On the Predictability of Fine-grained Cellular Network Throughput using Machine Learning Models

Omar Basit*†, Phuc Dinh*‡, Imran Khan*‡, Z. Jonny Kong*†, Y. Charlie Hu†, Dimitrios Koutsonikolas‡
Myungjin Lee§ Chaoyue Liu¶
†Purdue University, ‡Northeastern University, §Cisco Research, ¶University of California San Diego

2024 IEEE 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS) | 979-8-3503-6399-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/MASS62177.2024.00018

***Abstract*—Networking research has witnessed a renaissance from exploring the seemingly unlimited predictive power of machine learning (ML) models. One such promising direction is throughput prediction – accurately predicting the network bandwidth or achievable throughput of a client in real time using ML models can enable a wide variety of network applications to proactively adapt their behavior to the changing network dynamics to potentially achieve significantly improved QoE. Motivated by the key role of newer generations of cellular networks in supporting the new generation of latency-critical applications such as AR/MR, in this work, we focus on accurate throughput prediction in cellular networks at fine time-scales, e.g., in the order of 100 ms. Through a 4-day, 1000+ km driving trip, we collect a dataset of fine-grained throughput measurements under driving across all three major US operators. Using the collected dataset, we conduct the first feasibility study of predicting fine-grained application throughput in real-world cellular networks with mixed LTE/5G technologies. Our analysis shows that popular ML models previously claimed to predict well for various wireless networks scenarios (e.g., WiFi or single-technology network such as LTE only) do not predict well under app-centric metrics such as ARE95 and PARE10. Further, we uncover the root cause for the poor prediction accuracy of ML models as the inherent conflicting sample sequences in the fine-grained cellular network throughput data.**

***Index Terms*—Machine learning, time series, cellular network, throughput prediction, learnability**

I. Introduction

In the past years, machine learning (ML) has been widely applied in networking research to assist in optimizing the design of all layers of the network stack [1], from PHY-layer protocol design via predicting physical channel conditions [2]–[6], to better congestion control design [7]–[11], all the way to application-layer adaptation such as video streaming systems [12]–[14].

One central idea in the general approach of exploiting the synergy between ML and networking is throughput prediction, i.e., predicting the network bandwidth or achievable throughput of a client in real time. Accurate prediction of network throughput of a client device allows a wide variety of network applications running on the device including latency-sensitive applications such as video streaming [12], [14], [15], video analytics systems [16], and latency-critical applications such as AR/VR/MR [17]–[20], video conferencing [21], and CAVs [22] to proactively adapt their behavior to the changing network dynamics to achieve improved QoE.

Motivated by the critical role of throughput prediction in enabling the new-generation of adaptive network applications, and the key role of newer generations of cellular networks in supporting ubiquitous, high performance network access to these applications, in this work, we focus on throughput prediction in cellular networks at fine time-scales, e.g., in the order of 100 ms. The class of latency-critical applications such as AR/MR, CAV, and video conferencing, critically relies on such fine-grained throughput prediction of mobile clients.

To collect real-world throughput achieved by mobile devices, we designed a testbed consisting of several Android phones, Accuver XCAL Solo devices [23], laptops, power sources to power all of the devices, and cloud/edge servers, and conducted throughput measurements in a cross-country driving trip from Boston to Atlanta over all three major mobile networks by collecting application throughput as well a host of MAC and PHY-layer KPIs and signaling messages. Our dataset constitutes the first fine-grained, bidirectional (uplink and downlink) throughput measurement of today's cellular deployments, which consist of a mix of LTE and 5G technologies, under high mobility (driving).

We started our fine-grained cellular network throughput prediction study with ML models widely used in previous studies, including MLP, MVTS, and GDBT, and their typical architectures, e.g., small models with fewer layers, which were claimed to be sufficient to achieve high accuracy. Our analysis, however, shows that, although these models can achieve low RMSE values, e.g., 7.57 and 7.48 Mbps for GDBT and MVTS for Verizon uplink, their accuracy remains fairly low in terms of application-centric accuracy metrics, e.g., ARE95 and PARE10.

To understand the potential and limit of ML models in predicting fine-grained cellular network throughput, we utilize recent ML theories developed based on over-parameterized ML models, where the number of model parameters sufficiently exceeds the number of training samples, which suggest that over-parameterized ML models are guaranteed to achieve close-to-zero training loss given enough training time [24], while not suffering from the classical overfitting issue (i.e., the test loss is as low as the model can achieve) [25], [26]. These theories suggest that over-parameterized models can be

*Equal contribution.

2155-6814/24/$31.00 ©2024 IEEE
DOI 10.1109/MASS62177.2024.00018

Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.

an effective tool to test the learnability of the cellular network throughput at fine time-scales by ML models.

Our experiments with over-parameterized MLPs on predicting fine-grained cellular network throughput show that they suffer from (1) very slow convergence of training loss, and (2) high validation/test loss/error, both of which indicate cellular network throughput is not learnable or predictable at fine time-scales.

Training over over-parameterized MLPs is theoretically guaranteed to converge at a rate of $O(\exp(-t/\kappa))$ with κ being the so-called condition number. The slow convergence suggests a large condition number κ, which is often caused by conflicting data in the dataset, i.e., similar sequence of samples (inputs to the ML model) are followed by significantly different ground-truth labels. Hence, we hypothesize that learnability challenges of fine-grained cellular network throughput prediction originates from conflicting samples in the dataset. Our careful analysis of the dataset confirms that the throughput dataset indeed contains many occurrences of such conflicting sequences of samples.

We summarize our contributions as follows. (1) We conduct the first study of the feasibility of fine-grained cellular network throughput prediction, e.g., at the 100 ms granularity. (2) We found popularly used ML models previously claimed to predict well for various wireless networks scenarios (e.g., WiFi or single-technology cellular network such as LTE or mmWave only) do not predict well under app-oriented metrics such as ARE95 and PARE10. (3) We further uncover the root cause for the poor prediction accuracy of ML models as the inherent conflicting sample sequences in the fine-grained cellular network throughput data. Our dataset is publicly available [27].

II. Related Work

Fine-grained cellular throughput prediction. With the growing demand for real-time and latency-sensitive applications such as video conferencing and edge-assisted augmented reality (AR), various works have explored cellular throughput prediction at fine granularities, typically every few hundred ms or less. While these works often focus on predicting throughput under challenging driving conditions, they have several limitations. For example, [28] employs a transformer-based ML model to predict the transport block size (TBS) of the next 100 ms for both 5G and LTE. However, it remains unclear whether accurate TBS prediction translates to accurate app throughput prediction. PERCEIVE [29] demonstrates that LSTMs can accurately predict the next 100 ms throughput, but focuses only on LTE, which is known to have different performance characteristics from 5G [30], [31], in particular, a smaller throughput range and lower variability. Additionally, it only addresses the uplink direction and thus does not benefit applications with heavy downlink traffic such as video conferencing.

