Multi-cell Multi-beam Prediction using Auto-encoder LSTM for mmWave systems

Syed Hashim Ali Shah, *Graduate Student Member, IEEE* and Sundeep Rangan, *Fellow, IEEE*

***Abstract*—Millimeter wave (mmWave) systems rely on communication in narrow beams for directional and spatial multiplexing gains. A key challenge in realizing these systems is beam tracking, particularly in environments with high mobility and blockage. Additionally, in wide-area mmWave cellular systems, user equipment (UE) devices must often simultaneously track signals from multiple cells, since links to individual cells can be unreliable. Models of the channel dynamics across multiple cells and multiple beams are difficult to derive from first principles. In this work, we propose a fully data-driven approach based on a novel auto-encoder integrated long short term memory (LSTM) network, which predicts multiple beams from multiple cells, one time step in the future. The key innovation is to use an auto-encoder pre-processing step, which reduces the dimensionality of the input – the main challenge in multi-cell, multi-beam tracking. The prediction capability of the proposed network is verified and compared to common baseline predictors as well as popular machine learning (ML) based neural network predictors in realistic system-level simulations using a commercial ray-tracer. We observe that predictions from the proposed network, which utilizes auto-encoders for dimensionality reduction, offers significantly better best beam accuracy and lower beam misalignment loss than common baseline approaches. We also discuss outage prediction and proactive beam switching as applications of the multi-cell multi-beam prediction.**

***Index Terms*—Millimeter wave (mmWave) communications, LSTM, machine learning, cellular wireless, 5G, NR, dimensionality reduction, ray tracing, channel prediction, multi-connectivity, beamforming.**

I. Introduction

A. Motivation

Millimeter wave (mmWave) wireless systems have emerged as a key component of fifth generation (5G) cellular standards [2]. The abundance of available bandwidths at these frequencies can enable both massive broadband and ultra-low latency communications for use cases including vehicle to everything (V2X) communications, robotics, drones, healthcare, augmented reality and virtual reality.

A well-known challenge of mobile communication at these frequencies is beam tracking. To overcome the high isotropic path loss in the mmWave frequencies, both the transmitter (TX) and receiver (RX) must typically communicate in narrow, steerable directional beams. Tracking these beams at a high angular resolution is challenging, particularly in high mobility environments. In addition, mmWave signals are highly susceptible to blockage from humans, hands and many everyday building materials [3]–[6]. Thus, small changes in the orientation of the device or appearance of blockers can result in a rapid degradation of link quality in any given direction. As a result, mmWave systems often need to track and predict link quality along multiple directions to guarantee reliable communication. Moreover, most mmWave systems rely on dense cell deployments combined with multi-connectivity to provide macro-diversity resistance to blockage [7]. Multi-connectivity can be supported via carrier aggregation [8] where a mobile (UE) can be simultaneously connected to multiple cells. Hence, the mobile needs to track and predict link quality not only from multiple directions (beamforming) but also multiple directions from multiple cells (beamforming coupled with macro-diversity).

The broad goal of this work is to understand the problem of tracking multiple beams from multiple cells. We will refer to the signal path from one cell to a UE along a particular TX and RX direction pair as a *link*. Each link has a time-varying quality. The problem is to use past measurements to estimate a set of future link qualities as well as the best link indices (that have the best quality) with high accuracy. These future estimates can help mobile mmWave wireless systems accurately track links and proactively switch them when needed.

Traditional statistical prediction approaches are difficult, since link statistics are complex and difficult to model from first principles. The link qualities in particular can have intricate statistical relationships between different angles and base stations. We thus propose a machine learning approach where the prediction algorithms can be trained from data. Specifically, we formulate the multi-beam multi-cell prediction problem as a vector-valued sequence-to-sequence problem and solve it using recurrent neural networks (RNNs) and auto-encoders. In our work, we use a well-known RNN called long short term memory (LSTM), which has worked for similar problems [9], [10]. LSTMs can capture long-term dependencies and have been successful in a range of problems, particularly in natural language processing (NLP), speech recognition and robotics. Auto-encoders are used to reduce input dimensions of LSTM. Our contributions in this paper are:

(a) *Novel neural network architecture with an auto-encoding*

The authors are with NYU WIRELESS, Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA (Email: {s.hashim, srangan}@nyu.edu).

This work was supported by the National Science Foundation under Grants 1302336, 1564142, and 1547332, NIST, SRC and the industrial affiliates of NYU WIRELESS. A part of this paper was presented at the IEEE International Conference on Communications (ICC) held in Dublin, Ireland [1].

precoder: We propose an auto-encoder integrated LSTM network capable of predicting all link qualities with correct indices, one time step ahead in the future (e.g., $20\,\mathrm{ms}$, the typical period for reference signals in 5G NR [11]). A key challenge in these architectures is the large raw signal dimension, particularly when the number of base stations and directions is high, as discussed later in Section I-B. The high dimension can result in poor generalization and a high computational cost. We thus propose a novel auto-encoder based pre-coder for initial dimensionality reduction. We demonstrate in simulations that the auto-encoder based LSTM can offer a reduction of approximately 260% in the number of parameters with improved performance compared to standard dimensionality reduction methods such as principal component analysis (PCA).

(b) *Ray tracing evaluation:* To validate the method, we generate traces of link quality – signal to noise ratio (SNR) – using a commercial ray-tracer at mmWave frequencies. The traces mimic a car with a multiple antenna receiver moving in downtown Rosslyn, which is connected to multiple cells with multiple antennas. These traces capture most of the major channel characteristics like multi-path, mobility and blockages from buildings. To make the scenario more realistic, we also implement a measurement-based hand blockage model on top of the ray-tracer generated SNRs. The generated data set is also useful for ML-based wireless research, as discussed in Section I-B. After data generation, we test the prediction performances of the methods mentioned in (a) on the traces. Over multiple test trajectories of the generated data, the average test error in best link prediction from the proposed predictor is 90% of the time less than $2\,\mathrm{dB}$, outperforming optimally tuned baseline linear predictors (ML predictors) by at least 78% (10%) at the same percentile. Similarly, for the top 10 links, the average error is less than $2\,\mathrm{dB}$ for 94% of the time, which is 86% (8%) better than the linear baseline predictors (ML-based predictors). Furthermore, the error due to misalignment of best predicted beams is less than $2\,\mathrm{dB}$ for 98% of the time using the proposed predictor, outperforming baseline linear predictors by at least 86%.

(c) *Site-specific training:* An important implication of the work is that we offer a method for *site-specific* training, where the prediction of links from a particular collection of base stations can be optimized. Site-specific models can be run in the network (where the UE reports measurements to the network) or in the UE (where the network provides the UE parameters). This site-specific training, using an *edge server*, is demonstrated in Fig. 1.

(d) *Applications for beam management procedures:* We also discuss *outage prediction* and *proactive beam-switching* as applications of the proposed predictors. Although the desired predictors are not optimized for these applications, we observe that predictors still deliver adequate accuracy. The proposed auto-encoder integrated LSTM predictor can successfully predict outages 96% (80% for the best baseline linear predictor) of the time with a false alarm prediction lower than 5% (same for the best baseline linear predictor). Similarly, the proposed predictor can proactively switch beams with a 91% accuracy (81% for the best baseline linear predictor), while keeping the false beam-switching rate lower than 2% (10% for baseline predictor).

B. Related Work

There is now a growing body of work on deep learning methods for various forms of link prediction and channel estimation. For example, previous work on single link quality predictions have been done at sub-6 GHz frequency for a vehicular scenario in [13]. Work on link prediction based on LTE and WiMax measurements has been done in [14]. CSI estimation using deep learning has also been addressed in [15] and tested on sub-6 GHz measurements. The work [16] uses RNNs for a very simple LTE-MIMO system (only four links) with no blockages, and [17] uses LSTMs to predict RSSIs (one link) for different sub-6 GHz interfaces, while [18] has developed neural network models for single beam estimation from non-coherent measurements and validated these in experiments. Our work however, tackles multi-beam multi-cell prediction at the mmWave frontier, which is more complicated because of the channel impediments like narrow beams, severe blockages, complex interactions with the environment, etc. Importantly, since we consider tracking a much larger number of links, the role of the dimensionality reduction is key. As shown in [19], the number of beams increases as carrier frequency f_c increases ($\propto f_c^2$)[1]. Increasing f_c results in severe blockage [20] and penetration loss [21], which necessitates more macro-diversity (multi-cell connectivity), thereby increasing the input dimensions even more. As we move to next generation wireless networks (higher f_c) with even higher

[1]For example, if f_c is increased from 28 to $140\,\mathrm{GHz}$ (sub-THz, 6G), the number of directions to track increase by a factor of 32 [19], which equals increase in input dimension size in our case.

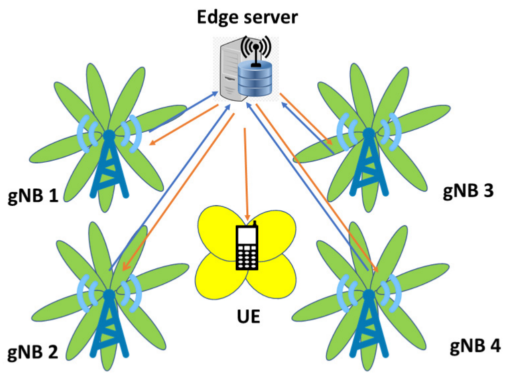


Fig. 1: Demonstration of site-specifc training. The blue arrows indicate gNBs sending data to the edge server. The edge server collects the data and trains the neural network. Once trained, the parameters of the network are broadcast to all gNBs and UEs, indicated by orange arrows. Similar architecture is proposed in [12]. The channel between the edge sever and UE/gNBs is not part of training the ML network.

dimensional inputs, the dimensionality reduction will become more crucial for ML-aided wireless communications. To the best of our knowledge, this is the first work that tackles the increased dimensionality problem (because of increased beams and multi-cells) for mmWave wireless systems using auto-encoders.

