description
$x(t)$	Bandwidth available at time t
τ	Future time index
$\mathbf{y}$	Predicted bandwidth in a future time window
θ^i	Model parameters trained by data from scenario i
m	Future time horizon for prediction
n	History window size
z^s	Aggregated time series at scale s
$H^{(s)}(\mathbf{x})$	Entropy measure of x at scale s
$\mathcal{A}$	Set of pre-trained LSTM models
$\mathcal{I}$	Subset of pre-trained LSTM models
$\hat{\mathbf{X}}^I(t)$	Predictions of models in $\mathcal{I}$ at time t
$\mu_i(t)$	Mean error of model i at time t
$\sigma_i^2(t)$	Error variance of model i at time t

finds the coefficients $\boldsymbol{\omega}$ that minimizes a weighted linear least squares cost function. We showed in [6] that RLS achieves higher 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, the LSTM network has recently emerged as a powerful tool for exploring temporal structures in sequential data.

As illustrated in Fig. 1(a), an LSTM network consists of layers of LSTM units. As illustrated in Fig. 1(b), 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 notation "gate". There are connections between these gates and the cell. Detailed LSTM reviews can be found in [8,27].

The input to our LSTM bandwidth prediction network is the recent bandwidth measurements, i.e, $\mathbf{x} = [x(t), x(t-1), \ldots, x(t-n+1)] \in R^n$, the output is the predicted bandwidth in a future time window $\mathbf{y} = [\hat{x}(t+1), \hat{x}(t+2), \ldots, \hat{x}(t+m)] \in R^m$. 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 Fig. 1(a), we build an LSTM network with one input layer, one output layer and two hidden layers, each with 256 and 128 LSTM units respectively.[2] 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 $\boldsymbol{\theta}$, 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. We can use bandwidth data collected from scenario i to train a LSTM network with parameters $\boldsymbol{\theta}^{(i)}$, and then use it to predict bandwidth for scenario i, i.e.,

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

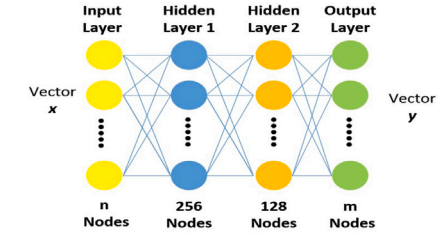
Another option is to train one universal LSTM network with parameters $\boldsymbol{\theta}^{(0)}$ using all data collected from all scenarios. Then the trained universal LSTM model can be used to predict bandwidth in all scenarios, i.e.,

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

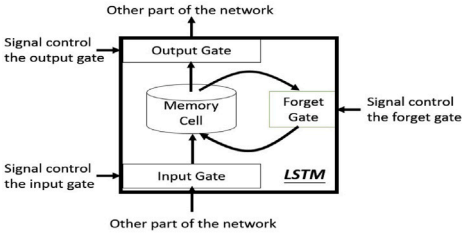
The third option is to train a LSTM network using data from scenario i, then use it to predict bandwidth in scenario j.

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

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 training, we use every consecutive six bandwidth measurements as one training data point. The bandwidth measurements of the first five seconds form the input vector, and the bandwidth of the sixth second is the output label. Likewise, for the general multiple seconds prediction, i.e., predicting the future bandwidth of the next m seconds based on the bandwidth of the previous n seconds, we use bandwidth measurements of every consecutive $n + m$ seconds as one data point. The first n measurements form an input vector, and the last m measurements form an output label vector.



(a) LSTM Network Architecture



(b) Internal Structure of LSTM Unit

Fig. 1. LSTM network for realtime bandwidth prediction.

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 [11] dataset from the University of Oslo, Norway. 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 1000 ms, and the duration for each trace ranges from 500 to 1000 s. However, we later found that the bandwidth traces are too short for Multi-Scale Entropy (MSE) analysis. We also collected our own long bandwidth traces on New York City MTA bus and subway. Fig. 2 shows some sample routes for our bandwidth

[2] We also tried an 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.

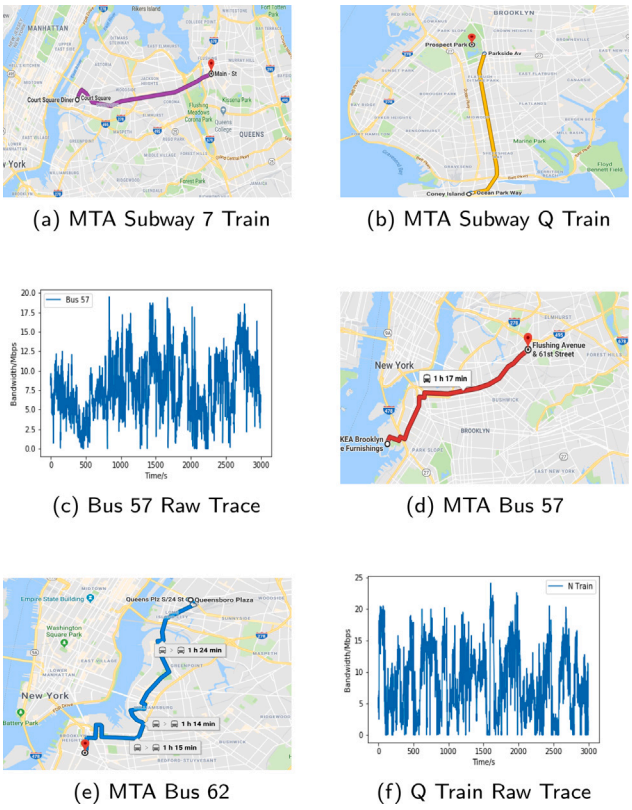


Fig. 2. New York City self-measured bandwidth.

collection, including Subway 7 Train, Subway Q Train, Bus B57, and B62. On each route, we conducted multiple experiments at different times of the day. For each experiment, we connect an LTE mobile phone with an unlimited data plan to a remote server in our lab. We run $iPerf$ and record TCP throughput every 1000 millisecond. All the bandwidth samples are logged on the server side. The duration of each trace ranges from 10,000 to 20,000 s. 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. Fig. 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 [28] 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 model performance.

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 results of the three algorithms on our NYU-METS Dataset is reported in Table 2. (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 improvements of LSTM over RLS and Harmonic Mean in RMSE are 12.4% and 25.9% respectively. For MAE, the accuracy improvements 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.