Throughput prediction under driving. Several other works on LTE and/or 5G throughput prediction also involve driving scenarios but focus on coarse-grained predictions. These works employ various models including linear regression [32], random forests [33], [34], support vector machines (SVM) [35], LSTMs and GDBTs [36], and transformers [37]. These works are tasked to predict the averaged throughput over a window of several seconds, where the short-term fluctuations are averaged out. Consequently, their use cases are limited to non-real-time applications such as video streaming.

Cellular network throughput explainability. Several studies have examined the predictability and explainability of cellular network throughput. For example, [38], [39] use Gramian Angular Field (GAF), an imaging technique, to separate the prediction errors due to model design versus inherent dataset unpredicability. However, they analyze the throughput logged at a coarse granularity of every 3 minutes, where the throughput dynamics are affected by different factors than at 100 ms.

Fine-grained cellular throughput datasets. While there have been several studies that collected cellular throughput datasets [29], [31], [35], [36], [40]–[42], most of these datasets include throughput samples at coarse granularities, e.g., once every several hundred of ms to several seconds. The only work that collected fine-grained throughput data [29] was limited to LTE and did not make the dataset public. We hope that our collected dataset, which logs throughput samples every 100 ms and records various lower-level network KPIs, will make a valuable addition to the public cellular network datasets available to the research community.

III. Methodology

Drive Tests. We drove from Boston to Atlanta covering a distance of 1000+ km over a 4-day period in 2023 (August 5 to 8). Our measurements were performed while driving on highways, through sub-urban areas, or inside cities.

Testbed and Data Logging. We built a testbed to collect the application layer throughput and various mobile network KPIs. The testbed consists of several Android phones, Accuver XCAL Solo devices [23], laptops, power sources to power all of the devices, and cloud/edge servers. The Android smartphones are 3 Samsung Galaxy S21/SM-G998U (unrooted), 5G capable phones, with each device having an unlimited data plan from the 3 major U.S. network carriers: AT&T, T-Mobile, and Verizon. The XCAL Solo is a handheld commercial tool that connects to an Android smartphone via USB-C port to the Android debugging interface to collect MAC and PHY-layer KPIs and signaling messages. To enable throughput measurements, we deployed one AWS Cloud server located in Northern Virginia for all three operators. Additionally, for measurements in city areas with Verizon, we deployed AWS Wavelength servers in all the major cities along the route: Boston, New York, Charlotte, and Atlanta. Wavelength servers are located inside Verizon's network in select cities and are specially designed for edge computing.

To measure the application layer throughput, we used the popular tool nuttcp [43] because of its fine-grained throughput reporting between consecutive samples. We used a pre-compiled version of nuttcp to run on Android smartphones (serving as the nuttcp-client) and the standard nuttcp, available

Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.

TABLE I: Number of 2-minute tests per operator & direction.

Operator	Uplink / Downlink
Verizon	241 / 249
T-Mobile	128 / 121
AT&T	251 / 249

TABLE II: List of 4G & 5G network KPIs logged through XCAL with their logging granularity.

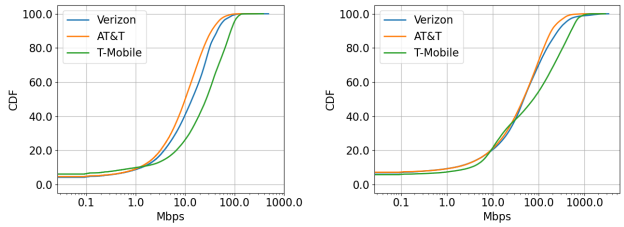
Network KPI	Logging Interval (ms) 4G / 5G
MAC Throughput	1000 / 100
MCS	20 / 0.010
TBS	10 / 0.010
RB	20 / 0.010
BLER	20 / 100
RSRP	10 / 160
RSRQ	10 / 160

through Linux packages on cloud/edge servers (serving as the nuttcp-server). We configured nuttcp for TCP traffic with a receive socket buffer size (receive window) of 32 MB and a buffer read/write size of 640 bytes (set low due to the short reporting interval). Each throughput test ran for 2 minutes, with the logging interval set to 10 ms. On a single smartphone, we alternated between a 2-minute uplink (UL) test and a 2-minute downlink (DL) test. Note that two consecutive DL (or UL) tests may not be contiguous, resulting in variable gaps between subsequent DL (or UL) tests. Table I summarizes the 2-minute tests we collected. Figures 1a, 1b show the application throughput CDF in the uplink and downlink direction, respectively, for each of the three operators. We observe that the coexistence of 5G and LTE technologies combined with large channel fluctuations while driving at different speeds results in very large throughput ranges – from a few Mbps up to 500 Mbps in the uplink direction and up to 3 Gbps in the downlink direction.

To collect lower-level network KPIs, we used the XCAL Solo devices connected to each of the three smartphones. Logging for all KPIs was set to the finest supported logging interval for each KPI. This led to each KPI being logged at different intervals, from 10 ms for 5G TBS to 100 ms for 5G MAC throughput. Logging intervals may also vary between 5G and 4G and can go as high as 1s (4G MAC throughput). Table II lists the network KPIs that XCAL logged during the nuttcp throughput tests and their logging intervals for 4G and 5G. Finally, to deal with the mismatch of application throughput logging intervals (10ms) and variable network KPI loggings, we use forward-filling, i.e., we pick the lowest logging interval feature (application throughput) and forward-fill the other features w.r.t to it.

Throughput prediction models. Recent work has shown that ML models are suitable for network throughput prediction at 1 s [37] and even 100 ms time-scale [29]. Therefore, we experimented with all three ML models that have been used in recent work, and additionally two statistical approaches, as a control, to predict application throughput at the 100 ms granularity:

- *MLP3:* The Multilayer Perceptron (MLP) is a classic neural network that we configure with 3 hidden layers with each layer having 12 neurons. By default, each hidden layer also contains batch normalization and dropout, the activation function used is ReLU, and the optimizer is Adam. We use MLPs because they have been found effective in throughput prediction, e.g., in [44].
- *GDBT:* Gradient Boosting Decision Trees (GDBT) is from the Gradient Boosting class of ML algorithms. It has been shown to predict 5G throughput accurately at a granularity of 1 second in [36].
- *MVTS:* Multivariate Time Series (MVTS) [45] is a transformer-based encoder model for multivariate time series predictions. It was used in [28] to predict uplink throughput in 5G/4G dual connectivity scenarios indirectly by first predicting 4G and 5G TBS.
- *EWMA:* Exponentially Weighted Moving Average (EWMA) is a well-known prediction technique for time series. We include EWMA as a simple statistical approach with an $\alpha = 0.5$ for comparison with sophisticated ML models.
- *LV:* Last-Value (LV) is a very basic prediction approach that simply uses the ground truth observed at sample $i-1$ as the prediction for sample i. LV will perform well if the time series have constant values or change slowly and smoothly. We also use this approach as a control to compare with the sophisticated ML models.



(a) Uplink throughput. (b) Downlink throughput.