A related line of work [22]–[24] tackles beam and blockage predictions at mmWave, leveraging sub-6 GHz links (*non-standalone* mode of operation). In our work, we solve the multi-cell multi-beam prediction problem solely based on mmWave links (*standalone* mode of operation). Also, [25] and [26] use ML for mmWave link blockage classification and prediction, while [27] uses gated recurrent unit (GRU) for blockage prediction and proactive hand-over in a simplistic environment. However, these works do not address the link quality (SNR) and link index prediction problem. Our work confronts the link SNR and index prediction problem in a realistic environment based on 3GPP parameters, measurement campaigns and proved works. These prediction capabilities will help in processes like proactive beam switching, handovers [27] and adaptive rate prediction. V2X, robotics and drone communications can also benefit from these proactive applications. To the best of our knowledge, this work is unique in solving the multi-beam, multi-cell magnitude and index prediction problem for mmWave systems.

Finally, a key challenge in ML methods is the need for large quantities of training data. A common theme in many prior works, such as [17], [24], [28], has been the use of ray tracing, which enables large quantities of training points to be generated via electromagnetic simulations. Ray tracing has also been vital in training deep generative models [29], [30]. This work also uses ray tracing combined with hand blockage models to capture local effects not included in a conventional ray tracer. Since the ray tracing scenario conforms with the 3GPP NR standard at mmWave frequencies, the generated data set is essential to foster research in ML-assisted wireless communications (similar to the DeepMIMO data [28])[2].

C. Organization

Section II defines some system parameters for link measurements based on 3GPP standards, which will help us align our work with the standard. We formulate the single-step ahead prediction problem in Section III and define some performance metrics, which are useful from a wireless communications perspective. Section IV presents proposed LSTM-based predictors and an argument about the need for dimensionality reduction, which will be achieved using auto-encoders and PCA. We also introduce some other ML-based neural network (NN) predictors for comparison in this section. In Section IV-D, we present some baseline linear predictors to which performance of NN-based predictors will be compared. Discussion on a detailed and realistic simulation setup based on a commercial ray tracer is included in Section V. In Section VI, training and tuning of hyper-parameters of the proposed predictors as well as the baseline predictors are given. In Section VII-A, we compare prediction performance of all predictors and observe how the proposed predictor outperforms the baseline linear predictors as well as other NN predictors. We discuss prediction performances of all the predictors for various applications in Section VII-B. Finally, Section VIII concludes the paper with a summary.

[2]The data set can be found at https://github.com/shastpi/mmWave-ray-tracer-dataset

II. System Parameters

Although our methodology is general, to make the analysis concrete we will focus on tracking and predicting the links for 3GPP NR-like systems, which can be reviewed below[3].

gNB and UE codebooks: In 5G NR terminology, the base station cell is called the gNB and the mobile is called the UE [31]. To simplify the analysis, we assume the gNB transmits from a codebook of N_{TX} possible directions, and the UE receives from a codebook of N_{RX} directions. Hence, for each gNB-UE pair there are $N_{\mathrm{TX}}N_{\mathrm{RX}}$ direction pairs. In general, we will assume that N_{TX} is equal to the number of TX antennas at the gNB, and N_{RX} is equal to the number of receive antennas at the UE. Hence, there is one codebook vector for each spatial degree of freedom. However, most of the framework can also be applied to over-sampled codebooks.

Reference signals (RS) for beam measurements: Beam tracking in 5G NR is done using reference signals such as synchronization signal blocks (SSBs) or channel state information reference signals (CSI-RS). SSBs are periodically broadcast on relatively wider beams from each 5G NR gNB for the purpose of base station discovery and downlink beam detection (usually in idle mode) [31]. CSI-RS on the other hand are sent on narrower beams during data transmission from gNB and enable beam tracking in mobile environments. The beamsweep is generally done in a hierarchical manner i.e., the SSBs with wider beams are first used to determine a *coarse* direction of transmission, which is then *refined* using reference signals like CSI-RS. However, in this work, we only consider narrow beams, which are referred to as refined beams after *beam refinement*. This brings us to our first assumption that *reference signals only use narrow beams for beam quality measurements (SNR)*. Measurements over narrow beams eliminate the hierarchical aspect of beam tracking and make beam tracking more challenging.

These RSs are transmitted in bursts with some periodicity T_{RS}. We set this interval to $20\,\mathrm{ms}$, which is consistent with SSB and CSI-RS periodicity in the 3GPP NR standards for carrier frequency of $28\,\mathrm{GHz}$ with a sub-carrier spacing (SCS) of $120\,\mathrm{kHz}$ [32]. In each RS burst, N_{RS} beams can be measured (typically with one TX direction for each RS). The parameters N_{RS} and T_{RS} are configurable. In simulations below, we will set $N_{\mathrm{RS}} = N_{\mathrm{TX}}$ allowing one RS in each downlink direction. The values of these and other important parameters are given in Table I.

[3]An excellent description of the NR protocol can be found in [31]

Parameter	**Value**	**Remarks**
Carrier frequency, f_c	28 GHz	Standard mmWave NR frequency for FR2
Occupied bandwidth, BW (MHz)	400	Maximum bandwidth for mmWave NR systems
Subcarrier spacing (SCS), kHz	120	SCS = 120 kHz and $BW = 400$ MHz are common for early 5G deployments at 28 GHz [32].
OFDM symbol duration, $T_{\rm sym}$ (μs)	8.92	Derived from SCS [33].
Number of UE (gNB) antennas, $N_{\rm RX}$ ($N_{\rm TX}$)	8 (64)	The array sizes at 28 GHz are similar to previous capacity analyses [34], [35].
Duration of one RS, $T_{\rm 1RS}$ (μs)	Varies	For SSB it is 4 OFDM symbols i.e., $4T_{\rm sym}$ [33]. For CSI-RS, it can be $\{1, 2, 4\}T_{\rm sym}$ depending on OFDM symbols used[33].
RS burst period, $T_{\rm RS}$ (ms)	20	From NR configuration [33].
Maximum number of RS during an SSB burst period $N_{\rm RS}$	64	$N_{\rm RS}$ has a maximum value of 64 for 5G NR systems for the default 120 KHz SCS [33].

TABLE I: 3GPP NR-based system parameters.

Network Model with Carrier Aggregation: Resilience to blockage at mmWave frequencies necessitates macro-diversity, i.e., the UE must be connected to multiple cells [7], [8]. To this end, we assume that the UE is connected to $N_{\rm gNB}$ gNBs via *carrier aggregation*, a key feature in 3GPP systems that enables simultaneous connections to multiple cells [8]. The cells either operate in different component carriers or within the same component carrier[4] – the analysis for this paper is identical. The above process does *not* require synchronization across cells.

The notions of RSs and carrier aggregation are introduced to justify an important assumption for our prediction method – *the UE/gNB are able to measure all the beams at each discrete time interval.* This discrete time interval in our case is $T_{\rm RS}$.

Our analysis can apply to both fully digital and analog beamforming at the UE. With fully digital beamforming, the UE can measure all $N_{\rm RX}$ directions every RS measurement period. Hence, after one RS burst of $N_{\rm TX}$ transmissions, all RX-TX pairs will have been measured. For analog beamforming during an RS burst allocated period, the UE can send uplink measurement signals (like sounding reference signals – SRS) to gNB from one of its beams, and a gNB with fully digital beamforming can measure all the beam-pairs for that particular beam[5]. In this manner, the complete beam sweep for all pairs will take $N_{\rm RX}$ such instances. In either beamforming case, since we assume carrier aggregation, the UE can measure the signal from all cells in each measurement burst. Henceforth, for simplicity, we will assume that beam sweeping is done at a fully digital beamformed UE via RS bursts. Therefore, the UE measures each synchronization resource individually. The other signals will appear as interference. Since mmWave systems are wideband and generally power limited, we have neglected this interference.

III. Problem Formulation

We index the discrete time steps (RS bursts) by $t = 0, 1, \ldots T$. Let $\gamma(i, j, l, t)$ denote the measured channel quality (i.e., SNR) from cell $i = 1, \ldots N_{\rm gNB}$, in TX direction $j = 1, \ldots N_{\rm TX}$, and RX direction $l = 1, \ldots, N_{\rm RX}$ at measurement period t. We merge the first three dimensions of the SNR tensor so it becomes $\gamma(k, t)$, where $k = 1, \cdots, K = N_{\rm TX} N_{\rm RX} N_{\rm gNB}$. We call each k a *link*. The matrix $\gamma(k, t)$ thus describes the variation of the link qualities over time. The variations will in general depend on UE motion, blocking, small-scale fading, hand blockage and other channel characteristics. The SNR measurement can be a wideband average SNR or effective SNR when there is frequency-selective fading.

We will often train on multiple trajectories where each trajectory is some route of the UE experiencing some blockage. In this case, we denote the SNR tensor for n-th trajectory as $\gamma_n(k, t)$. A trajectory consists of traces of SNRs on all beam-pairs at all gNBs for T time steps (refer to Section V-B for the exact definition). We consider predictors of the form,

$$\gamma^p(:, t) = \mathcal{P}[\gamma(:, t - M : t)], \tag{1}$$

where $\mathcal{P}[\cdot]$ is the prediction function, and we have used the python-like[6] notation to indicate that the predictor depends on all K links from the previous M time samples. The output is a prediction of all K links. The predictor can be a simple linear one such as moving average, or it can be more complex such as LSTM or GRU. Given training data of the n-th training trajectory, $\gamma_n(k, t)$, $n = 1, \ldots, N$, the predictors will be trained with the standard mean squared error (MSE) loss as defined in [37],

$$\begin{aligned}\mathcal{L} &:= \frac{1}{NTK} \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{t=1}^{T} (\gamma_n^p(k, t) - \gamma_n(k, t))^2 \\ &= \frac{1}{NTK} \|\gamma_n^p - \gamma_n\|_F^2.\end{aligned} \tag{2}$$

The proposed ML-based technique of minimizing loss in (2) has no theoretical guarantees, which tends to be the case in most ML works. Therefore, we opt the methodology followed by other ML-aided wireless works [18], [38] and instead rely on developing good training and testing data sets. Moreover, even for classic algorithms, theoretical guarantees are typically only given for simplified versions of the problems [18]. For complex problems, validation on data is done to prove effectiveness of the proposed scheme, which is consistent with

[4]According to [36], the UE can track up to 21 inter and intra-carrier frequency cells.

[5]RSs may be broadcast to all UEs whereas SRSs are gNB specific, so this method will affect scheduling efficiency of the gNB, because it has to get ready to measure all the uplink measurements. Furthermore, this assumption is only valid if the channel is reciprocal.