Table 3 compares the accuracy of per-scenario LSTM with RLS on the HSDPA dataset. The unit is kbps. LSTM still outperforms RLS in

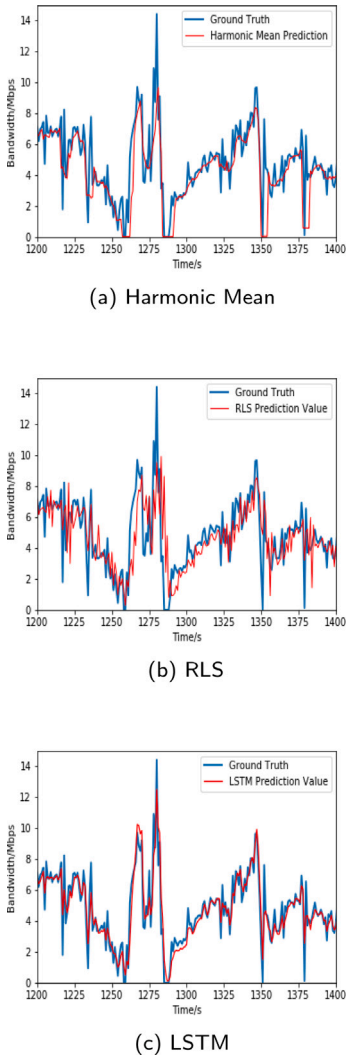


Fig. 3. Harmonic Mean, RLS and LSTM predictions on subway 7 Train.

Table 2
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 Impro over RLS	**14.0%**	**6.7%**	**11.8%**	**21.2%**	**8.2%**
Relative MAE Impro over RLS	**13.6%**	**5.9%**	**11.9%**	**21.6%**	**11.0%**
Relative RMSE Impro over Harmonic	**31.8%**	**26.7%**	**20.4%**	**31.1%**	**19.5%**
Relative MAE Impro over Harmonic	**24.9%**	**19.7%**	**15.8%**	**27.7%**	**18.9%**

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,

Table 3
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 Impro	**17.3%**	**10.6%**	**12.2%**	**17.8%**	**17.4%**	**9.3%**
Relative MAE Impro	**16.5%**	**16.9%**	**9.5%**	**19.3%**	**11.0%**	**10.6%**

Table 4
Prediction RMSE on RLS and LSTM.

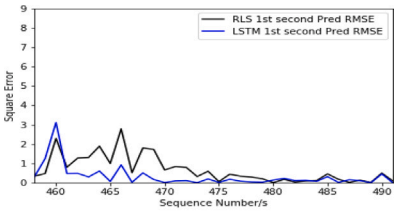
	1st s	2nd s	3rd s	4th s	5th s
RLS	2.57	2.88	3.16	3.53	3.76
$LSTM(10,1)$	**2.26**	–	–	–	–
$LSTM(10,2)$	2.27	**2.66**	–	–	–
$LSTM(10,3)$	2.29	2.68	**2.96**	–	–
$LSTM(10,4)$	2.33	2.69	2.97	**3.21**	–
$LSTM(10,5)$	2.40	2.71	2.98	3.22	**3.40**
Improvement over RLS	**13.7%**	**8.2%**	**6.8%**	**9.9%**	**10.6%**

and Car, then test its accuracy on individual transportation scenarios. However, its 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 do not report the detailed statistics here. We defer the discussion on *cross-scenario* prediction to the next section, and will investigate further on universal prediction in our future research.

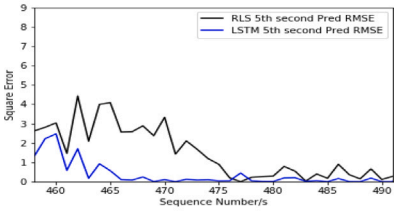
4.3. Multi-second prediction

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, the 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 $t+i$, $1 \le i \le m$. To make RLS generate prediction i seconds ahead, we simply update RLS parameters by using the bandwidth of i seconds ahead, instead of the next second, as the targeted output.

Table 4 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. The best prediction accuracy for interval i is achieved by $LSTM(10,i)$, marked in bold fonts. Intuitively, $LSTM(n,m)$ 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 ($m_1 \le m_2 < m$). Figs. 4(a) and 4(b) 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 RMSE of LSTM is lower than RLS most of the time. The accuracy improvement of LSTM is more prominent for the five second prediction interval. Fig. 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 specific prediction intervals; LSTM's accuracy decays slower than RLS as the interval increases.



(a) One Second Prediction



(b) Five Second Prediction

Fig. 4. RLS vs LSTM multi-second prediction.

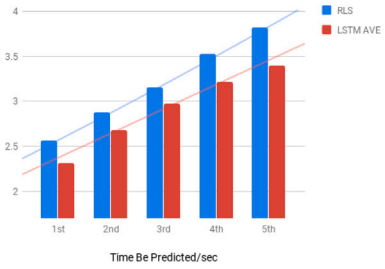


Fig. 5. Impact of prediction interval on LSTM and RLS.

Table 5
Computation time of training and prediction.

(a) Offline training		
Trainsize	Batchsize = 2	Batchsize = 4
13,000	120 s/epoch	62 s/epoch
10,000	95 s/epoch	50 s/epoch
5,000	49 s/epoch	26 s/epoch
3,000	35 s/epoch	14 s/epoch
1,000	11 s/epoch	6 s/epoch
(b) Online prediction		
Prediction size	$LSTM(10,1)$	$LSTM(10,5)$
12,500	4953 ms	5706 ms
5,000	2396 ms	2590 ms
2,500	1191 ms	1239 ms
500	266 ms	386 ms
50	38 ms	53 ms

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: 4th Gen Intel Core i5-4210U (1.70 GHz 1600 MHz 3 MB). Neural Network Structure: Hidden Layer 1 & 2 have 256 and 128 nodes, respectively. The offline training and online prediction computation time is presented in Table 5. Even though the offline training time is long, as shown in Table 5b, the online prediction consumption time is short. It takes less than six seconds to predict 12,500 five-second bandwidth vectors 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.