Fig. 1: CDFs of application throughput at 100 ms logging interval for all three cellular operators.

Data Preparation. The dataset is divided into three parts; training, validation, and test in a ratio of 7:1.5:1.5. The split is done for each 2-minute test, i.e., the first 70% of the 2 minutes is taken as the training set, the next 15% as the validation set and the rest as the test set. Splitting the dataset this way provides more locality among training and testing sets compared to splitting at the 2-minute test granularity. For each feature selected as an input, we use a history window of N (default to 8) samples as input as in [44] to the ML model, i.e., for throughput prediction at time sample i, we give samples $i-8$ to $i-1$ of all the input features to the model. Thus, when giving a history of 8 samples, the starting 7 samples will be dropped for each 2-minute dataset.

For ML models, e.g., MLP3 and MVTS, all throughput values, which range between 0 to 3000 Mbps, are applied with log scaling, i.e., $x' = ln(x+1)$ before fed as input to the ML models. Log scaling was applied because the raw throughput values are clustered around low values, i.e., closer to 0, and with log scaling the scaled values become more uniformly spread (in the scaled range [0, 7]) and this allows the ML

Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.

models to be trained more effectively.

Since we are predicting app throughput at 100 ms granularity, the input values for every feature are converted to 100 ms granularity, using either simple averaging (e.g., for throughput which are logged at 10 ms granularity), or using the last reported value for the specified interval (e.g., 5G RSRP values logged at 10 ms granularity, but it would not make sense to use averaging for RSRP values, which are in dBm).

Finally, we observed "black-out" periods in our throughput dataset, where the throughput sequence consists of back-to-back zero values, and the first non-zero throughout samples afterwards almost always differ from each other. Since these zero throughput are special samples and no learning schemes are expected to be able to predict, we removed such zero throughput samples so that any window of samples (e.g., 8) contains no more than 7 zero samples.

Accuracy Metrics. We evaluate the performance of the prediction models based on three commonly used metrics:

- *RMSE:* Root Mean Square Error between the predicted throughput value and the ground truth throughput value. The lower the RMSE value the more accurate the predictor.
- *ARE95:* Absolute Relative Error at 95th Percentile. We would like to achieve a low ARE95 value.
- *PARE10:* Percentage of Absolute Relative Errors below 10%. We would like to achieve a high PARE10 value.

IV. Prediction Results

For the statistical models, EWMA and LV, the input can only be the application throughput. For the ML models, we experimented with 3 types of inputs: (1) application throughput, (2) network KPIs, and (3) both app throughput and network KPIs. To choose which network KPIs to select, we first calculated the Pearson correlation coefficient of each 4G/5G KPI with the app throughput and chose the KPI with the highest correlation coefficient. A similar approach was taken in [28]. Table III shows that the KPI with the highest correlation with app throughput is MAC throughput, and thus we selected 5G/4G MAC throughput as input network feature.

Next, we show prediction results using the three types of inputs separately.

A. Application throughput as input (Verizon)

We first look at using app throughput as the only input feature with the goal of examining if there is enough pattern in just the history of app throughput to predict the app throughput for the next time sample. The overall prediction results for both uplink and downlink for Verizon are shown in Table IV (second column), and the CDF of prediction errors are shown in Figures 2(a)(d). We make the following observations. (1) For uplink, GDBT and MVTS achieve the highest prediction accuracy, with RMSE values of 7.57 and 7.48, respectively. (2) Somewhat surprisingly, both EWMA and Last-value, which are simple statistical models, achieve similar (only slightly lower) prediction accuracy compared to the ML models, with RMSE values of 8.97 and 9.16, respectively. (3) However, even a low RMSE value in the order of 10 Mbps can have a very high relative error when the ground-truth value is low. Indeed, Table IV shows that the accuracy of ML models is fairly low in terms of ARE95 and PARE10. In particular, GDBT and MVTS only achieve ARE95 values of 81.8% and 81.3% and PARE10 of 40.1% and 40.7%, respectively, i.e., about 60% of the predictions have relative error larger than 10%. (4) Finally, the prediction accuracy in terms of ARE95 and PARE10 for downlink throughput across the models is similar to uplink throughput. The larger RMSE for the downlink results is expected as the range of throughput for downlink is an order of magnitude larger than for uplink (see Figures 1a, 1b).

B. MAC throughput as input (Verizon)

Next, we look at using MAC throughput as an input feature to see if using network KPIs as features can be beneficial in app throughput prediction. As stated previously, we chose MAC throughput because offline analysis shows it has the highest correlation with app throughput. The results in Table IV (third column) and Figures 2(b), 2(e) show that the prediction accuracy of various models is very similar to that when using app throughput as input. Hence, network features do not appear to provide more useful information than app throughput when used as input to prediction models.

C. App throughput and MAC throughout as input (Verizon)

Finally, we look into whether using both app throughput and MAC throughput helps the models learn and predict better than using either input feature alone. The results, also shown in Table IV (last column) and Figures 2(c), 2(f) are again very similar to the previous two scenarios, suggesting that combining app throughput with network KPIs does not appear to help, and thus using just app throughput as input to the prediction models appears sufficient.

D. Throughput prediction across operators

The above findings about poor throughput prediction accuracy for the Verizon dataset largely hold true for the other operators, as shown in Tables V, VI and Figures 3, 4 although the specific numbers differ. Focusing on using app throughput as input, we make the following observations. First, the uplink throughput prediction RMSEs for GDBT and MVTS are lower for AT&T, at 5.17 and 5.12 Mbps but higher for T-Mobile, at 12.3 and 12.2, respectively. The same trend can be seen for downlink throughput predictions. This can be explained by Figures 1a, 1b, which show that the range and average throughput values for the three operators are different; in particular, the average throughput is lower for AT&T uplink, at 14.7 Mbps, and higher for T-Mobile uplink, at 35.5 Mbps.

Second, the ARE95 values for the two models are lower for AT&T (71.9% and 70.8%) and similar for T-Mobile (82.0% and 81.4%) compared to Verizon (81.8% and 81.3%).

Finally, the PARE10 values for the two predictors are lower (less accurate) for AT&T (37.6% and 37.8%) but higher (more accurate) for T-Mobile (44.5% and 46.%), compared to Verizon (40.1% and 40.7%).

Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.

TABLE III: Pearson Correlations Coefficients of network KPIs with app throughput.