[6]In python, indexing of a vector $\mathbf{v}$ using $\mathbf{v}(a : b)$ means that we would like to obtain the values of the vector from index range $[a, b)$. Note that the index b is not included in the final values. So in (1), we are trying to predict the SNR values at time t based on the previous M SNR values on all links. Another notation we use from python is the **:** . If $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{A}$[1, :] means that we want the data from row 1 and *all* columns of $\mathbf{A}$. This idea can be similarly extended to the tensor γ.

our approach of testing. For our work, we will use several metrics in test. For example, given a test trajectory, $\gamma(k,t)$, we can evaluate the root mean squared error (RMSE) loss :

$$\epsilon_{\mathrm{RMSE}} := \sqrt{\frac{1}{TK}\sum_{k=1}^{K}\sum_{t=M}^{T}(\gamma^p(k,t)-\gamma(k,t))^2}.$$

However, the RMSE loss, while good for training, is not necessarily representative of performance. We thus consider two other test metrics: the *Top C link(s) prediction error* and *beam misalignment error*.

Top C link(s) prediction error, ϵ_C , captures the difference between the true C links with maximum SNR(s) and C predicted links with maximum SNR(s) at time t. Specifically, consider a test trajectory, $\gamma(k,t)$. At time t, let $\widehat{\gamma}(u,t)$ denote the values of $\gamma(k,t)$ sorted over k in descending order. Hence, $\widehat{\gamma}(u,t)$ represents the link qualities in sorted order. Similarly, for the predictions $\gamma^p(k,t)$, we define $\widehat{\gamma}^p(k,t)$. The *top C RMSE* is then defined as,

$$\epsilon_C = \sqrt{\frac{1}{CT}\sum_{t=1}^{T}\sum_{u=1}^{C}(\widehat{\gamma}^p(u,t)-\widehat{\gamma}(u,t))^2}. \tag{3}$$

We will use $C=\{1,10\}$ in this work. This metric captures how well a predictor can predict best C links from the available links. This metric is similar to Top-1 and Top-3 metrics discussed in [22]. The motivation for introducing this metric for $C=1$ is that if a UE/gNB is tracking multiple links, it will always choose the link with the best quality (SNR in our case) to transmit/receive on so it can yield maximum gains during communication. One of the use cases that can be derived from this metric is *proactive link rate adaption* where UE/gNB will adapt its transmission rate according to predicted SNR. Likewise, a use case for $C=10$ is proactive beam switching i.e., if the best beam/link is predicted to be blocked, the UE/gNB could switch to any of the predicted unblocked links. Hence, ϵ_C will indicate the accuracy of the best C link predictions.

Beam misalignment error ξ: The predictor must estimate not only the future best link quality but also the index of the best link. An instance might occur when a predictor predicts the best link quality accurately but mispredicts the index of the best link, which will result in *misalignment loss* in real time[7]. So in order to capture how precise best link index predictions are, we introduce ξ, which can be written as :

$$\xi = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\gamma(k^p(t),t)-\max_k \gamma(k,t))^2}, \tag{4}$$

[7]Qualitatively, this loss occurs to the predictors predicting wrong best beam indices. For example, let us consider a case with only 2 beams: beam a and beam b. The true SNRs on these beams are 10 and 15 dB, respectively. The network predicts the SNRs to be 13 dB and 12 dB, respectively. If we were to trust the predictions by network, we would choose the beam with maximum predicted SNR for communication, which is beam a. However, beam b is actually better than beam a, hence our choice to choose beam a over beam b will result in some degradation of performance. This degradation in the above example is 5 dB, so our beam misalignment error comes out to be 5 dB.

where $k^p(t)$ is the best predicted index from the estimated SNRs and can be written as:

$$k^p(t) = \arg\max_k \gamma^p(k,t). \tag{5}$$

A similar metric was used in [18]. The beam misalignment error is the difference between true SNR on the true best beam index (maximum true SNR) and true SNR on the best predicted beam index. The beam misalignment error we define in (4) is measured in dB − not to be confused with degrees, which is another measure of beam misalignment. Measuring loss in degrees might have different circumstances depending on the communication architecture e.g., a sub-6 GHz system might be able to provide a decent throughput even with a large misalignment loss in degrees, while the same is not true for mmWave systems. Due to this inconsistency of measuring misalignment loss in degrees, we opt to measure it in dB instead. Another advantage of measuring loss in dB is that it can be translated directly into other system performance metrics like throughput.

IV. Proposed Auto-Encoder Integrated LSTM Network

A. LSTM

In this work, we consider an LSTM [9], which is widely used for sequence-to-sequence prediction problems. LSTM is a natural choice for wireless tracking problems due to its ability to capture short-term dependencies (e.g., multi-path fading) and long-term dependencies (e.g., shadowing and blocking). LSTM networks from a research viewpoint − although relatively old − have been successful in ML-aided wireless communications [1], [39]–[41]. Moreover, LSTM based predictors are less complex in terms of architecture design as compared state of art sequence predictors like self-attention based transformers (e.g., BERT, which themselves have underlying RNNs). LSTM networks are designed to circumvent the *vanishing gradient* problem, which is prominent in RNNs. The aforementioned success and comparatively reduced complexity makes LSTM networks a suitable choice for next generation ML-aided wireless communications. The standard LSTM operation with q hidden units and d dimensional input is governed by the following set of equations:

$$\mathbf{g}(t) = \tanh(\mathbf{W}^{gx}\mathbf{x}(t)+\mathbf{U}^{gh}\mathbf{h}(t-1)+\mathbf{b}_g), \tag{6a}$$

$$\mathbf{i}(t) = \sigma(\mathbf{W}^{ix}\mathbf{x}(t)+\mathbf{U}^{ih}\mathbf{h}(t-1)+\mathbf{b}_i), \tag{6b}$$

$$\mathbf{f}(t) = \sigma(\mathbf{W}^{fx}\mathbf{x}(t)+\mathbf{U}^{fh}\mathbf{h}(t-1)+\mathbf{b}_f), \tag{6c}$$

$$\mathbf{o}(t) = \sigma(\mathbf{W}^{ox}\mathbf{x}(t)+\mathbf{U}^{oh}\mathbf{h}(t-1)+\mathbf{b}_o) \tag{6d}$$

$$\mathbf{s}(t) = \mathbf{g}(t)\odot\mathbf{i}(t)+\mathbf{s}(t-1)\odot\mathbf{f}(t), \tag{6e}$$

$$\mathbf{h}(t) = \phi(\mathbf{s}(t))\odot\mathbf{o}(t), \tag{6f}$$

$$\mathbf{z}(t) = \mathrm{ReLU}(\mathbf{W}^{zh}\mathbf{h}(t)+\mathbf{b}_z), \tag{6g}$$

where $\mathbf{x}(t) \in \mathbb{R}^d$ is the input vector to the LSTM unit, $\mathbf{i}(t) \in \mathbb{R}^q$ is the input gate, $\mathbf{f}(t) \in \mathbb{R}^q$ is the forget gate, $\mathbf{o}(t) \in \mathbb{R}^q$ is the output gate, $\mathbf{h}(t) \in \mathbb{R}^q$ is the hidden state vector, $\mathbf{s}(t) \in \mathbb{R}^q$ is the cell state vector and $\mathbf{g}(t) \in \mathbb{R}^q$ is the cell input activation vector. $\{\mathbf{W}^{gx}, \mathbf{W}^{ix}, \mathbf{W}^{fx}, \mathbf{W}^{ox}\} \in \mathbb{R}^{q\times d}$, $\{\mathbf{U}^{gh}, \mathbf{U}^{ih}, \mathbf{U}^{fh}, \mathbf{U}^{oh}\} \in \mathbb{R}^{q\times q}$ and $\{\mathbf{b}_g, \mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o\} \in \mathbb{R}^q$ are the weights and biases of respective gates that need to be learned during the LSTM training. The input gate decides whether the current incoming data is contributing new information to the network. The forget gate flushes out unwanted data from the memory. The output gate dictates what to show at the network output [10], [42]. The cell state keeps track of the memory of the unit, which includes both short-term and long-term memories. The hidden state vector is eventually used to predict the output variable. The hidden state can extract short-term, long-term or both types of memory stored in the cell state to make the prediction. The last equation (6g) represents a fully connected NN that takes the predicted hidden state vector from LSTM as an input and maps it to a d dimensional output $\mathbf{z}(t)$. Hence during training, the network also needs to learn weight matrix $\mathbf{W}^{zh} \in \mathbb{R}^{d\times q}$ and bias vector $\mathbf{b}_z \in \mathbb{R}^d$. In the above equations, $\odot$ represents element-wise multiplication, σ and ReLU represent *sigmoid* and *rectified linear unit* activation functions, respectively. These are given by:

$$\sigma(x) = \frac{1}{1+exp(-x)}, \tag{7}$$

$$\mathrm{ReLU}(x) = \max\{0, x\}. \tag{8}$$

A visualization of an LSTM cell unrolling in time with all the aforementioned parameters can be found in [10].

B. Dimensionality reduction via auto-encoding

The LSTM outputs will be the predicted link qualities one time step in the future:

$$\mathbf{z}(t) = \{\gamma^p(k,t),\ k = 1, \ldots, K\}. \tag{9}$$

For the inputs, we could use the raw measured SNR values[8]:

$$\mathbf{x}(t) = \gamma(:, t-M:t).$$

Now, the total number of parameters L_{LSTM} that are needed to train an LSTM network with q hidden units and d dimensional input is:

$$L_{\mathrm{LSTM}} = 4(qd + q^2 + q) + qd + d. \tag{10}$$

Using the raw SNR values (9) as inputs, the input and output dimensions would be $d = K = N_{\mathrm{TX}}N_{\mathrm{RX}}N_{\mathrm{gNB}}$. As we will see in Section VI-B, this number can be prohibitively large, therefore requiring a large number of LSTM parameters. The large number of parameters increases the generalization error and inference complexity. Thus, we also consider employing a dimensionality reduction of the form:

$$\mathbf{x}(t-M:t) = \Phi\left(\gamma(:, t-M:t)\right), \text{Encoding}, \tag{11}$$

$$\hat{\mathbf{x}}(t-M:t) = \Psi\left(\mathbf{x}(t-M:t)\right), \text{Decoding}, \tag{12}$$

[8]We use python-like notation here as well.

which transforms the K-dimensional SNR data at each time window M to some lower dimension $d' \leq K$ before it is sent to the LSTM. We call $\Phi(.)$ and $\Psi(.)$ encoder and decoder, respectively. The LSTM predicts $\mathbf{x}^p(t)$ one time step ahead and the decoder converts the predictions back to SNRs. The classic dimensionality reduction method is PCA, which can be trained on the set of SNR values $\gamma_n(k,t)$ over the training trajectories n and times t. The LSTM predicts $\mathbf{x}^p(t)$ and a decoder provides us with predicted SNRs $\gamma^p(k,t)$. We call this method *LSTM-PCA*. We will quantify the performance of dimensionality reduction methods using the following RMSE metric υ:

$$\upsilon = \sqrt{\frac{1}{NTK}\sum_{t=1}^{T}\sum_{n=1}^{N}\sum_{k=1}^{K}[\Psi(\Phi(\gamma_n(k,t)) - \gamma_n(k,t)]^2}. \tag{13}$$

We define the dimensionality reduction factor κ as:

$$\kappa = \frac{d'}{K} = \frac{d'}{N_{\mathrm{TX}}N_{\mathrm{RX}}N_{\mathrm{gNB}}}. \tag{14}$$

Both metrics above [(14),(13)] characterize the performance of a dimensionality reduction technique. The limitation of PCA is that it only performs linear dimensionality reduction – it is essentially a projection from the d-dimensional space to a lower d'-dimensional space. We thus consider auto-encoder based approach.