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)* [29] has been proposed to measure the complexity of physical and physiologic time series. Given a discrete-time series $\{x(i), 1 \le i \le N\}$, a coarse-grained time series $\{z^{(s)}(j)\}$ can be constructed at scale factor of $s \ge 1$:

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

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

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

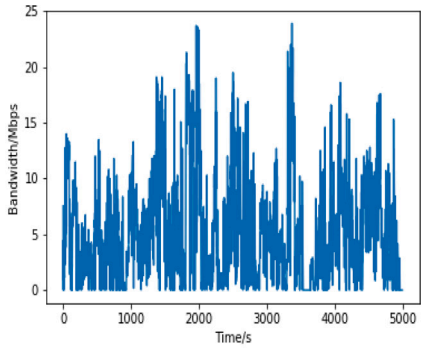
where $p(\mathbf{z^{(s)}})$ is the probability density of the constructed signal at scale s. By varying s, one can examine the complexity/regularity of $\mathbf{x}$ at different time scales. The Multi-Scale Entropy curve $H^{(s)}(\mathbf{x})$ also reveals the temporal correlation structures of the time series [29].

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 $S = [s_0, s_1, \ldots, s_m]$, we generate a MSE vector for scenario i as $MSE_i \triangleq [H^{(s)}(\mathbf{x_i}), s \in S]$, 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). 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.

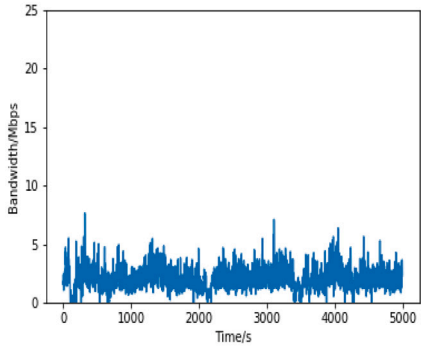
5.2. MSE analysis of NYC MTA traces

We apply MSE analysis to bandwidth traces from every scenario in New York City MTA dataset. Figs. 6(a) and 6(b) plot the raw bandwidth traces for two sample traces. They present different variability on different scenarios. Fig. 6(c) shows the results of Multi-Scale Entropy for five sample traces. The scale is from 1 to 15. According to [29], to make the MSE analysis valid, the sequence should be at least 1000 points at each scale. From the result of Fig. 6(c), we find that the same routes share similar MSE patterns. For example, 7A Train and 7B Train traces were 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.

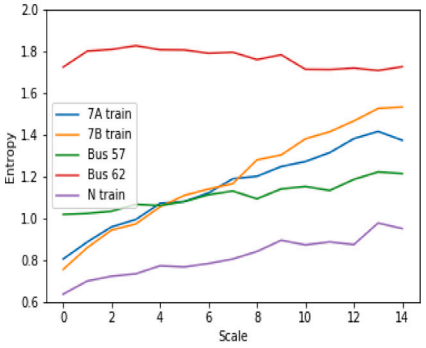
Table 6a shows the *cross-scenario* prediction accuracy in RMSE. Each row is for a model trained using data from some mobility scenario; each column is the prediction accuracy for the testset from some mobility scenario. For example, Row 3 & Column 1 shows that the LSTM model trained by Bus 57 data can achieve RMSE of 2.276 when predicting bandwidth for the 7A Train testset. The bold font indicates the best prediction model in each scenario. In Table 6a, the best models are distributed diagonally, which means, the best prediction model for a mobility scenario is trained using data from the same scenario. Table 6b shows the MSE similarity between different mobility scenarios. Table 7 reports for each scenario i the correlation between its MSE similarity with the other scenarios and the accuracy of *cross-scenario* prediction using models trained for the other scenarios. Correlation close to "−1"



(a) 7A Train Bandwidth



(b) Bus 62 Bandwidth



(c) MSE Analysis

Fig. 6. Multi-scale entropy of different mobility scenarios.

suggests that higher MSE similarity between two scenarios leads to higher cross-scenario prediction accuracy (lower RMSE), and vice-versa. In Table 7, all the correlation values are high. That is to say, by analyzing the temporal pattern similarity between bandwidth traces, we can infer the accuracy of cross-scenario prediction. 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.

6. Model selection and fusion

In the previous sections, we introduced and evaluated the feasibility of LSTM models on bandwidth prediction in mobile scenarios. In addition, the MSE analysis reveals the correlation between cross-scenario

Table 6
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	–

prediction accuracy and the similarity between mobility scenarios. For practical applications, a set of LSTM models will be trained offline for representative mobility scenarios. A subset of LSTM models will be chosen to run online to generate realtime prediction. *The main questions are: (1) which pre-trained models will be chosen to run online? (2) how to fuse the predictions of individual models to generate more accurate prediction? (3) how to adapt model selection and fusion to user mobility pattern changes?*

More specifically, given a set $\mathcal{A}$ of pre-trained LSTM models, at time t, one can select a subset $\mathcal{I} \subset \mathcal{A}$ of models and run them in parallel to generate a set of predictions $\{\hat{x}^{(i)}(t), i \in \mathcal{I}\}$ based on the recent bandwidth measurement $\{x(t-1), \ldots, x(t-n)\}$. The final prediction is calculated based on the predictions of the selected models:

$$\hat{x}(t) = \mathbf{f}(\{\hat{x}^{(i)}(t), i \in \mathcal{I}\}),$$

where $\mathbf{f}(\cdot)$ is the function representing fusion strategy.

6.1. Model switching

One simple strategy is *model switching*, i.e., only using the prediction from the best-performing model in the recent past as the final prediction, in other words,

$$\hat{x}(t) = \hat{x}^{(i^*)}(t), \quad i^* = \underset{i \in \mathcal{I}}{\mathbf{argmin}} \sum_{k=1}^{W} \rho_k \left| \hat{x}^{(i)}(t-k) - x(t-k) \right|, \tag{6}$$

where W is the width of the history window, and ρ_k is the weight assigned to the prediction error k steps back. The intuition is to select the most suitable LSTM model based on the performance of the candidate models in the previous W time slots.

Obviously, the history window size W is a critical parameter for the performance and responsiveness of the model switching algorithm. If the window size is too small, e.g., 1 s, the model switching algorithm would be too sensitive to model performance oscillation, and cause too frequent model switches and poor prediction performance. If the window size is too large, e.g. 20 min, the model switching algorithm will be relatively stable and can be locked into the most suitable model if the mobility scenario is stable. On the other hand, it will become too sluggish to respond to sudden mobility scenario changes, e.g., a user transfers from a bus to a subway. The model switching algorithm will take a long time to converge to the most suitable model for the new scenario. We will demonstrate this trade-off through our evaluation results.