	MAC 4G / 5G	TBS 4G / 5G	MCS 4G / 5G	RB 4G / 5G	BLER 4G / 5G	RSRQ 4G / 5G	RSRP 4G / 5G
Uplink							
Verizon	0.566 / 0.466	0.493 / 0.403	0.403 / 0.377	0.419 / 0.252	-0.059 / -0.057	0.209 / 0.121	0.404 / 0.295
AT&T	0.567 / 0.531	0.502 / 0.419	0.399 / 0.497	0.332 / 0.195	-0.039 / -0.130	0.162 / 0.227	0.347 / 0.423
T-Mobile	0.480 / 0.617	0.410 / 0.582	0.373 / 0.513	0.327 / 0.355	-0.166 / -0.230	0.185 / 0.207	0.422 / 0.496
Downlink							
Verizon	0.182 / 0.739	0.096 / 0.185	0.0006 / 0.283	0.043 / 0.509	-0.062 / -0.080	0.065 / 0.179	0.220 / 0.127
AT&T	0.363 / 0.558	0.229 / 0.496	0.191 / 0.285	0.196 / 0.444	-0.011 / -0.084	0.045 / 0.237	0.194 / 0.133
T-Mobile	0.195 / 0.626	0.173 / 0.521	0.165 / 0.249	0.159 / 0.471	0.010 / 0.030	0.036 / 0.149	0.252 / 0.252

TABLE IV: App throughput prediction at 100-ms results using different input features, Verizon.

Model	Using App Throughput			Using MAC Throughput			Using MAC & App Throughput		
	RMSE (Mbps)	ARE95 (%)	PARE10 (%)	RMSE (Mbps)	ARE95 (%)	PARE10 (%)	RMSE (Mbps)	ARE95 (%)	PARE10 (%)
Uplink									
MLP3	11.62	211	25.6	12.93	106.5	17.7	11.5	89.2	25.1
GDBT	7.57	81.8	40.1	7.85	88.7	38.4	7.48	84.1	40.6
MVTS	7.48	81.3	40.7	7.77	81.2	41.1	7.40	84.1	40.8
EWMA	8.97	97.9	29.9	8.97	97.9	29.9	8.97	97.9	29.9
LV	9.16	93.7	37.7	9.16	93.7	37.7	9.16	93.7	37.7
Downlink									
MLP3	116.4	153.6	15.2	134.3	98.6	4.8	115.5	99.2	8.2
GDBT	62.34	123.1	25.9	68.3	94.9	16.9	63.81	115.2	26.4
MVTS	69.8	129.3	29.2	63.9	107.6	28.2	65.1	107.0	26.7
EWMA	73.91	174.5	19.01	73.91	174.5	19.01	73.91	174.5	19.01
LV	73.3	144.9	28.1	73.3	144.9	28.1	73.3	144.9	28.1

TABLE V: App throughput prediction at 100-ms results using different input features, AT&T.

Model	Using App Throughput			Using MAC Throughput			Using MAC & App Throughput		
	RMSE (Mbps)	ARE95 (%)	PARE10 (%)	RMSE (Mbps)	ARE95 (%)	PARE10 (%)	RMSE (Mbps)	ARE95 (%)	PARE10 (%)
Uplink									
MLP3	9.38	145.1	26.2	10.1	68.1	13.0	11.25	79.1	21.2
GDBT	5.17	71.9	37.6	4.85	64.8	36.3	5.0	70.7	38.6
MVTS	5.12	70.8	37.8	5.37	66.3	38.3	4.89	68.9	39.5
EWMA	6.19	85.0	34.2	6.19	85.0	34.2	6.19	85.0	34.2
LV	6.16	93.1	29.3	6.16	93.1	29.3	6.16	93.1	29.3
Downlink									
MLP3	44.1	110.3	19.4	54.73	101.2	89.5	43.7	114.3	20.0
GDBT	33.2	101.2	28.1	33.8	91.4	19.5	33.9	99.4	28.5
MVTS	32.7	106.0	29.1	34.2	99.3	28.3	33.9	99.1	28.5
EWMA	40.4	144.3	20.7	40.4	144.3	20.7	40.4	144.3	20.7
LV	39.1	139.3	27.2	39.1	139.3	27.2	39.1	139.3	27.2

V. Why Can't ML Models Predict Cellular Network Throughput Accurately?

Our results suggest that the ML models widely used in previous throughput prediction studies [29], [32]–[34], [36], [37] cannot predict the app throughput well at fine granularity. To verify this fundamental limitation for the predictability of cellular network throughput, we utilize over-parameterized ML models (e.g., sufficiently wide MLPs) and recent ML theories developed based on them [24]–[26].

Over-parameterized ML models, where the number of model parameters largely exceeds the number of training samples, are recently found to be guaranteed to achieve close-to-zero training loss given enough training time [24], while not suffering from the classical overfitting issue (i.e., the test loss as low as the model can achieve) [25], [26]. Hence, over-parameterized model can be an effective tool to test the learnability of the cellular network throughput by ML models.

In this section, we first show that the over-parameterized MLPs suffer from the following training issues: very slow convergence of training loss and high validation/test loss/error, both of which indicate cellular network throughput is not learnable or predictable at fine granularity (e.g., 100 ms). We further show that both limitations originate from the same structure of the cellular network throughput data. For clarity, we focus on predicting Verizon uplink throughput only (at 100 ms granularity as before).

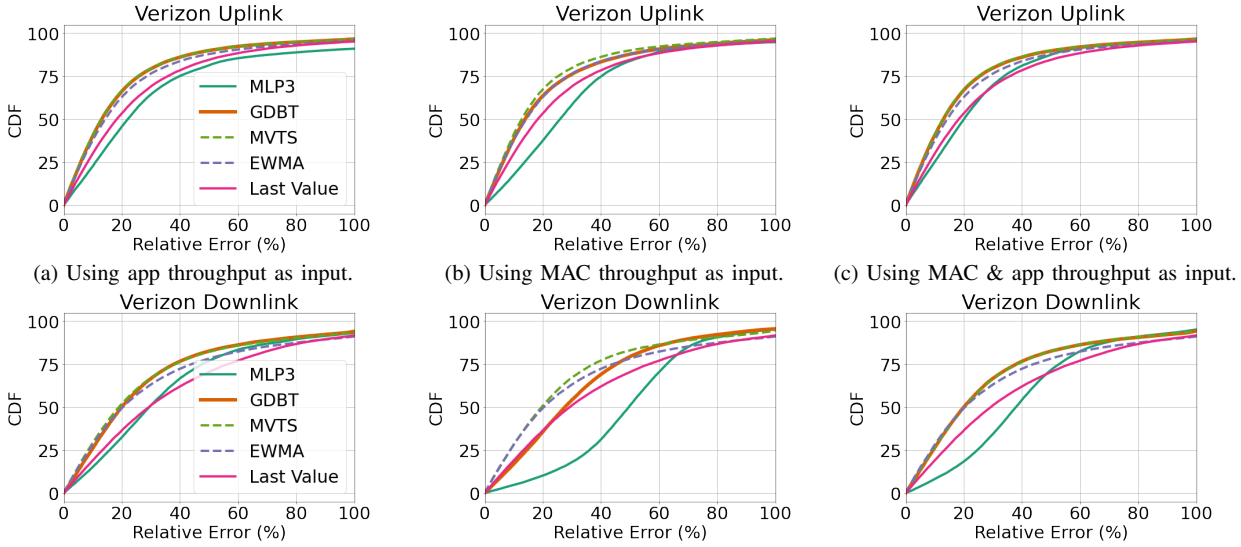
A. Experimental setting

Model architecture: We use a 5-layer ReLU-activated MLP, with 1000 neurons in each hidden layer. This MLP has more than 4 million trainable parameters, which largely exceeds the training data size of approximately 270K, hence over-parameterized. All the model parameters are randomly i.i.d. initialized following normal distribution. The loss function used is MSE loss, the same as in the prior MLP3 model. The model is trained with mini-batch variant of stochastic gradient descent (SGD) with a batch size $32,784$. The learning rate is tuned using grid search, and is selected to be the largest under which the training does not diverge. We do not implement any explicit regularizer (for example, dropout, weight decay, etc), as it has been found that over-parameterized ML models enjoy

Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.