Choice of auto-encoders: To address the dimensionality reduction, we need to choose an auto-encoder that best serves our purpose. We choose undercomplete auto-encoders [43], which compress (encodes) large dimensional input data into lower dimensional signals (*bottleneck*). These signals are then used to recreate the original data[9]. We use undercomplete auto-encoders consisting of convolutional neural network (CNN) layers and hence are termed as convolutional auto-encoders (CAEs). In CAEs, the encoding function $\Phi(\cdot)$ is realized as a CNN. In addition, we train a *decoder* network Ψ that maps the low-dimensional $\mathbf{x}(t)$ back to the original space. Several loss functions are possible, and in this case, we use the standard MSE loss between the original $\gamma_n(k,t)$ and their reconstruction. See [44] for an example. We integrate the designed auto-encoder with LSTM and call this scheme *LSTM-AC*. The dimensionality of the hidden states d' as well as the encoder and decoder architectures are parameters in the network. We discuss their selection and design in Section VI-A.

Regardless of the dimensionality reduction method used, the network is trained on the one-step ahead MSE prediction loss (2). In training, we use M time steps of input, $\mathbf{x}(t-M), \ldots, \mathbf{x}(t-1)$, to generate each $\mathbf{z}(t)$. The parameter M, indicating the memory of the network, dictates the number of time steps over which the LSTM network unfolds. M is another parameter that needs to be tuned, and its value can be found in Section VI-B. Final design of the proposed LSTM-AC architecture is shown in Fig. 2.

[9]Another option maybe is de-noising auto-encoders (DAEs). Since we do not assume any noise in measuring SNRs, DAEs are not the best choice.

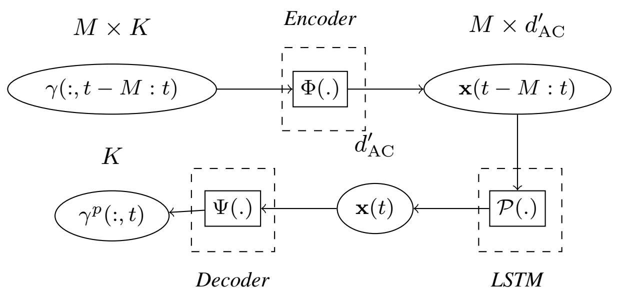


Fig. 2: Flowchart of the proposed LSTM-AC predictor. Ellipses represent input/output data while boxes represent neural networks (which are to be trained). The dimensions of respective inputs/outputs are shown on the top of ellipses. Dimensions $M \times K$ mean the data consists of M previous time samples of dimensions K.

C. Other ML-based NN predictors

In addition to LSTM-PCA and LSTM-AC, we also test the performance of some additional ML-based NN predictors to investigate how ML-based solutions perform for the given problem. These predictors are Vanilla RNN, Transformer and CNN. Vanilla RNN, which is a simple predecessor of LSTM, is tested to see how much gain LSTM provides over simple RNNs. Transformer-based predictor (used extensively in NLP) is much more complex than LSTM, because it uses self-attention[10] and is probed to observe how complex predictors perform. CNNs are explored to note how generic NN architectures perform, which are not designed for time series. All the aforementioned predictors will use auto-encoder in their architecture, since networks with large dimensional inputs are hard to train and provide poor generalization performance. The complexity, design and training of these ML-based predictors is discussed in Section VI-B.

D. Baseline linear predictors

We will also compare the prediction performance of the LSTM-based predictors to simple baseline linear predictors. The first is a simple moving average,

$$\gamma^p(k,t) = \frac{1}{M}\sum_{m=1}^{M} \gamma(k, t-m), \tag{15}$$

which takes the average of the previous M time steps. The parameter M can be optimized in the training phase. A more general estimator is a linear estimator,

$$\gamma^p(k,t) = \sum_{v=1}^{K}\sum_{m=1}^{M} W_{k,v,m}\gamma(\ell, t-m), \tag{16}$$

which takes a linear combination of the links in previous times. We allow dependencies from the predicted link k from all measured links v. The weights in the model, $W_{k,v,m}$, can be learned from minimizing the mean squared loss. Similar to moving average, M is a parameter that needs to be optimized for the linear estimator.

[10]Transformers themselves consist of RNNs in their architectures.

V. Simulation Setup

A. Scenario/Layout

A vital step in testing the prediction capabilities of different predictors is to generate a realistic data set of SNRs. The data should ideally come from real-life measurements. However, measurements which include exhaustive beamsweep of all the links between UE and multiple gNBs are hard to obtain and are not currently available. We therefore adopt a ray-tracer based approach, which enables much larger volumes of data. The ray tracing is accurate in that it captures paths from all propagation phenomena like diffraction, reflections and transmissions. We use the commercial ray tracer from Remcom called Wireless Insite [45], which has been widely used in research communities [24], [46] and has been verified through mmWave measurements [47], [48]. This ray-tracing package has also been widely used in many ML experiments [22], [28], [49], [50].

The first step in setting up the ray-tracer is to import the scenario layout. In our case, the scenario is downtown Rosslyn, Virginia[11]. The layout consists of building locations and dimensions in the area. The layout also includes materials from which these buildings are made so that the propagation mechanics like reflection, refraction and penetration of the scenario are accurately captured. Once the layout is imported into the ray-tracer, we place four gNBs (labeled BS in Fig. 3a) at some of the intersections in the city. The gNBs are approximately $200\,\mathrm{m}$ apart, translating to a cell radius of roughly $100\,\mathrm{m}$, which is consistent with the 3GPP Urban Micro "UMi" scenario [51]. These gNBs need to be assigned certain parameters like f_c, transmit bandwidth BW, etc. The ray tracer also needs to consider the total number of paths i.e., the number of paths to consider from each gNB to each receiver point. We set this property equal to 20 in accordance with the 3GPP UMi NLOS scenario [51]. The ray tracer is configured to show the paths with a maximum of 2 reflections, 1 transmission and 0 diffractions. As described in [52], [53], mmWave systems will mostly rely on reflections for multi-path propagation, justifying the choice to mostly focus on reflections. Similarly, the 1 transmission means we only consider penetration of a signal through one obstacle[12]. The main mode of signal propagation in our work is line of sight (LOS) paths and non-line of sight (NLOS) reflected paths. The main sources of reflections are the buildings and terrain (ground). Once all the aforementioned parameters (listed in Table I) have been set, we place receiver points over the entire layout grid spaced $0.5\,\mathrm{m}$ apart both in x and y axes. We deploy isotropic antennas at gNB and receiver points. Adding beamforming on top of these traces will be discussed in Section V-D. We now execute the ray-tracing. The ray-tracer output provides us with propagation information: (1) Received power on all the paths at each receiver point for

[11]This layout is also provided by Remcom inside Wireless Insite.

[12]Transmissions at mmWave are quite attenuated because of high penetration loss [21]. Hence, there is a very small chance that a signal transmitted through two different obstacles is received, so we only choose one transmission.

each gNB, (2) The spatial information (e.g., path lengths, angle of arrivals and departures) of all these paths at each receiver point for each gNB and (3) The temporal information (i.e., delays) of all these paths at each receiver point for each gNB. All this information from the ray-tracer is sufficient to start modeling link quality (SNRs). In the next section, we discuss the addition of mobility to the current scenario.

B. Mobility

As mentioned above, the ray-tracer provides all the received signal information on the points in the layout grid (spaced $0.5\,\mathrm{m}$). The next step is to add mobility to the scenario. The goal is to mimic a vehicle (UE) moving downtown with velocity given in Table IV (from [51]). We use the MATLAB Navigation Toolbox [54], which implements rapidly-exploring random tree (RRT) algorithm [55] to achieve this goal. MATLAB enables us to control various aspects of mobility: (a) Generating random routes for UEs, (b) Handling UE velocity in these routes and (c) Preventing collisions with obstacles (buildings) in these routes.

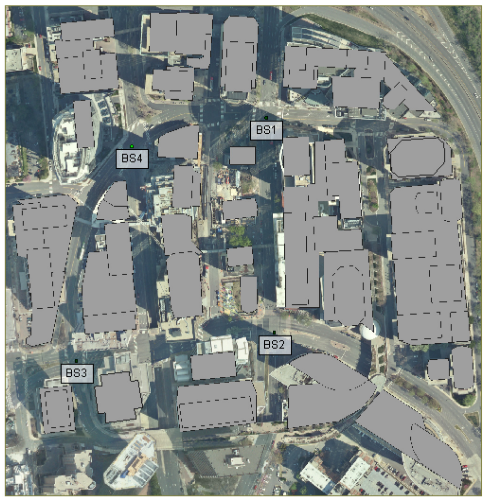
We start by importing the obstacle layout from the ray tracer to MATLAB. This layout is converted into *Binary Occupancy Grid* where length and width of each grid square is set to $0.5\,\mathrm{m}$. The binary occupancy grid assigns ones to the grid points where obstacles are present and zeros otherwise. At the beginning of each route, a starting point and an end point of the UE are sampled from the uniform distribution over the grid[13]. Similarly, a random velocity with distributions from Table IV is assigned to the UE. To avoid collisions, we use the *Navigation Toolbox* [54] from MATLAB, which works on a binary occupancy grid and ensures that the UE does not collide with any buildings during the course of its route. The UE continues to move until a total of $T = 3000$ samples spaced $20\,\mathrm{ms}$ apart ($60\,\mathrm{s}$ for each trajectory, in accordance to the beam measurement periodicity from 3GPP [33]) are collected. We refer to these $T = 3000$ samples as a *trajectory*[14]. A total of 200 trajectories (100 for training and 100 for testing) are generated. A generated trajectory with a binary occupancy grid is shown in Fig. 3b.