6.2. Bayes model fusion

A more sophisticated way is to design the fusion function within the information fusion framework. Information Fusion is the process of integrating multiple information sources, such as readings from multiple sensors, to produce more consistent, accurate, and complete information than that provided by any individual source. For our bandwidth prediction problem, different LSTM models are trained with different training data from different mobility scenarios. As a result, they have different capabilities to capture different temporal patterns in bandwidth history for future bandwidth prediction. Therefore, predictions from different LSTM models contain complementary information, and can be fused together to generate more accurate bandwidth prediction. Our model fusion uses the predictions of several chosen LSTM Models as the sources of input information, and the output is the final prediction value. While model switching only relies on the single best-performing LSTM model, model fusion synthesizes the final prediction based on the predictions of all the chosen models, can potentially generate more accurate and more robust prediction. We resort to the Bayes framework for prediction fusion. In particular, we calculate the posterior probability of future bandwidth given the predictions generated by different LSTM models.

Given the model predictions $\hat{\mathbf{X}}^{\mathcal{I}}(t) \triangleq \{\hat{x}^{(i)}(t), i \in \mathcal{I}\}$, we want to estimate the posterior distribution of the true value $P\left(x(t) = x|\hat{\mathbf{X}}^{\mathcal{I}}(t)\right)$. According to the Bayes Theorem,

$$P\left(x(t) = x|\hat{\mathbf{X}}^{\mathcal{I}}(t)\right) = \frac{P\left(\hat{\mathbf{X}}^{\mathcal{I}}(t)|x(t) = x\right) P(x(t) = x)}{P\left(\hat{\mathbf{X}}^{\mathcal{I}}(t)\right)} \tag{7}$$

Assuming that the predictions are conditionally independent given the true value, then we have

$$P\left(\hat{x}^{(1)}(t), \ldots \hat{x}^{(\mathcal{I})}(t)|x(t) = x\right) = \prod_{i=1}^{\mathcal{I}} P\left(\hat{x}^{(i)}(t)|x(t) = x\right) \tag{8}$$

Based on Eq. (8) and the law of total probability, the posterior distribution of $x(t)$ can be calculated as:

$$P\left(x(t)|\hat{\mathbf{X}}^{\mathcal{I}}(t)\right) = \frac{\prod_{i=1}^{\mathcal{I}} P\left(\hat{x}^{(i)}(t)|x(t) = x\right) P(x(t) = x)}{\sum_{j=1}^{m} \prod_{i=1}^{\mathcal{I}} P\left(\hat{x}^{(i)}(t)|x(t) = x_j\right) P\left(x(t) = x_j\right)} \tag{9}$$

Then we can generate the maximum a posterior probability (MAP) estimate as our final estimation:

$$\hat{x}(t) = \underset{x}{\mathbf{argmax}} P\left(x(t) = x|\hat{\mathbf{X}}^{\mathcal{I}}(t)\right).$$

If we further assume the priori of $x(t)$ is a uniform distribution, the posterior distribution of $x(t)$ can be simplified to:

$$P\left(x(t) = x|\hat{\mathbf{X}}^{\mathcal{I}}(t)\right) = \frac{\prod_{i=1}^{\mathcal{I}} P\left(\hat{x}^{(i)}(t)|x(t) = x\right)}{\sum_{j=1}^{m} \prod_{i=1}^{\mathcal{I}} P\left(\hat{x}^{(i)}(t)|x(t) = x_j\right)}, \tag{10}$$

and the MAP estimate becomes the Maximum Likelihood estimate (MLE):

$$\hat{x}(t) = \underset{x}{\mathbf{argmax}} \prod_{i=1}^{\mathcal{I}} P\left(\hat{x}^{(i)}(t)|x(t) = x\right).$$

Instead of using MAP or MLE estimate, we use the expectation of the posterior distribution of $x(t)$ as our estimate.

$$\hat{x}(t) = \sum_{j=1}^{m} x_j P\left(x(t) = x_j|\hat{\mathbf{X}}^{\mathcal{I}}(t)\right). \tag{11}$$

We further assume the error distribution of each model follows Gaussian: $P\left(\hat{x}^{(i)}(t) - x(t)|x(t) = x\right) \sim \mathcal{N}(\mu_i(t), \sigma_i^2(t))$, and estimate $\mu_i(t)$

Table 7
Correlation between MSE similarity and cross-scenario prediction accuracy.

7A Train	7B Train	Bus 57	Bus 62	N Train
−0.916	−0.943	−0.945	−0.937	−0.994

Table 8
Model switching performance at different window sizes.

(a) Window size = 1

Data	Test_Ave	Ideal LSTM		Model switch			Random switch		
		RMSE	RMSE_Ratio	RMSE	RMSE_Ratio	Gap_LSTM	RMSE	RMSE_Ratio	Gap_Random
7A	6.4055	2.2754	35.52%	2.3996	37.46%	+5.46%	2.5704	40.13%	−6.64%
B57	10.0388	2.3164	23.07%	2.4988	24.89%	+7.88%	2.8715	28.60%	−12.98%
B62	2.5499	0.7336	28.77%	0.7723	30.29%	+5.28%	0.7690	30.16%	+0.43%
7A_B62	4.1867	1.5956	38.11%	1.6903	40.37%	+5.93%	1.7972	42.92%	−5.95%
7A_B57	8.3336	2.3263	27.91%	2.4653	29.58%	+5.97%	2.6645	31.97%	−7.48%
B57_B62	5.9507	1.6719	28.10%	1.7937	30.14%	+7.29%	2.0355	34.21%	−11.88%

(b) Window Size = 10

Data	Test_Ave	Ideal LSTM		Model switch			Random switch		
		RMSE	RMSE_Ratio	RMSE	RMSE_Ratio	Gap_LSTM	RMSE	RMSE_Ratio	Gap_Random
7A	6.4251	2.2788	35.47%	2.3451	36.50%	+2.91%	2.5745	40.07%	−8.91%
B57	10.0314	2.3197	23.12%	2.4429	24.35%	+5.31%	2.8743	28.65%	−15.01%
B62	2.5508	0.7341	28.78%	0.7763	30.43%	+5.76%	0.7696	30.17%	+0.87%
7A_B62	4.1915	1.5963	38.08%	1.6441	39.23%	+3.00%	1.7981	42.90%	−8.56%
7A_B57	8.3463	2.3280	27.89%	2.3913	28.65%	+2.72%	2.6665	31.95%	−10.32%
B57_B62	5.9409	1.6728	28.16%	1.7522	29.49%	+4.75%	2.0357	34.27%	−13.93%