TABLE VI: App throughput prediction at 100-ms results using different input features, T-Mobile.

Model	Using App Throughput RMSE (Mbps)	ARE95 (%)	PARE10 (%)	Using MAC Throughput RMSE (Mbps)	ARE95 (%)	PARE10 (%)	Using MAC & App Throughput RMSE (Mbps)	ARE95 (%)	PARE10 (%)
Uplink									
MLP3	19.64	175.5	17.0	23.5	89.1	15.7	21.0	92.4	16.5
GDBT	12.3	82.0	44.5	11.5	74.2	37.4	12.1	80.2	44.3
MVTS	12.22	81.4	46.1	12.36	78.4	44.1	12.35	78.54	43.5
EWMA	14.8	95.7	33.2	14.8	95.7	33.2	14.8	95.7	33.2
LV	13.9	100.0	42.9	13.9	100.0	42.9	13.9	100.0	42.9
Downlink									
MLP3	193.2	103.5	13.3	212.35	103.3	7.8	194.9	97.1	105.3
GDBT	98.6	110.1	27.2	111.5	111.9	21.1	101.4	108.1	26.7
MVTS	100.7	109.3	26.2	114.	99.7	21.1	102.0	115.8	29.7
EWMA	116.4	150.1	21.3	116.4	150.1	21.3	116.4	150.1	21.3
LV	114.1	150.6	29.3	114.1	150.6	29.3	114.1	150.6	29.3



(a) Using app throughput as input. (b) Using MAC throughput as input. (c) Using MAC & app throughput as input.

(d) Using app throughput as input. (e) Using MAC throughput as input. (f) Using MAC & app throughput as input.

Fig. 2: CDF of relative prediction error for Verizon.

"implicit regularization" [25] and explicit regularizers do not necessarily bring in benefits but may slow down training.

B. Unlearnability and Unpredictability

We run the over-parameterized MLP with normalized inputs for $20K$ epochs. In Figure 5a, we observe a decreasing trend in the training loss, and the minima achieved is 0.139. The loss decreases at a rate of about 0.0007 per $1K$ epochs.

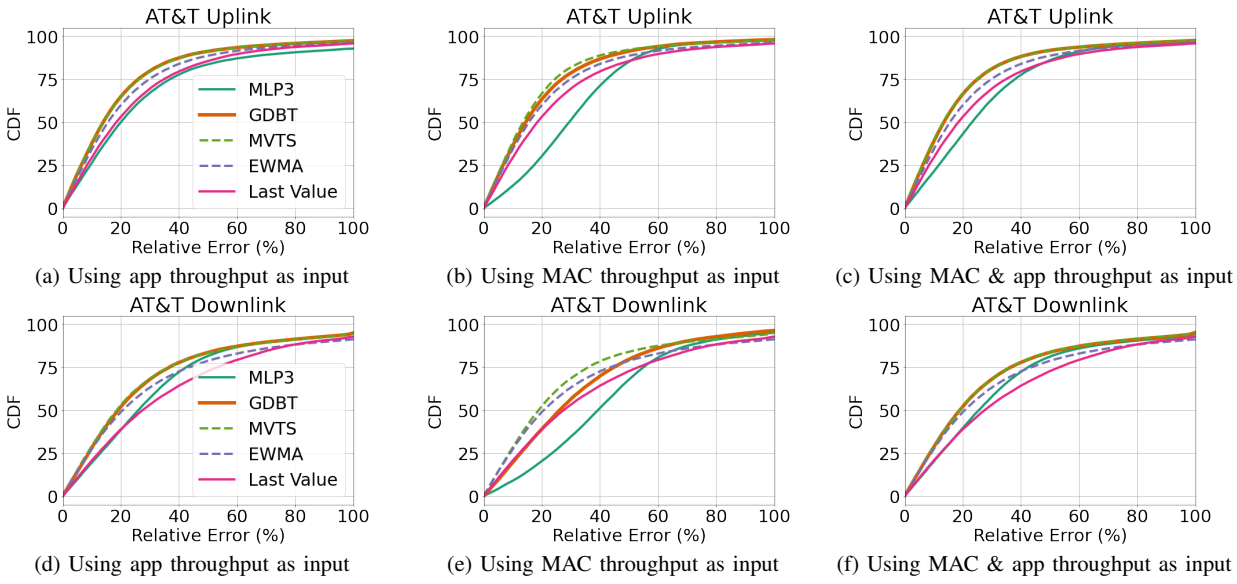
Recent deep learning theory [24], [46] guarantees that the training loss of this over-parameterized MLP trained with SGD asymptotically converges to zero at a rate of $O(\exp(-t/\kappa))$, where κ is the so-called condition number that is mainly determined by the training dataset. Figure 5a shows that the training loss consistently decreases but very slowly, which indicates that the associated condition number κ is extremely large and the data is not learnable at a long but finite time.

In addition, we note that in Figure 5b, the validation loss stopped improving even if the training loss is still slowly improving. The minimum validation losses (unsmoothed) achieved for log normalization is 0.151 at epoch $14K$. We can see that the validation loss minimum was achieved quite early and then the loss starts to slowly increase. Hence, even if much longer training time is provided so that the training loss becomes close to zero, the validation/test loss would still remain high, which means that the samples in the validation/test set remains largely unpredictable no matter how much the training set is fit.

To put the training and validations losses in perspective with the test accuracies, we achieved RMSE, ARE95, and PARE10 of 7.46, 79.9 and 40.6. Comparing these with the results in Table IV for Uplink using app throughput, we see that these accuracies are slightly better than the best we achieved using smaller models, e.g., MVTS.