C. Hand blockage modeling

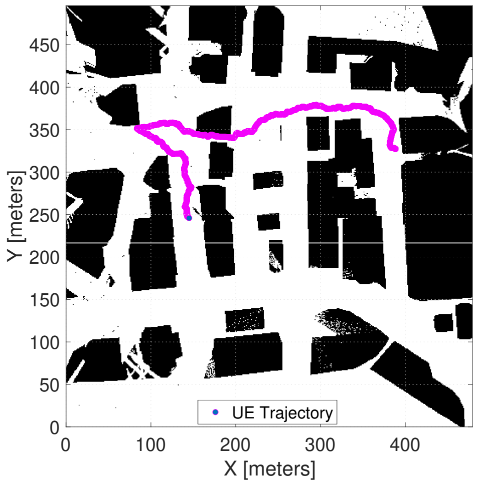
So far, the link quality generated from simulation trajectories captures the effect of multi-path, mobility and blockage by buildings. To make our simulations more realistic, we add hand blockage on link quality as well. As mentioned in Section I, hand blockage is also something that has to be overcome in the mmWave regime. We use a linear interpolated hand blockage model from [56] based on measurements at $28\,\mathrm{GHz}$. The model depends on the angle of arrivals of different paths

[13]It is ensured that the UE does not start inside any of the buildings (through a binary occupancy grid).

[14]Multiple routes might be generated during the trajectory until the required number of samples are collected. Multiple routes are connected together by their end/start points i.e., the end point of the older route becomes the start point of the new route, ensuring continuity.



(a) Actual layout of Downtown Rosslyn, VA



(b) Layout imported to MATLAB with UE trajectory

Fig. 3: Demonstration of UE mobility in one of the trajectories in Downtown Rosslyn, VA

at the UE and the orientation of the UE (landscape or portrait). Since we use a ray tracer for our simulations, all the spatial information needed to implement the hand blockage model is available. For orientation, we choose one randomly at the start of each trajectory with equal probability. A hand blockage event on any path is triggered if the azimuth (elevation) angle of arrival $\zeta(\theta)$ falls in the range $[\zeta_1 \pm \chi/2]$ ($[\theta_1 \pm \eta/2]$). The range signifies the azimuth (elevation) angular spread. Both azimuth and elevation angle of arrival conditions need to be true for a blockage event to be initiated. These conditions are similar to what has been proposed by the 3GPP standard to model hand blockage [51]. The values of ζ_1, χ, θ_1 and η have been listed in Table II. After triggering, the time dynamics of the blockage event are controlled by random variables τ_{d3dB}, τ_{r3dB} and τ_{Block}. Where τ_{D} is the total blockage event time , τ_{d3dB} and τ_{r3dB} represent the time taken for the signal to decay or rise by $3\,\mathrm{dB}$, respectively. These values (in ms) are generated upon triggering a blockage event and are provided in Table III [56], where $\mathrm{Weibull}(\alpha, \beta)$ denotes a weibull

Orientation	ζ_1	χ	θ_1	η	**Blockage Attenuation (A) (dB)**
Portrait mode	260^o	120^o	100^o	80^o	Weibull Distribution with $\alpha = 16.7$ and $\beta = 4.61$
Landscape mode	40^o	160^o	110^o	75^o	

TABLE II: Parameters for triggering hand blockage event

random variable with a probability distribution function (PDF) $f_{\text{Weibull}}(.)$ given by:

$$f_{\text{Weibull}}(x|\alpha,\beta) = \frac{\beta}{\alpha}(\frac{x}{\alpha})^{\beta-1}\exp(-(x/\alpha)^{\beta}),\ x \geq 0.$$

Parameters		τ_{d3dB}	τ_{r3dB}	τ_{D}
Weibull model parameters	α	0.044	0.045	0.59
	β	2.07	1.76	6.32

TABLE III: Parameters to model time dynamics of a hand blockage event

The blockage event on a particular path is modeled based on the parameters above using linear interpolation. We now define some parameters that will aid in this interpolation,

$$\tau_{\text{decay,rise}} = \frac{A\tau_{\text{d3dB,r3dB}}}{3}, \tag{17}$$

where τ_{decay} is the time needed for the signal level to decay to A dB from the initial signal level. τ_{rise} is the time needed to rise to the normal signal level from A dB. The 3 in the denominator is because the transition is measured every $3\,\text{dB}$. The total time of blockage event is given by:

$$\tau_{\text{Block}} = \max(\tau_{\text{decay}} + \tau_{\text{rise}}, \tau_D). \tag{18}$$

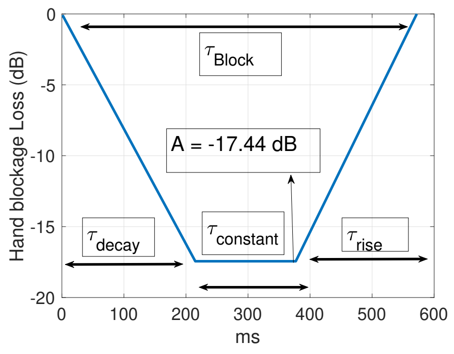
The time during which the signal level remains constant at A dB during a blockage interval τ_{constant} is given by:

$$\tau_{\text{constant}} = \tau_{\text{Block}} - (\tau_{\text{decay}} + \tau_{\text{rise}}). \tag{19}$$

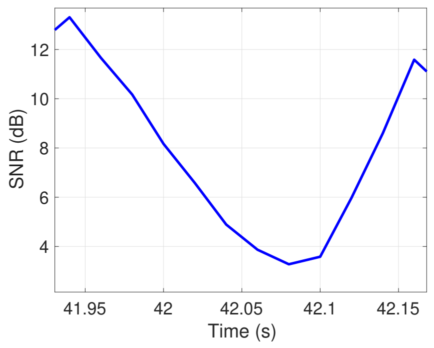
We now have all the parameters required to represent the blockage event in time. The loss suffered by hand blockage $\rho(\tau)$ at time sample τ can be represented in the following piece-wise linear manner:

$$\rho(\tau) := \begin{cases} \frac{A}{\tau_{\text{decay}}}\tau, & 0 \leq \tau < \tau_{decay}, \\ A, & \tau_{\text{decay}} \leq \tau < \tau_{\text{decay}} + \tau_{\text{constant}}, \\ A - \frac{A}{\tau_{\text{rise}}}\tau, & \\ \quad \tau_{\text{decay}} + \tau_{\text{constant}} \leq \tau < & \\ \quad\quad \tau_{\text{decay}} + \tau_{\text{constant}} + \tau_{\text{rise}} = \tau_{\text{Block}}. & \end{cases} \tag{20}$$

It should be noted that $A < 0$ since it measures loss. Fig. 4a shows a blockage event labeled with all the parameters mentioned above. Fig. 4b shows an instance in the trajectory where a link suffers from hand blockage. The figure shows how the SNR degrades by $10\,\text{dB}$ in $100\,\text{ms}$ just by hand blockage. Other factors that contribute to this degradation (not shown in figure) in the simulation setup are blockage by buildings and fast fading (since coherence time of channel is small). The factors all together may result in very frequent degradation of a link that is being tracked. This necessitates a good predictor, which can accurately predict on all links so the gNB/UE can always track the best link.



(a) Sample hand blockage event with labeled variables



(b) Hand blockage event identified in one of the links from one of the trajectories

Fig. 4: Hand Blockage Model

D. Beamforming codebook design

A 4×2 uniform planar array (UPA) with $\lambda/2$ antenna spacing is assumed at the UE and an 8×8 UPA is assumed at the gNB. These sizes for $28\,\text{GHz}$ are similar to past capacity analyses such as [34]. We assume two identical antenna arrays at the UE and gNB for full 360 degree coverage, like practical devices [57] (i.e., one array covering the front hemisphere and the other covering the rear). Let $\mathcal{F}_j := \{\mathbf{f}_j^{(1)}, \mathbf{f}_j^{(2)}\}$ ($\mathcal{W}_l := \{\mathbf{w}_l^{(1)}, \mathbf{w}_l^{(2)}\}$) denote the pair of gNB (UE) beamforming vectors corresponding to the j-th (l-th) TX (RX) direction, where $\mathbf{f}_j^{(1)}, \mathbf{f}_j^{(2)} \in \mathbb{C}^{N_{\text{TX}}}$ ($\mathbf{w}_l^{(1)}, \mathbf{w}_l^{(2)} \in \mathbb{C}^{N_{\text{RX}}}$), correspond to the front and rear antenna arrays, respectively. We consider a simple beamforming codebook based on the steering vector of a UPA, such that the main lobes of the beam patterns cover the hemisphere, equally spaced in both azimuth and elevation. We refer the reader to [19] for the expressions of $\mathbf{f}_j^{(1)}, \mathbf{f}_j^{(2)}, \mathbf{w}_l^{(1)}$ and $\mathbf{w}_l^{(2)}$.