(c) Window Size = 100

Data	Test_Ave	Ideal LSTM		Model switch			Random switch		
		RMSE	RMSE_Ratio	RMSE	RMSE_Ratio	Gap_LSTM	RMSE	RMSE_Ratio	Gap_Random
7A	6.5600	2.2899	34.91%	2.3325	35.56%	+1.86%	2.5914	39.50%	−9.99%
B57	9.9867	2.3134	23.16%	2.3361	23.39%	+0.98%	2.8794	28.83%	−18.87%
B62	2.5623	0.7385	28.82%	0.7462	29.12%	+1.04%	0.7738	30.20%	−3.57%
7A_B62	4.2097	1.5889	37.74%	1.6157	38.38%	+1.69%	1.7922	42.57%	−9.85%
7A_B57	8.4441	2.3339	27.64%	2.3711	28.08%	+1.60%	2.6758	31.69%	−11.39%
B57_B62	5.8557	1.6565	28.29%	1.6720	28.55%	+0.93%	2.0228	34.54%	−17.34%

(d) Window Size = 200

Data	Test_Ave	Ideal LSTM		Model switch			Random switch		
		RMSE	RMSE_Ratio	RMSE	RMSE_Ratio	Gap_LSTM	RMSE	RMSE_Ratio	Gap_Random
7A	6.6472	2.3196	34.90%	2.3507	35.36%	+1.34%	2.6397	39.71%	−10.95%
B57	9.9589	2.3404	23.50%	2.3578	23.68%	+0.75%	2.8477	28.59%	−17.20%
B62	2.5683	0.7439	28.96%	0.7505	29.22%	+0.89%	0.7744	30.15%	−3.09%
7A_B62	4.2000	1.5899	37.86%	1.6085	38.30%	+1.17%	1.7956	42.75%	−10.42%
7A_B57	8.5230	2.3479	27.55%	2.3687	27.79%	+0.89%	2.7063	31.75%	−12.47%
B57_B62	5.7690	1.6587	28.75%	1.6703	28.95%	+0.70%	1.9829	34.37%	−15.76%

(e) Window Size = 1000

Data	Test_Ave	Ideal LSTM		Model switch			Random switch		
		RMSE	RMSE_Ratio	RMSE	RMSE_Ratio	Gap_LSTM	RMSE	RMSE_Ratio	Gap_Random
7A	7.6957	2.4845	32.28%	2.4825	32.29%	−0.08%	2.9225	37.98%	−15.06%
B57	10.1685	2.4819	24.41%	2.4819	24.41%	+0.00%	3.0676	30.17%	−19.09%
B62	2.5891	0.7194	27.78%	0.7196	27.79%	+0.03%	0.7432	28.70%	−3.18%
7A_B62	4.0690	1.4994	36.85%	1.4984	36.86%	−0.07%	1.744	42.86%	−14.08%
7A_B57	9.2563	2.4092	26.03%	2.4251	26.20%	+0.66%	2.8058	30.31%	−13.57%
B57_B62	5.1321	1.5881	30.94%	1.5930	31.04%	+0.31%	1.9164	37.34%	−16.88%

and $\sigma_i^2(t)$ using error samples from a recent history window of size W:

$$\mu_i(t) \approx \bar{\mu}_i(t) = \frac{1}{W}\sum_{k=1}^{W} \hat{x}^{(i)}(t-k) - x(t-k), \tag{12}$$

$$\sigma_i^2(t) \approx \frac{1}{W-1}\sum_{k=1}^{W} \left(\hat{x}^{(i)}(t-k) - \bar{\mu}_i(t-k)\right)^2. \tag{13}$$

To calculate the final estimation according to (11), we focus on a range of possible values as:

$$\left[\min_{i\in I}\left(\hat{x}^{(i)}(t) + \mu_i(t) - 3\sigma_i(t)\right), \max_{i\in I}\left(\hat{x}^{(i)}(t) + \mu_i(t) + 3\sigma_i(t)\right)\right], \tag{14}$$

where $3\sigma_i(t)$ reflects the fact that in a Gaussian distribution, approximately 99.7% of the probability is distributed within three times of the standard deviation from the mean value. We further equally divide the range into m bins, and calculate the probability mass for the center of each bin according to (10), and finally the estimated average according to (11).

6.3. Evaluation

6.3.1. Transfer emulation

For evaluation, we use candidate models trained from 7A, B57, and B62 traces. We concatenate two original traces from two mobility scenarios into one synthetic testing trace to emulate user transfer from one transportation platform, e.g. bus, to another, e.g., subway. For Model Switching, we use Random Switching as a baseline, which randomly selects a model to generate prediction for the next time slot. To make results repeatable, we use round-robin scheduling to approximate random switching.

6.3.2. Model switching performance

In Table 8, "Data" is evaluation dataset. "Test_Ave" is the average bandwidth of the testset in Mbps. "Ideal LSTM" represents the performance of the most suitable LSTM Model for the current scenario. If the testset contains transfer, "Ideal LSTM" represents the ideal instant switching to the most appropriate model. "Model Switch" is for

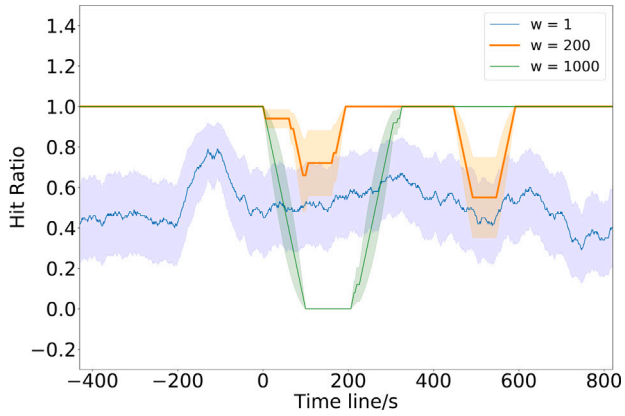


Fig. 7. Hit Ratio comparison among various window size.

our model switching algorithm described in (6). "RMSE" and "Ratio" are the prediction Root Mean Square Error and the percentage over "Test_Ave" for the corresponding algorithms, respectively. "Gap_LSTM" and "Gap_Random" represent the relative RMSE changes compared to the ideal LSTM algorithm and the Random Switching respectively, and "+/-" means how much worse or better our proposed model switching algorithm is.