We hypothesize that these issues are caused by the structure of the dataset, where there exist many similar inputs that have significantly different ground-truth labels. In practice, large condition numbers are often caused by very similar inputs, because their similarity makes these inputs hard to distinguish. Theoretically, the more similar the samples are, the larger the condition number is, which slows down the training at a rate of $O(\exp(-t/\kappa))$. Intuitively, when many similar inputs with significantly different labels exist, the ML model function has

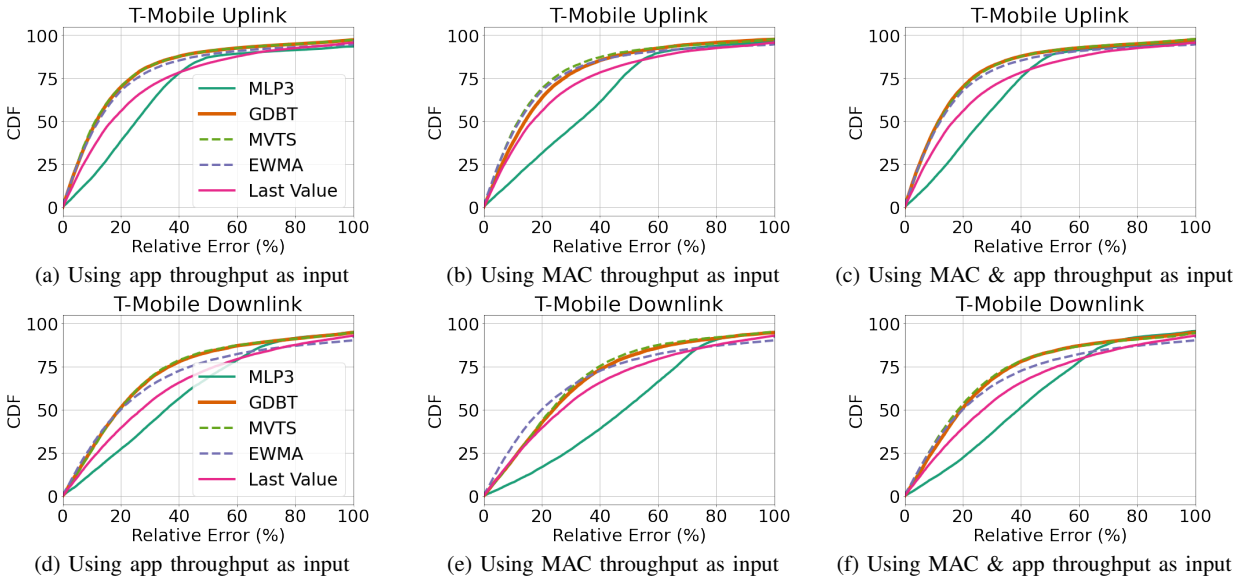
Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.



(a) Using app throughput as input (b) Using MAC throughput as input (c) Using MAC & app throughput as input

(d) Using app throughput as input (e) Using MAC throughput as input (f) Using MAC & app throughput as input

Fig. 3: CDF of relative prediction error for AT&T.



(a) Using app throughput as input (b) Using MAC throughput as input (c) Using MAC & app throughput as input

(d) Using app throughput as input (e) Using MAC throughput as input (f) Using MAC & app throughput as input

Fig. 4: CDF of relative prediction error for T-Mobile.

to be very "spiky" and very well fine-tuned to fit the training data well, which makes the training exceptionally hard and slow. This is illustrated in Figure 6.

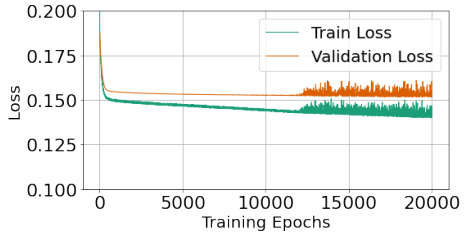
Moreover, the existence of such similar inputs indicates that, within those particular input regions, the ground-truth labels are intrinsically quite uncertain and have large variance. In this sense, for any unseen data (i.e., from the validation or test sets), the ML model generically could not generate accurate and precise predictions.

C. Verification of the hypothesis

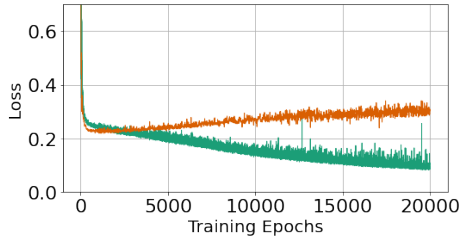
To validate this hypothesis, we analyze the "badness" of the dataset, measured as the percentage of *conflicting* sample pairs, defined as a pair of samples sharing similar throughput history, i.e., with the angle between the two input vectors less than a predefined threshold α, but having current throughput values (labels) that differ significantly, i.e., with absolute difference larger than a threshold β. We use the angle between input vectors as the measure of similarity of samples, because [47] found that, due to the homogeneity of the ReLU activation function, the condition number of a ReLU MLP is directly related to the angle separation, instead of other metrics, such as Euclidean distance, between data samples.

In Figure 7, we take all pairs of input vectors in our Verizon uplink dataset and calculate the angle between the pairs. We observe that most of the pairs have small angle separation, with the peak at 3°. This indicates that a large portion of

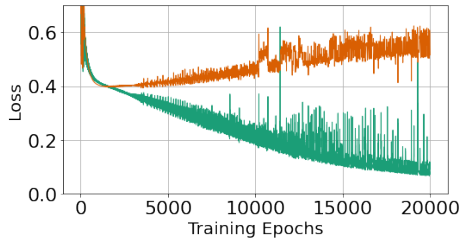
Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.



(a) For complete dataset.



(b) For dataset filtering such that no two inputs have angle < 1 °.



(c) For dataset filtering such that no two inputs have angle < 3 °.

Fig. 5: Training and validation loss of over-parameterized MLP models for Verizon uplink.

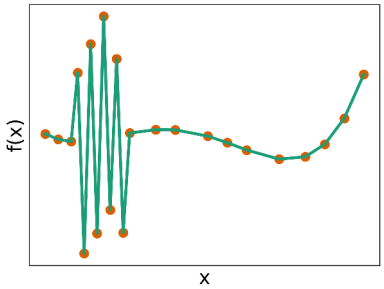


Fig. 6: Illustration of the necessity of a very "spiky" function/model to fit the dataset that contains similar inputs x but very different labels.

input vectors point to similar directions, resulting in a very high condition number and thus making fine-tuning of ML models necessary to fit the training data, hence slowing down the learning of the model, as seen in Figure 5a.

Figure 8 further plots the statistics of the badness of the uplink throughput trace for Verizon, for α = 1° and 3° under varying β threshold values. For example, for $\alpha = 1°$, 58% of all input vector pairs are similar, e.g., their angles are less than 1°, out of which, 81% have very different labels under $\beta = 1.0$, i.e., their labels differ more than 1.0, which account for

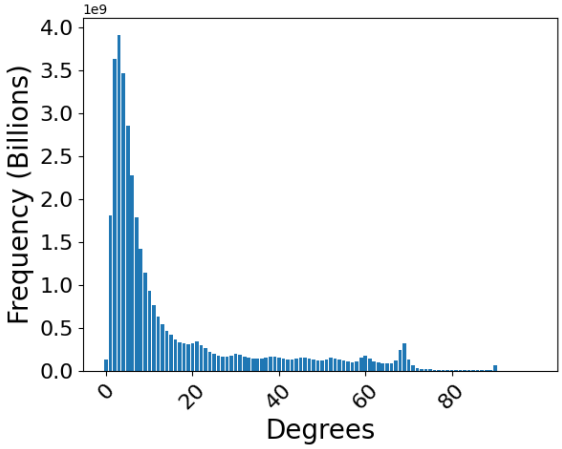


Fig. 7: Histogram of the angles between all pairs of input vectors, Verizon uplink.