E. SNR calculation

Given the rays for respective paths from the ray tracer for a trajectory n, we compute the narrowband channel matrix for

Parameters	**Value**	**Parameters**	**Value**	**Parameters**	**Value**
f_c	28 GHz	Bandwidth (‡)	400 MHz	N_{TX} (‡)	64
N_{RX} (‡)	8	N_{gNB} [36]	4	UE height (†)	1.7 m
gNB height (†)	10 m	Cell Radius [58]	∼100 m	UE Velocity (†)	$\mathcal{U}$[0,27] m/s
Transmit Power [59]	23 dBm	Noise Figure [60]	9 dB	Sampling Interval [33]	20 ms
N_{Paths}	20	Number of Reflections	2	Number of Transmissions	0

TABLE IV: Simulation parameters used for generating channel trajectories with ray-tracer. (†: [51], ‡: [61]). $\mathcal{U}[a,b]$ denotes a uniform random variable over $[a,b]$.

the i-th gNB[15], $\mathbf{H}_{i,n}(t)$. We apply the beamforming vectors to compute the SNR on each link. The expression for $\gamma_n(i,j,l,t)$ is as follows:

$$\gamma_n(i,j,l,t) = 10\log_{10}\left(\max_{\substack{\mathbf{w}_l\in\mathcal{W}_l,\\ \mathbf{f}_j\in\mathcal{F}_j}} \frac{|\mathbf{w}_l^H\mathbf{H}_{i,n}(t)\mathbf{f}_j|^2}{k_{\text{B}}(BW)N_F T_0}\right), \quad (21)$$

where k_{B} is Boltzmann's constant, BW denotes the system bandwidth, N_F is the noise figure, and T_0 is the temperature. We flatten the $i=1,\cdots,N_{\text{gNB}}$, $j=1,\cdots,N_{\text{TX}}$ and $l=1,\cdots,N_{\text{RX}}$ dimensions to $k=1,\cdots,N_{\text{TX}}N_{\text{RX}}N_{\text{gNB}}=K$, which is often done in machine learning problems. $\gamma_n(k,t)$ is given by:

$$\underset{i=1,\cdots,N_{\text{gNB}}\ ,j=1,\cdots,N_{\text{TX}}\ ,l=1,\cdots,N_{\text{RX}}}{\gamma_n(i,j,l,t)} \rightarrow \text{Flatten} \rightarrow \quad (22)$$
$$\underset{k=1,\cdots,N_{\text{TX}}N_{\text{RX}}N_{\text{gNB}}}{\gamma_n(k,t)}.$$

In total, we have $N_{\text{traj}}=200$ trajectories of $\gamma_n(k,t)$ with $T=3000$ samples, each spaced 20 ms apart. All the values of SNRs are in dB. The values of SNRs obtained by simulations will have large fluctuations depending on the link between UE and gNB. These large fluctuations will result in a large range of SNRs[16], which will result in poor performance of the predictors. To make the range smaller, we clip the SNRs at a lower threshold γ_{lower} and an upper threshold γ_{upper} which can be calculated from the classic Shannon's equation:

$$\begin{aligned}\gamma_{\text{lower}} &= 10\log_{10}(2^{\eta_{\text{lower}}}-1)\ \text{dB} &= -7.5\,\text{dB},\\ \gamma_{\text{upper}} &= 10\log_{10}(2^{\eta_{\text{upper}}}-1)+\Delta\ \text{dB} &= 25.27\,\text{dB},\end{aligned} \quad (23)$$

where $\eta_{\text{lower}}=0.2344\,\text{bps/Hz}$ is the spectral efficiency offered by the lowest modulation and coding scheme (MCS 0) according to 3GPP NR standards [62]. Hence, γ_{lower} is the ideal lowest SNR at which a signal can be decoded. If the UE/gNB is not able to measure SNR on a link (from blockage or any other reason), it reports a value of γ_{lower} on that link. Similarly, $\eta_{\text{upper}}=7.4063\,\text{bps/Hz}$ dictates upper bound on SNR since there is no change in throughput afterwards. Δ indicates how far the system is operating from the Shannon capacity and is set to 3 dB [35].

VI. Training and tuning predictors

A. Performance evaluation of auto-encoders and PCA

The encoders and decoders of the auto-encoder are designed to reduce the dimensions of the links from K to d'_{AC} and then back to K. We use 50% of the SNR trajectories for training ($N_{\text{train}}=100$, total number of training samples $=N_{\text{train}}\times T$). The depth (number of layers) and width (hidden units) contribute to the number of auto-encoder parameters L_{AC} that are to be optimized. Too many parameters will cause processing inefficiency, while fewer parameters will result in information loss. Training epochs will similarly impact the training time and over/under-fitting of the data. We use cross-validation to roughly find these hyper-parameters that provide a good processing-accuracy trade-off. These parameters along with CAE architecture is shown in Fig. 5. The proposed CAE takes input tensor with dimensions $N_{\text{train}}T\times N_{\text{gNB}}\times N_{\text{RX}}\times N_{\text{TX}}$ (refer to Table IV for values) and the encoder returns a tensor of dimensions $N_{\text{train}}T\times N_{\text{gNB}}\times N_{\text{RX}}\times 8$[17], reducing input dimensions by a factor of 8. The tensor is then flattened into a d'_{AC} dimensional vector. This flattening is necessary to make the CAE compatible with ML-based predictors since next-step prediction will happen over these flattened latent variables. For decoding, there is a reshape layer that reshapes the flattened vector into a tensor of the dimensions mentioned above. The CAE decoder maps the compressed tensor back to the original SNR dimensions.

Following the discussion above, we reduced the dimensions of the SNR data from $K=2048$ to $d'_{AC}=256$. Similarly, we use PCA for dimensionality reduction over all the training trajectory SNRs. For PCA to get the same order of accuracy as auto-encoder (Table V), we need more dimensions as compared to CAE. The parameters that need to be tuned for PCA, L_{PCA}[18] $> L_{\text{AC}}$ (from Table V). Although PCA has more trainable parameters than CAE, it is easier to train because of its linear nature (e.g., using singular value decomposition). An ML-based predictor will have to predict a 512 dimensional vector for PCA and a 256 dimensional vector for auto-encoder. This difference in dimensions for predictor inputs will cause the processing intensity of PCA-based predictors to significantly increase as compared to CAE-based designs. We will show this in the next sub-section, where LSTM-AC and LSTM-PCA are compared.

B. Training ML-based predictors

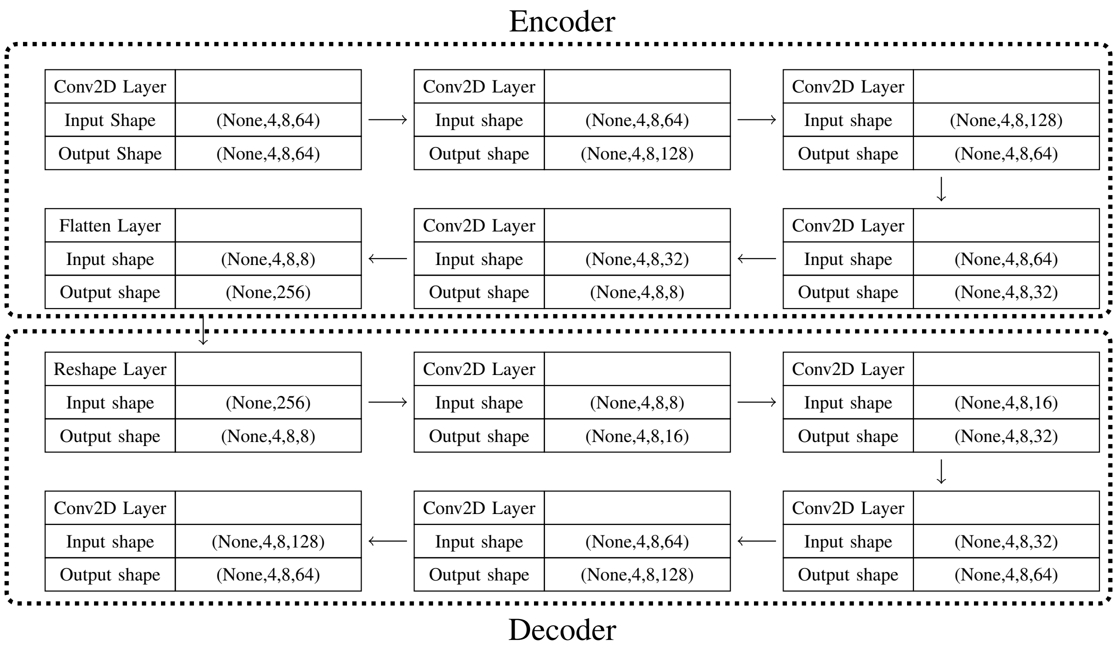
After creating an appropriate auto-encoder and PCA encoder/decoder set, we train LSTM-PCA, LSTM-AC, Vanilla RNN with AC, Transformer with AC and CNN with AC. ML-based methods are trained over 100 train trajectories

[15]We consider narrowband since primary synchronization signal (PSS) (used for estimating link quality) is narrowband. Tracking based on wideband SNR is an interesting aspect to look at in the future.

[16]Practically, this range is a function of the receiver sensitivity

[17]The CAE compresses the N_{TX} dimension because it contributes the most to the number of input dimensions.

[18]$L_{\text{PCA}}=K\times d'_{\text{PCA}}$



Encoder
Conv2D Layer
Input Shape (None,4,8,64)
Output Shape (None,4,8,64)
Conv2D Layer
Input shape (None,4,8,64)
Output shape (None,4,8,128)
Conv2D Layer
Input shape (None,4,8,128)
Output shape (None,4,8,64)
Conv2D Layer
Input shape (None,4,8,64)
Output shape (None,4,8,32)
Conv2D Layer
Input shape (None,4,8,32)
Output shape (None,4,8,8)
Flatten Layer
Input shape (None,4,8,8)
Output shape (None,256)
Reshape Layer
Input shape (None,256)
Output shape (None,4,8,8)
Conv2D Layer
Input shape (None,4,8,8)
Output shape (None,4,8,16)
Conv2D Layer
Input shape (None,4,8,16)
Output shape (None,4,8,32)
Conv2D Layer
Input shape (None,4,8,32)
Output shape (None,4,8,64)
Conv2D Layer
Input shape (None,4,8,64)
Output shape (None,4,8,128)
Conv2D Layer
Input shape (None,4,8,128)
Output shape (None,4,8,64)
Decoder

Fig. 5: Architecture of the proposed CAE. Kernel size for all the layers is 3×3. The activation is ReLU and padding is *same*. CAE is trained for 10 epochs with MSE loss.