As Table 8 shows, our model switching algorithm can already meet the needs of practical applications. Secondly, it shows that window size has a big impact on the switching performance. We can find that Model Switching outperforms Random Switching, with 10% or more improvement. The larger the window size is, the bigger the performance improvement. Note that for B62, the performance is not as remarkable as the others because other models have similar performance as the B62 model on the B62 trace. When compared with "Ideal LSTM", the "Gap_LSTM" at different the Window Sizes, except the extreme cases like Window Size = 1 or 10 (switch too often to converge to the right model), can even reach under one to two percents. As the window size increases from 1 to 1000, model switching algorithm closely approaches the ideal LSTM performance, especially in the non-transfer scenarios (7A, B57, and B62). For example, for B62, the performance gap reduces from +5.28% (Window Size = 1) to +0.03% (Window Size = 1000). But for transfer scenarios (7A_B62, 7A_B57, and B57_B62), the performance is relatively worse. This suggests that our model switching algorithm can be used in practical applications.

Other than the average performance over the whole period, we also care about the transient performance during the switching period. Even if the long-term average prediction RMSE looks good, bad prediction during the switching period may also result in terrible user QoE.

Fig. 7 plots the transient performance of our model switching algorithm around the emulated user transfer event. We use an emulated transfer from B62 to B57 as an example here. In Fig. 7, X-axis is the timeline; "0" is the transfer moment. "W" is the switching window size. Y-axis is the "Hit Ratio", which means the percentage that switching algorithm selects the most suitable model during the past 100 s. At every moment, the corresponding point represents a Hit Ratio, and the shadow is the variance of the 100-seconds binary (Non-Hit/Hit) sequence. We can find that, for "W = 1000", our model switching algorithm can converge to the most suitable model and lock into it for a long time, but once the transfer happens, its hit ratio goes down to "0" (wrong model) for a very long time. As for "W = 1", the emulated transfer does not affect its Hit Ratio. However, small window size leads to too frequent model switches; the variance that was marked as shadows in Fig. 7 can show that. As for "W = 200", the performance is between "W = 1" and "W = 1000". It selects model accurately and it responds to transfer quickly. In this example, window size "W = 200" is a better choice. Generally speaking, the model switching algorithm makes sense for long-term performance; the algorithm can converge to the most suitable LSTM model and results in good prediction performance. The transient performance immediately after the transfer depends on the window size.

6.3.3. Model fusion performance

Model fusion is more complicated and takes into account more factors than Model Switching. We compare the performance of Bayes Model Fusion with Model Switching in our evaluation. We also investigate the impact of fusion sliding window size W. The error distribution estimation in (12) (13) is affected by the window size. If the window size is too small, it may result in inaccurate error distribution, and consequently bad fusion performance. If the window is too large, it may become too sluggish to react to transfers. Similar to Model Switching performance, we study both the long-term average performance as well as the transient performance after transfer.

Similar to Table 8, Table 10 reports the long-term average performance. "LSTM" is for the most suitable LSTM model for the current scenario. For a trace with transfers, "LSTM" refers to always using the model corresponding to each scenario. "Model Switch" is for our model switching algorithm. "Model Fusion" is for our proposed Bayes model fusion algorithm. "Gap_L" and "Gap_SW" represent relative RMSE changes compared to the ideal LSTM and our model switching algorithm, respectively, and "+/-" means how much model fusion is worse/better than the corresponding algorithm. We used the bin size of 0.2 to discretize the target range defined in (14). To consider more recent prediction errors, we double the weights for prediction errors within the latest 30% time window when estimating the error distribution.

For comparison between Bayes Model Fusion and Model Switching, we keep their Window Sizes identical. Models we used here are 7A, B57, and B62. In the single scenario (7A, B57, and B62) and scenarios that contain transfers, Model Fusion has good performance in small Window Sizes like 50 and 100, the "Gap_SW" values are negative, which means Model Fusion is better than Model Switching. At Window Size 50, for trace 7A and 7A_B62, for single scenario and scenarios that contain transfers, model fusion are 3.77% and 3.66% better than Model Switching respectively. But for larger sliding window size like 1000, for both kinds of scenarios, Model Fusion are worse than Model Switching in almost all cases.

To evaluate the short-term performance around transfers, we synthesize a trace to emulate cyclic transfer: from B62 transfers to B57, and from B57 transfers to 7A, and from 7A transfers back to B62. In Table 9, for Fusion and Switch, we use the previous 50 s as a sliding window. "Time period after transfer" is the time window used for transient performance evaluation. We set it to 10, 20, 30, and 50 s. According to the result in "Gap_SW", Fusion is around 7% better than Model Switch in average, which means Fusion outperforms Model Switching in the transition period.

Intuitively, Bayes based fusion is a relatively complicated multi-sources decision making process. At each moment, the final prediction is generated based on predictions of all models. But in Model Switching, each prediction is picked from one model. In the transient period after a transfer from one mobility scenario to another, model fusion is more robust than model switching. On the other hand, when a user stays with a mobility scenario for a long time, model switching with a large window can converge to the LSTM model most appropriate for the scenario.

7. Conclusion

In this paper, we studied realtime mobile bandwidth prediction using LSTM deep neural networks and Bayes fusion. We collected a rich set of bandwidth traces from different mobility scenarios. We developed LSTM recurrent neural network models to capture the temporal structures in mobile bandwidth traces for accurate prediction.

Table 9
Transient Performance after Transfer: fusion vs. switching.