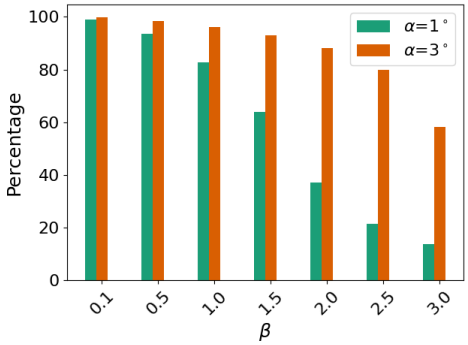


Fig. 8: Statistics of conflicting sample pairs. The vertical axis represents the percentage of similar pairs with conflicting labels, out of the total similar pairs for their respective α values. For $\alpha = 1°$ and 3°, similar vectors account for 58% and 85% of the whole dataset, respectively.

47% of all input vector pairs in the data set. Note that the label range (after the log normalization) is $(0, 7)$. A label difference of $\beta > 1.0$ is considered quite large. These badness statistics clearly show that there is a large portion of conflicting samples within the cellular network throughput dataset, confirming our hypothesis.

A few examples of these similar pairs can be seen in Table VII. The table also shows the denormalized input vectors and labels in Mbps as a more easily understandable reference. We observe that, for each pair, one label is more in line with the trend of the vector but the other label is a big shift from the trend. For example, for the first pair, where both vectors consist mostly of 0 values, the first (denormalized) label is 0 Mbps but the second label is 466.5 Mbps.

We further verify our hypothesis by removing these "conflicting samples" from the complete dataset (including both training and validation sets) and retrain the MLP model. Specifically, we filter out all the similar input vectors, by incrementally removing one vector for each similar input vector pairs, so that in the remaining dataset no pair of input

Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.

TABLE VII: Samples of similar vectors, showing the input pairs that have very low angle between them but have a big difference in label.

1^{st} Vector, Normalized 1^{st} Vector, Denormalized	2^{nd} Vector, Normalized 2^{nd} Vector, Denormalized	Angle	1^{st} Label, Nmlz. 1^{st} Label, DeNmlz.	2^{nd} Label, Nmlz. 2^{nd} Label, DeNmlz.
(0, 0.4, 0, 0, 0, 0, 0, 0) (0, 0.5, 0, 0, 0, 0, 0, 0)	(0, 1.1, 0, 0, 0, 0, 0, 0) (0, 2.3, 0, 0, 0, 0, 0, 0)	0	0 0	6.1 466.5
(1.2, 1.3, 0.9, 0, 0, 0, 0, 0) (2.6, 2.7, 1.5, 0, 0, 0, 0, 0)	(3.9, 4.0, 2.8, 0, 0, 0, 0, 0) (51.7, 53.6, 16.9, 0, 0, 0, 0, 0)	0.7	0 0	5.7 326.2
(4.5, 4.6, 4.6, 4.5, 4.6, 4.4, 0, 0) (95.6, 102.0, 106.9, 94.0, 101.0, 88.2, 0, 0)	(4.0, 4.0, 4.1, 4.0, 4.1, 3.9, 0, 09) (58.0, 55.6, 61.1, 55.0, 65.0, 51.3, 0, 0.1)	0.8	5.6 286.0	0 0
(4.4, 4.2, 4.5, 4.4, 4.4, 4.6, 4.6, 3.3) (84.4, 71.7, 98.0, 84.1, 88.0, 99.9, 101.8, 27.3)	(2.6, 2.5, 2.7, 2.6, 2.7, 2.7, 2.6, 1.9) (13.1, 12.2, 14.3, 13.1, 14.3, 14.0, 13.7, 5.8)	0.9	5.1 165.1	0 0

vectors has angle less than the threshold α. The training and validation loss curves are shown in Figure 5b (for threshold $\alpha = 1°$) and Figure 5c (for threshold $\alpha = 3°$). It is easy to see that, after filtering, the convergence of training loss becomes faster, confirming our hypothesis that the "conflicting samples" slow down the neural network training. However, we observe that, after filtering, the validation loss becomes worse, suggesting that these similar input vectors contain "true" information that cannot be simply discarded. This re-affirms the difficulty of learning and predicting the cellular network throughput.

VI. Conclusion

We studied the feasibility of accurate throughput prediction in today's cellular networks under high mobility and at fine time-scales, e.g., in the order of 100 ms. Using a dataset we collected through a 4-day, 1000+ km driving trip, we found that popular ML models previously claimed to predict well for various wireless networks scenarios can not predict well under app-centric metrics such as ARE95 and PARE10. We further utilized recent ML theories developed based on over-parameterized ML models to uncover the root cause for the limitations of ML models as the conflicting sample sequences in fine-grained cellular network throughput – cellular network throughout can be so dynamic that very similar sample throughput sequences can be followed by very different next samples, forcing the ML models to learn "spiky" functions, which are often very slow to learn and not predicting well.

Acknowledgement. This work is supported in part by Cisco Research and by NSF grants CNS-2312834, CNS-2112778, and CNS-2211459.

References

[1] "Networking Research in the Age of AI/ML: More Science, Less Hubris," https://infocom2024.ieee-infocom.org/program/keynote.

[2] M. Ghoshal, S. Mohanti, and D. Koutsonikolas, "Enabling Emerging Applications in 5G Through UE-Assisted Proactive PHY Frame Configuration," in *Proc. of IEEE PIMRC*, 2024.

[3] S. Aggarwal, U. S. Sardesai, V. Sinha, D. D. Mohan, M. Ghoshal, and D. Koutsonikolas, "LiBRA: Learning-Based Link Adaptation Leveraging PHY Layer Information in 60 GHz WLANs," in *Proc. of ACM CoNEXT*, 2020.

[4] M. Polese, F. Restuccia, and T. Melodia, "DeepBeam: Deep Waveform Learning for Coordination-Free Beam Management in mmWave Networks," in *Proc. of ACM MobiHoc*, 2021.

[5] J. Hall, N. Thawdar, T. Melodia, J. Jornet, and F. Restuccia, "Deep Learning at the Physical Layer for Adaptive Terahertz Communications," *IEEE Transactions on Terahertz Science and Technology*, 2023.

[6] Y. Zhang, T. Osman, and A. Alkhateeb, "Online beam learning with interference nulling for millimeter wave MIMO systems," *IEEE Transactions on Wireless Communications*, vol. 23, no. 5, 2024.

[7] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan, "An experimental study of the learnability of congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 479–490, 2014.

[8] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "PCC vivace: Online-learning congestion control," in *Proc. of NSDI*, 2018.