Dimensionality reduction technique	**Total trainable parameters**	$d'_{\{\mathrm{AC,PCA}\}}$	κ	υ (dB)
Auto-encoder	377,304 (L_{AC})	256	$\frac{1}{8}$	0.4321
PCA	1,048,576 (L_{PCA})	512	$\frac{1}{4}$	0.4384

TABLE V: Comparison of reconstruction performance and dimensionality reduction of CAE and PCA.

of encoded SNRs (of dimensions d'_{AC} obtained using auto-encoder) to solve the one time step ahead prediction problem using MSE loss[19]. LSTM-PCA is trained similarly over PCA-encoded SNRs (of dimension d'_{PCA}). The hyper-parameters for all these methods are found using cross-validation. The trainable parameter count (complexity) for these methods is listed in Table VI. We see that vanilla RNN with AC is the most processing efficient, while LSTM-PCA is most expensive. As discussed in Section IV-B, this complexity is due to the larger input dimensions for LSTM-PCA. We also see that Transformer with AC has a lot of training parameters, which can be attributed to the underlying self-attention mechanism of the network. Overall, we note that RNN-based predictors – which use auto-encoders LSTM-AC and Vanilla RNN – are approximately 260% more processing efficient than LSTM-PCA. This processing efficiency justifies the use of auto-encoders for dimensionality reduction in context of the multi-cell multi-beam prediction problem.

C. Tuning parameters for baseline predictors

There is not much space for tuning for baseline predictors except for the parameter M (window size in this case). The training method for baseline predictors is to find a value of M that minimizes the losses in (3)-(4)[20]. The values of M that provide a good trade-off between the two losses are obtained by brute-force method iterating over values of M localized to $\{1, \cdots, 50\}$ for all the N_{train} trajectories. For moving average, the best window size turns out to be $M^{\text{MVA}} = 14$, while $M^{\text{LR}} = 10$ is the best for linear estimators. This tuning is done to ensure we are comparing the performances of ML-based predictors to baseline predictors, which are best (at least locally) in their own domain.

VII. Prediction Performance and Applications

A. Prediction performance metrics evaluation

In this sub-section, we present the generalization error analysis of all the predictors. As mentioned in Section I, we take a site-specific training approach, where a site comprises of a group of gNBs and a UE in a particular environment. This training enables capturing useful correlations across time and across gNBs. The networks are trained over known trajectories within the site as mentioned in Section VI. The predictor performance is measured over *new* trajectories (i.e., trajectories that the network has not seen before). This training and testing procedure is consistent with the ones widely used in the ML community. The generalization ability of the predictors is the testing of prediction performance over the new trajectories near the site. We test the predictors on N_{test} (= 100)[21] new

[19]All ML-based predictors have been trained for equal number of epochs (20) so the comparison is fair. Training times differ for each network based on their complexity.

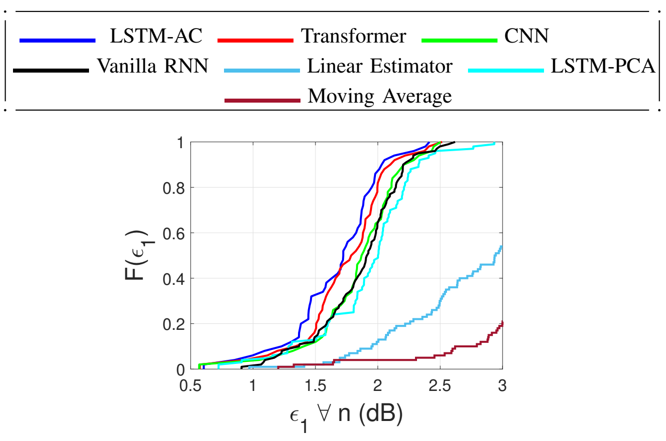
[20]Baseline predictors do not need any dimensionality reduction since they are already simple.

[21]These test trajectories are the same for auto-encoders and ML-based methods.

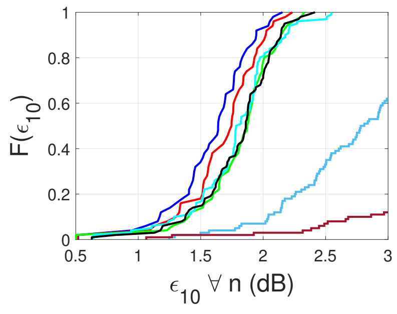
Models	**No. of parameters L**	**Models**	**No. of parameters L**
Auto-encoder (AC)	377,304	Transformer with AC ($M = 8$)	4,190,808
LSTM-AC ($M = 4$)	1,264,344	Vanilla RNN with AC ($M = 5$)	1,165,016
LSTM-PCA ($M = 9$)	4,592,128	CNN ($M = 8$)	1,755,584

TABLE VI: Comparison of complexity in terms of trainable parameters for different ML-based predictors.

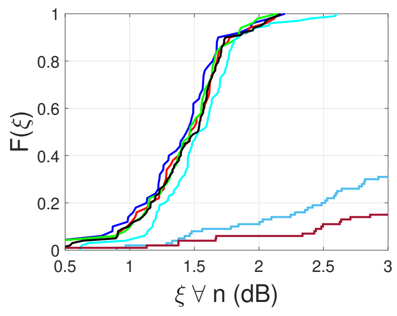
trajectories[22]. We calculate the metrics ϵ_C and ξ (from Section III) for all test trajectories (a total of N_{test} points), and for all the predictors (ML-based and baseline). The performance comparison is captured over all trajectories in form of a cumulative density function (CDF), $F(\cdot)$. These CDFs for ϵ_C and ξ for different predictors are shown in Fig. 6 and are summarized in Table VII.



(a) Top 1 link Prediction performance



(b) Top 10 link Prediction performance



(c) Misalignment performance

Fig. 6: Comparing performance of different predictors based on performance metrics from Section III

[22]Each prediction takes less than 100 μs on average for LSTM-PCA (worst case). This means prediction can be done in real-time. This computation was done on a 2018 Apple Macbook Pro i5 without any GPU support.

Methods/Performance	$P(\epsilon_1 < 2\,\text{dB})$	$P(\epsilon_{10} < 2\,\text{dB})$	$P(\xi < 2\,\text{dB})$
LSTM-AC	90%	94%	98%
Transformer with AC	80%	88%	94%
CNN with AC	70%	80%	98%
Vanilla RNN with AC	70%	72%	94%
LSTM-PCA	50%	81%	94%
Linear Estimator	12%	8%	12%
Moving Average	4%	3%	5%

TABLE VII: Performance of all the predictors based on metrics from (3) and (4). $P(y)$ means probability that event y happened.

It can be observed from Table VII and Figs. 6a and 6b that LSTM-AC has the best prediction performances among the predictors tested. For example, LSTM-AC keeps the top 1 (top 10) prediction error below $2\,\text{dB}$ 90% (94%) of the time, outperforming the transformer-based predictor by 10% (6%). This is an interesting observation because generally, transformers outperform LSTMs particularly in fields of computer vision (CV) and NLP. The better performance of LSTMs can be explained by limited training data, small number of training epochs, lack of transformer depth/width and dependence of transformer complexity on M. In [63], [64] authors show LSTMs can outperform transformers in scarcity of training data. Regarding training epochs, both architectures were trained for 20 epochs to make comparison fair. Therefore, Transformer might not have trained enough hence impacting its prediction capacity. Additionally, we consider the simplest transformer design that converged (loss reduced in training). Even the simplest transformer with AC has 4.2 million parameters (400% more than LSTM-AC). Since, one of the goals of this work is designing processing efficient networks, we did not modify the width or depth of the transformer, which can result in worse prediction performance. Moreover, complexity of transformer increases with increasing memory (M), which is not case for LSTM. This increased complexity is not justified in terms of the bias-variance trade-off (as compared to LSTM) and will result in performance degradation[23].

Comparing dimensionality reduction techniques, we see that LSTM-AC has a 40% (13%) gain in top 1 (top 10) link prediction over LSTM-PCA. This gain is due to better encoding/decoding performance of auto-encoders as compared to PCA. LSTM-AC also outperforms baseline linear predictors by 78% (86%). Overall, ML-based predictors perform better as compared to linear predictors (which is expected) and auto-encoder-based predictors perform better than the PCA-based predictor. We can also observe from Table VII and Fig. 6c that all ML-based predictors have similar misalignment error performances (between 94% to 98%) outperforming baseline linear predictors by at least 82%. This means that these ML-based predictors are able to successfully predict the beam indices within $2\,\text{dB}$ of the best beam at least 94% of

[23]These are only few reasons. The final performance evaluation is a function of data set, choice of loss functions, metrics, hyper-parameters etc.

the time. The takeaway of these analyses is that NN-based methods with auto-encoder pre-processing offer significantly better performance than standard pre-processing such as PCA or linear prediction. Among the NN-based predictors, LSTM-AC in particular provides a good processing efficiency and performance trade-off.

B. Applications

In this sub-section, we discuss some applications of multi-cell multi-beam tracking based on the network predictions. These applications are just a byproduct of the prediction problem that minimizes the loss in (2). Hence, these applications are just a subset of the core prediction problem we address in this paper. The applications themselves may be solved using a relatively simpler approach if the problem is formulated according to the application[24]. However, we look at these applications in context of the problem formulated in this work: based on the M previous measurements, predict the next time slot SNRs on all beams from all cells. The applications we discuss are *outage prediction* and *proactive beam switching*.

1) Outage prediction: As mentioned in Section I-B, a lot of work using RNNs has been done explicitly for blockage prediction purposes. Since we are predicting on all the beam-pairs from all the gNBs (using all predictors), we can predict blockages on any link. One extreme case of these blockages is outage i.e., all the available links are blocked, hence the UE/gNB goes into outage. Using this definition, we can define outage when the maximum SNR from all the links falls below a threshold $\gamma_{\text{lower}} + \Delta$ dB. Mathematically, we define a true outage event as a binary variable B given by:

$$B(t) = \mathbb{1}(\max_k \gamma_n(k,t) < \gamma_{\text{lower}} + \Delta). \tag{24}$$

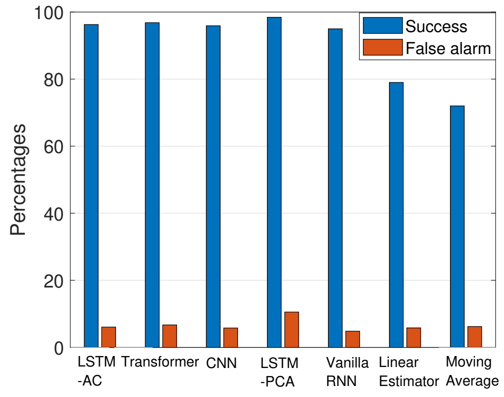
We can similarly define a predicted outage event $B^p(t)$ as:

$$B^p(t) = \mathbb{1}(\max_k \gamma_n^p(k,t) < \gamma_{\text{lower}} + \Delta). \tag{25}$$

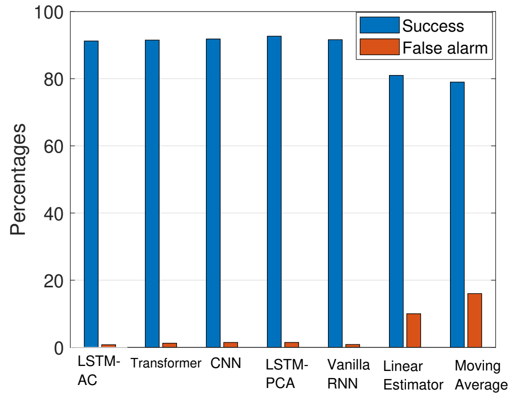
With these definitions, the following two metrics can be used to capture the outage prediction performances of different predictors:

- *Outage detection accuracy:* When $B(t) = B^p(t) = 1$, an outage is correctly predicted. Outage detection accuracy is the ratio of correctly predicted outages to the total number of true outages over all trajectories.
- *Outage false alarm ratio:* When $B(t) = 0$ and $B^p(t) = 1$, an outage is predicted when there was none. This is the ratio of falsely predicted outages to the total number of predicted outages over all trajectories.