(a) Time period after transfer = 10				
Data	Test_Ave	Fusion	Model switch	Gap_SW
B62->B57	12.305	1.1934	1.6450	−27.45%
B57->7A	1.4508	1.1439	1.1254	+1.64%
7A->B62	2.2770	0.5344	0.5603	−4.63%
(b) Time period after transfer = 20				
Data	Test_Ave	Fusion	Model switch	Gap_SW
B62->B57	12.4525	1.2555	1.4611	−14.07%
B57->7A	1.9904	0.8911	0.9222	−3.36%
7A->B62	2.4240	0.6196	0.6268	−1.15%
(c) Time period after transfer = 30				
Data	Test_Ave	Fusion	Model switch	Gap_SW
B62->B57	12.1997	1.1931	1.3491	−11.56%
B57->7A	2.8603	1.0754	1.1220	−4.15%
7A->B62	2.2000	0.5485	0.5709	−3.92%
(d) Time period after transfer = 50				
Data	Test_Ave	Fusion	Model switch	Gap_SW
B62->B57	11.6208	1.3019	1.3658	−4.67%
B57->7A	3.1592	1.2954	1.4028	−7.66%
7A->B62	2.0882	0.5505	0.5645	−2.48%

For both next-second and multi-second ahead predictions, LSTM outperforms other state-of-the-art prediction algorithms, such as RLS, by 12% in Root Mean Square Error (RMSE) and by 17% in Mean Average Error (MAE). Using Multi-Scale Entropy analysis, we investigated the connection between MSE and cross-scenario prediction accuracy. For practical applications, we studied how to dynamically select pre-trained LSTM models to match the current mobility scenario through Model Switching. We also designed Bayes Model Fusion to make fused predictions based on parallel predictions from multiple models. Our results demonstrate that LSTM models can effectively mine the rich temporal patterns in mobile bandwidth traces for accurate realtime bandwidth prediction. Due to low time complexity for realtime inference, our LSTM models can be readily adopted by mobile apps, such as video streaming, conferencing, and online gaming, to improve their user QoE.

For future work, we will continue our mobile bandwidth measurement campaign. We will also study the feasibility of using additional information, e.g., GPS data, signal strength, and speed/acceleration sensor readings, to assist mobility scenario identification and model selection/fusion. We also plan to develop LSTM models for the emerging 5G mobile networks. For adoption in practical applications, we will integrate our realtime bandwidth prediction modules into our video streaming systems to replace their current prediction module using simple history-average. Finally, we will explore more data fusion methods, such as two-level fusion, more complicated estimation distribution, and other prediction models to further improve the prediction accuracy.

CRediT authorship contribution statement

Lifan Mei: Conceptualization, Methodology, Software, Investigation, Resources, Data curation, Writing - original draft, Visualization, Project administration. **Runchen Hu:** Validation, Software. **Houwei Cao:** Conceptualization, Methodology. **Yong Liu:** Conceptualization, Methodology, Writing - review & editing, Visualization, Supervision, Project administration. **Zifan Han:** Funding acquisition, Conceptualization. **Feng Li:** Funding acquisition, Conceptualization. **Jin Li:** Funding acquisition, Conceptualization, Project administration.

Acknowledgments

We thank all the anonymous reviewers for the valuable comments. This work was supported in part by the Gift Fund from Huawei Technologies.

Table 10
Long time overall fusion evaluation.

(a) Window Size = 50							
Data	Test_Ave	LSTM	Model switch		Model fusion		
		RMSE	RMSE	Gap_L	RMSE	Gap_SW	Gap_L
7A	6.4753	2.2920	2.3916	+4.35%	2.3015	−3.77%	+0.42%
B57	10.0092	2.3323	2.3766	+1.90%	2.3551	−0.90%	+0.98%
B62	2.5575	0.7366	0.7486	+1.63%	0.7433	−0.71%	+0.91%
7A_B62	4.1960	1.5981	1.6614	+3.96%	1.6006	−3.66%	+0.16%
7A_B57	8.3854	2.3344	2.4079	+3.15%	2.3501	−2.40%	+0.67%
B57_B62	5.9022	1.6752	1.7069	+1.89%	1.6829	−1.41%	+0.46%
(b) Window Size = 100							
Data	Test_Ave	LSTM	Model switch		Model fusion		
		RMSE	RMSE	Gap_L	RMSE	Gap_SW	Gap_L
7A	6.5600	2.2899	2.3325	+1.86%	2.2928	−1.70%	+0.13%
B57	9.9867	2.3134	2.3361	+0.98%	2.3374	+0.06%	+1.04%
B62	2.5623	0.7385	0.7462	+1.04%	0.7400	−0.83%	+0.21%
7A_B62	4.2097	1.5889	1.6157	+1.69%	1.5856	−1.87%	−0.21%
7A_B57	8.4441	2.3339	2.3711	+1.60%	2.3424	−1.21%	+0.37%
B57_B62	5.8557	1.6565	1.6720	+0.93%	1.6645	−0.45%	+0.48%
(c) Window Size = 200							
Data	Test_Ave	LSTM	Model switch		Model fusion		
		RMSE	RMSE	Gap_L	RMSE	Gap_SW	Gap_L
7A	6.6472	2.3196	2.3507	+1.34%	2.3350	−0.67%	+0.66%
B57	9.9589	2.3404	2.3578	+0.75%	2.3723	+0.61%	+1.36%
B62	2.5683	0.7439	0.7505	+0.89%	0.7438	−0.89%	−0.01%
7A_B62	4.2000	1.5899	1.6085	+1.17%	1.5942	−0.89%	+0.27%
7A_B57	8.5230	2.3479	2.3687	+0.89%	2.3632	−0.23%	+0.65%
B57_B62	5.7690	1.6587	1.6703	+0.70%	1.6725	+0.13%	+0.83%
(d) Window Size = 500							
Data	Test_Ave	LSTM	Model switch		Model fusion		
		RMSE	RMSE	Gap_L	RMSE	Gap_SW	Gap_L
7A	6.9432	2.4314	2.4394	+0.33%	2.4500	+0.43%	+0.77%
B57	9.8853	2.3891	2.4210	+1.33%	2.4344	+0.55%	+1.90%
B62	2.5797	0.7218	0.7232	+0.19%	0.7185	−0.65%	−0.46%
7A_B62	4.1600	1.5967	1.6044	+0.48%	1.6046	+0.01%	+0.49%
7A_B57	8.7715	2.3956	2.4025	+0.29%	2.4141	+0.48%	+0.77%
B57_B62	5.4994	1.6394	1.6604	+1.28%	1.6628	+0.14%	+1.42%
(e) Window Size = 1000							
Data	Test_Ave	LSTM	Model switch		Model fusion		
		RMSE	RMSE	Gap_L	RMSE	Gap_SW	Gap_L
7A	7.6957	2.4845	2.4825	−0.08%	2.5446	+2.50%	+2.42%
B57	10.1685	2.4819	2.4819	+0.00%	2.5267	+1.80%	+1.80%
B62	2.5891	0.7194	0.7196	+0.03%	0.7108	−1.22%	−1.19%
7A_B62	4.0690	1.4994	1.4984	−0.07%	1.5322	+2.26%	+2.19%
7A_B57	9.2563	2.4092	2.4251	+0.66%	2.4418	+0.69%	+1.35%
B57_B62	5.1321	1.5881	1.5930	+0.31%	1.6193	+1.65%	+1.97%