[9] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, "Owl: Congestion control with partially invisible networks via reinforcement learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[10] S. Rajasekaran, M. Ghobadi, G. Kumar, and A. Akella, "Congestion control in machine learning clusters," in *Proc. of ACM HotNets*, 2022.

[11] X. Liao, H. Tian, C. Zeng, X. Wan, and K. Chen, "Towards fair and efficient learning-based congestion control," *arXiv preprint arXiv:2403.01798*, 2024.

[12] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," in *Proc. of ACM SIGCOMM*, 2017.

[13] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: A randomized experiment in video streaming," in *Proc. of USENIX NSDI*, 2020.

[14] J. Meng, Q. Xu, and Y. C. Hu, "Proactive energy-aware adaptive video streaming on mobile devices," in *Proc. of USENIX ATC*, 2021.

[15] B. Han *et al.*, "ViVo: Visibility-Aware Mobile Volumetric Video Streaming," in *Proc. of ACM MobiCom*, 2020.

[16] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and Delay-Tolerance," in *Proc. of USENIX NSDI*, 2017.

[17] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, and N. Dai, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," in *Proc. of ACM MobiCom*, 2017.

[18] J. Meng, S. Paul, and Y. C. Hu, "Coterie: Exploiting frame similarity to enable high-quality multiplayer VR on commodity mobile devices," in *Proc. of ASPLOS*, 2020. [Online]. Available: https://doi.org/10.1145/3373376.3378516

[19] Z. J. Kong, Q. Xu, J. Meng, and Y. C. Hu, "Accumo: Accuracy-centric multitask offloading in edge-assisted mobile augmented reality," in *Proc. of ACM MobiCom*, 2023.

[20] Z. J. Kong, Q. Xu, and Y. C. Hu, "Arise: High-capacity ar offloading inference serving via proactive scheduling," in *Proc. of ACM MobiSys*, 2024.

[21] M. Rudow, F. Y. Yan, A. Kumar, G. Ananthanarayanan, M. Ellis, and K. Rashmi, "Tambur: Efficient loss recovery for videoconferencing via streaming codes," in *Proc. of USENIX NSDI*, 2023.

[22] X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao, "EMP: Edge-assisted Multi-vehicle Perception," in *Proc. of ACM MobiCom*, 2021.

[23] "XCAL Solo," https://accuver.com/sub/products/view.php?idx=11;.

[24] C. Liu, L. Zhu, and M. Belkin, "Loss landscapes and optimization in over-parameterized non-linear systems and neural networks," *Applied and Computational Harmonic Analysis*, vol. 59, pp. 85–116, 2022.

[25] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. of ICLR*, 2016.

[26] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off,"

Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.

Proceedings of the National Academy of Sciences, vol. 116, no. 32, pp. 15 849–15 854, 2019.
[27] "Fine-grained cellular throughput prediction dataset," 2024. [Online]. Available: https://github.com/NUWiNS/MASS_2024_Throughput_Prediction/tree/main
[28] J. Jung, S. Lee, J. Shin, and Y. Kim, "Self-attention-based uplink radio resource prediction in 5g dual connectivity," *IEEE Internet of Things Journal*, vol. 10, pp. 19 925–19 936, 2023.
[29] J. Lee *et al.*, "PERCEIVE: Deep Learning-Based Cellular Uplink Prediction Using Real-Time Scheduling Patterns," in *Proc. of ACM MobiSys*, 2020.
[30] M. Ghoshal *et al.*, "An In-Depth Study of Uplink Performance of 5G mmWave Networks," in *Proc. of ACM 5G-MeMU*, 2022.
[31] M. Ghoshal, I. Khan, Z. J. Kong, P. Dinh, J. Meng, Y. C. Hu, and D. Koutsonikolas, "Performance of Cellular Networks on the Wheels," in *Proc. of ACM IMC*, 2023.
[32] E. Eyceyurt, Y. Egi, and J. Zec, "Machine-learning-based uplink throughput prediction from physical layer measurements," *Electronics*, vol. 11, no. 8, p. 1227, 2022.
[33] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei, "Linkforecast: Cellular link bandwidth prediction in lte networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1582–1594, 2017.
[34] F. Jomrich, A. Herzberger, T. Meuser, B. Richerzhagen, R. Steinmetz, and C. Wille, "Cellular bandwidth prediction for highly automated driving," in *Proc. of VEHITS 2018*, 2018.
[35] D. Minovski, N. Ögren, K. Mitra, and C. Åhlund, "Throughput prediction using machine learning in lte and 5g networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1825–1840, 2021.
[36] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. Fezeu, U. K. Dayalan, S. Verma, P. Ji, T. Li *et al.*, "Lumos5g: Mapping and predicting commercial mmwave 5g throughput," in *Proc. of ACM IMC*, 2020.
[37] T. Azmin, M. Ahmadinejad, and N. Shahriar, "Bandwidth prediction in 5g mobile networks using informer," in *Proc. of IEEE Conference on Network of the Future (NoF)*, 2022.
[38] C. Fiandrino, G. Attanasio, M. Fiore, and J. Widmer, "Toward native explainable and robust ai in 6g networks: Current state, challenges and road ahead," *Computer Communications*, vol. 193, pp. 47–52, 2022.
[39] C. Fiandrino, E. Perez Gomez, P. Férnandez Pérez, H. Mohammadalizadeh, M. Fiore, J. Widmer *et al.*, "Aichronolens: Advancing explainability for time series ai forecasting in mobile networks," in *Proc. of IEEE INFOCOM*, 2024.
[40] L. Mei, R. Hu, H. Cao, Y. Liu, Z. Han, F. Li, and J. Li, "Realtime mobile bandwidth prediction using lstm neural network," in *Proc. of PAM*, 2019.
[41] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: A 5g dataset with channel and context metrics," in *Proc. of ACM MM*, 2020.
[42] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, F. Qian, and Z.-L. Zhang, "A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications," in *Proc. of ACM SIGCOMM*, 2021.
[43] *nuttcp - Network Performance Measurement Tool*, Online. [Online]. Available: https://www.nuttcp.net
[44] S. Aggarwal, Z. Kong, M. Ghoshal, Y. C. Hu, and D. Koutsonikolas, "Throughput Prediction on 60 GHz Mobile Devices for High-Bandwidth, Latency-Sensitive Applications," in *Proc. of PAM*, 2021.
[45] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, p. 2114–2124.
[46] C. Liu, D. Drusvyatskiy, M. Belkin, D. Davis, and Y. Ma, "Aiming towards the minimizers: fast convergence of sgd for overparametrized problems," *Advances in neural information processing systems*, vol. 36, 2023.
[47] C. Liu and L. Hui, "Relu soothes the ntk condition number and accelerates optimization for wide neural networks," 2023. [Online]. Available: https://arxiv.org/abs/2305.08813

Authorized licensed use limited to: Purdue University. Downloaded on June 05,2025 at 16:00:09 UTC from IEEE Xplore. Restrictions apply.