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] T. Bai, R. Vaze, R.W. Heath, Using random shape theory to model blockage in random cellular networks, in: 2012 International Conference on Signal Processing and Communications, SPCOM, IEEE, 2012, pp. 1–5.
[2] T. Bai, R. Vaze, R.W. Heath, Analysis of blockage effects on urban cellular networks, IEEE Trans. Wireless Commun. 13 (9) (2014) 5070–5083.
[3] G. Tian, Y. Liu, Towards agile and smooth video adaptation in dynamic HTTP streaming, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, ACM, 2012, pp. 109–120.
[4] X. Yin, A. Jindal, V. Sekar, B. Sinopoli, A control-theoretic approach for dynamic adaptive video streaming over HTTP, in: ACM SIGCOMM Computer Communication Review, Vol. 45, (4) ACM, 2015, pp. 325–338.

[5] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, B. Sinopoli, CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction, in: Proceedings of the 2016 ACM SIGCOMM Conference, ACM, 2016, pp. 272–285.
[6] E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu, J. Lyu, Real-time bandwidth prediction and rate adaptation for video calls over cellular networks, in: Proceedings of the 7th International Conference on Multimedia Systems, ACM, 2016, p. 12.
[7] K. Winstein, A. Sivaraman, H. Balakrishnan, Stochastic forecasts achieve high throughput and low delay over cellular networks, in: Presented As Part of the 10th {$USENIX$} Symposium on Networked Systems Design and Implementation ({$NSDI$} 13), 2013, pp. 459–471.
[8] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to Forget: Continual Prediction with LSTM, IET, 1999.
[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, 2014, arXiv preprint arXiv:1406.1078.
[10] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, et al., Deep neural networks for acoustic modeling in speech recognition, IEEE Signal Process. Mag. 29 (2012).
[11] http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/, HSDPA.
[12] Q. He, C. Dovrolis, M. Ammar, On the predictability of large transfer TCP throughput, Comput. Netw. 51 (14) (2007) 3959–3977.
[13] M. Mirza, J. Sommers, P. Barford, X. Zhu, A machine learning approach to TCP throughput prediction, in: ACM SIGMETRICS Performance Evaluation Review, Vol. 35, (1) ACM, 2007, pp. 97–108.
[14] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (3) (2004) 199–222.
[15] J. Jiang, V. Sekar, H. Zhang, Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive, IEEE/ACM Trans. Netw. (ToN) 22 (1) (2014) 326–340.
[16] C. Midoglu, L. Wimmer, A. Lutu, Ö. Alay, C. Griwodz, MONROE-Nettest: A configurable tool for dissecting speed measurements in mobile broadband networks, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, IEEE, 2018, pp. 342–347.
[17] A. Lutu, Y.R. Siwakoti, Ö. Alay, D. Baltrūnas, A. Elmokashfi, The good, the bad and the implications of profiling mobile broadband coverage, Comput. Netw. 107 (2016) 76–93.
[18] S. Sen, J. Yoon, J. Hare, J. Ormont, S. Banerjee, Can they hear me now? A case for a client-Assisted approach to monitoring wide-Area wireless networks, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, 2011, pp. 99–116.
[19] M. Dighriri, A.S.D. Alfoudi, G.M. Lee, T. Baker, R. Pereira, Comparison data traffic scheduling techniques for classifying QoS over 5G mobile networks, in: 2017 31st International Conference on Advanced Information Networking and Applications Workshops, WAINA, IEEE, 2017, pp. 492–497.
[20] M. Dighriri, G.M. Lee, T. Baker, Big data environment for smart healthcare applications over 5g mobile network, in: Applications of Big Data Analytics, Springer, 2018, pp. 1–29.
[21] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, W. Wei, LinkForecast: Cellular link bandwidth prediction in LTE networks, IEEE Trans. Mob. Comput. 17 (7) (2017) 1582–1594.
[22] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z.H. Abbas, S. Khaf, A deep learning approach for energy efficient computational offloading in mobile edge computing, IEEE Access 7 (2019) 149623–149633.
[23] A. Azzouni, G. Pujolle, NeuTM: A neural network-based framework for traffic matrix prediction in SDN, in: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2018, pp. 1–5.
[24] H. Mao, R. Netravali, M. Alizadeh, Neural adaptive video streaming with pensieve, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ACM, 2017, pp. 197–210.
[25] R.G. Brown, P.Y. Hwang, et al., Introduction to random signals and applied Kalman filtering, Vol. 3, Wiley New York, 1992.
[26] S. Haykin, Adaptive filter theory. Pearson education India, in: 27th Annual International Conference of the Engineering in Medicine and Biology Society, IEEE Press, 2008, pp. 1212–1215.
[27] M. Sundermeyer, R. Schlüter, H. Ney, LSTM neural networks for language modeling, in: Thirteenth Annual Conference of the International Speech Communication Association, 2012.
[28] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[29] M. Costa, A.L. Goldberger, C.-K. Peng, Multiscale entropy analysis of complex physiologic time series, Phys. Rev. Lett. 89 (6) (2002) 068102.

Lifan Mei completed his Bachelor's degree in Automation at Xi'An Jiaotong University (XJTU), Xi'An, China in 2015, and Master in Electrical and Computer Engineering at New York University (NYU) in 2017. He was a visiting student at UC Berkeley and National Chiao-Tung University (NCTU), Taiwan (RoC) in 2013. From 2017 to now, he is a Ph.D. Candidate at Electrical and Computer Engineering at New York University (NYU). His research focuses on Computer and Mobile Network, Bandwidth Prediction, Routing Optimization, Machine Learning etc.