These metrics have been shown in Fig. 7a for $\Delta = 3\,\text{dB}$. We can observe that NN-based techniques have more than 96% outage prediction accuracy as compared to 80% for the best baseline linear predictor. Higher prediction accuracy can be attributed to the ability of NN-based predictors to foresee



(a) Outage prediction comparison



(b) Proactive beam-switching performance

Fig. 7: Applications of predictors

sudden channel variations (triggering of a hand blockage event or a blockage caused by building). The false alarm ratio for all the policies except LSTM-PCA is less than 6%. Hence, the designed predictors are able to correctly detect outages 96% of the time, while also keeping the false alarm rate low. LSTM-AC in particular delivers an outage accuracy of more than 96% with a false alarm rate of around 4% and is comparably processing efficient. The outage prediction capacity of these predictors can be used for various proactive purposes e.g., a UE can turn off its radio frequency front end (RFFE) to save power when it senses an outage. Similarly, a gNB can smartly allocate resources to different UEs from the predicted outages.

2) Proactive beam switching: The predictors designed can also be used for proactive link switching. We assume that at the start of every test trajectory, UE is served by best available link (k_0). At time t, a beam/link needs to be switched if there is a better link (with greater SNR) available as compared to the serving link. For proactive link switching application, we define two events: successful proactive link switch and false proactive link switch. A *successful proactive link switch* occurs when the following three conditions are true:

$$\gamma_n(k_0,t) < \max_{\{k:k\neq k_0\}} \gamma_n(k,t) \qquad \text{:link switch needed,}$$

$$\gamma_n^p(k_0,t) < \max_{\{k:k\neq k_0\}} \gamma_n^p(k,t) \qquad \text{:link switch predicted,} \tag{26}$$

[24]For example, proactive beam switching discussed below can be formulated as a markov decision process, centered around optimizing beam switching based on some observed action and state space.

$$\gamma_n(k^p_{\text{new}}, t) > \gamma_n(k_0, t) \text{ :the new predicted link index, } k^p_{\text{new}} \text{has a better SNR.} \quad (27)$$

After a successful switch, the UE updates the best-serving link $k_0 = k^p_{\text{new}}$. Similarly, a *false proactive link* switch can be defined as:

$$\begin{aligned} \gamma_n(k_0, t) &> \max_{\{k:k\neq k_0\}} \gamma_n(k, t) \quad \text{:link switch not needed,} \\ \gamma^p_n(k_0, t) &< \max_{\{k:k\neq k_0\}} \gamma^p_n(k, t) \quad \text{:link switch predicted.} \end{aligned} \quad (28)$$

We present the results of successful and false link switch prediction percentages in Fig. 7b[25]. We see that ML-based predictors are successful in proactive beam switching 91% of the time as compared to linear estimator and moving average at around 80%. However, looking at the false alarm percentage, we see that the false alarm percentage of ML-based predictors is less than 2% as compared to 10% (16%) to that of the linear estimator (moving average), meaning LSTM predictors not only accurately predict the beam switching in advance but also keep the false beam switch rate low. LSTM-AC and Vanilla RNN will be preferred in this case because of their low complexities. The superior performance of ML-based predictors can be explained by the ability of the neural networks to predict multiple beams from multiple cells more accurately as compared to baseline linear predictors. This proactive beam switching can be translated into handovers if the beams switched are from different gNBs[26].

VIII. Summary and Future Work

Beam tracking is a fundamental challenge in all mmWave systems. In this work, we have proposed an auto-encoder integrated LSTM network for multi-cell multi-beam prediction. Auto-encoders reduce input dimensionality of the predictor – a major problem in multi-cell multi-beam tracking scenarios – enabling processing efficient design of accurate LSTM predictors. Notably, the method can track signals from multiple cell sites and is applicable for procedures including handover and carrier aggregation with multiple cells. The method was validated on detailed ray tracing measurements. There is significant opportunity to build on this work. Most importantly, we have looked at narrowband measurements similar to what is obtained with reference signals in 5G NR. A key research direction is to predict the wideband channel characteristics from intermittent narrowband measurements. A second avenue of future research is to validate the work on larger training data sets. We have already accumulated ray tracing on five large cities in our work [30] and a similar campaign can be used here. Another direction for the future is tightening the assumption from "UE is able to measure all the links to predict all the links" to "UE is able to measure a subset of links to predict all the links". This new assumption gives rise to a new problem, which is finding how to choose the subset of links from which the predictors can extrapolate the link qualities of all the links.

[25]Successful link switch percentage is the ratio of the sum of successful link switches to total number of link switches needed. False alarm switch percentage is the ratio of the sum of false link switches to total number of link switches predicted.

[26]In general, handovers are much more expensive than beam switching, but if there is exchange of information about UEs within the gNBs, handovers can be handled in a similar manner to beam switching.

References

[1] S. H. A. Shah, M. Sharma, and S. Rangan, "LSTM-based multi-link prediction for mmWave and sub-THz wireless systems," in *IEEE ICC*, June 2020, pp. 1–6.

[2] T. S. Rappaport *et al.*, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, May 2013.

[3] ——, *Millimeter Wave Wireless Communications*. Pearson Education, 2014.

[4] C. Slezak *et al.*, "Empirical effects of dynamic human-body blockage in 60 GHz communications," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 60–66, December 2018.

[5] T. Bai and R. W. Heath, "Coverage analysis for millimeter wave cellular networks with blockage effects," in *Proc. IEEE Global Conf. on Signal and Information Processing*, February 2013, pp. 727–730.

[6] V. Raghavan *et al.*, "Spatio-temporal impact of hand and body blockage for millimeter-wave user equipment design at 28 GHz," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 46–52, December 2018.

[7] J. Choi, "On the macro diversity with multiple BSs to mitigate blockage in millimeter-wave communications," *IEEE Commun. Lett.*, vol. 18, no. 9, pp. 1653–1656, Jul. 2014.

[8] G. Yuan *et al.*, "Carrier aggregation for LTE-advanced mobile communication systems," *IEEE Commun. Mag.*, vol. 48, no. 2, pp. 88–93, Jan. 2010.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, December 1997.

[10] C. E. Zachary C. Lipton, John Berkowitz, "A critical review of recurrent neural networks for sequence learning," *preprint arXiv:1506.00019*.

[11] 3GPP, "TS 38.300, NR and NG-RAN overall description; stage 2," Apr. 2020.

[12] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. of the IEEE*, vol. 107, no. 8, pp. 1655–1674, July 2019.

[13] J. Joo *et al.*, "Deep learning-based channel prediction in realistic vehicular communications," *IEEE Access*, vol. 7, pp. 27 846–27 858, February 2019.

[14] J. D. Herath *et al.*, "A deep learning model for wireless channel quality prediction," in *IEEE ICC*, May 2019, pp. 1–6.

[15] C. Luo *et al.*, "Channel state information prediction for 5G wireless communications: A deep learning approach," *IEEE Trans. on Network Science and Engineering*, pp. 1–1, June 2018.

[16] W. Jiang and H. D. Schotten, "Neural network-based fading channel prediction: A comprehensive overview," *IEEE Access*, vol. 7, pp. 118 112–118 124, August 2019.

[17] A. Kulkarni *et al.*, "Deepchannel: Wireless channel quality prediction using deep learning," *IEEE Trans. on Vehicular Techno.*, vol. 69, no. 1, pp. 443–456, October 2020.

[18] H. Yan *et al.*, "mmRAPID: Machine learning assisted noncoherent compressive millimeter-wave beam alignment," in *Proceedings of the 4th ACM Workshop on Millimeter-Wave Networks and Sensing Systems*, July 2020, pp. 1–6.

[19] S. H. A. Shah *et al.*, "Power-efficient beam tracking during connected mode DRX in mmWave and sub-THz systems," *preprint TechRxiv:13046447*, Oct 2020.

[20] S. H. Ali Shah *et al.*, "Power efficient discontinuous reception in THz and mmWave wireless systems," in *Proc. IEEE Intl. Workshop on Process. Adv. in Wireless Commun. (SPAWC)*, Jul. 2019, pp. 1–5.

[21] N. Al-Falahy and O. Alani, "Millimetre wave frequency band as a candidate spectrum for 5G network architecture: A survey," *Physical Commun.*, vol. 32, Nov 2018.

[22] M. Alrabeiah and A. Alkhateeb, "Deep learning for mmwave beam and blockage prediction using sub-6 GHz channels," *IEEE Trans. on Commun.*, vol. 68, no. 9, pp. 5504–5518, June 2020.

[23] F. Göttsch and M. Kaneko, "Deep learning-based beamforming and blockage prediction for sub-6GHz/mm wave mobile networks," in *IEEE GLOBECOM 2020*, January 2020, pp. 1–6.

[24] A. Alkhateeb *et al.*, "Deep learning coordinated beamforming for highly-mobile millimeter wave systems," *IEEE Access*, vol. 6, pp. 37 328–37 348, June 2018.

[25] M. Zarifneshat *et al.*, "Learning-based blockage prediction for robust links in dynamic millimeter wave networks," in *16th Annual IEEE Intl. Conf. on Sensing, Commun., and Netw. (SECON)*, September 2019, pp. 1–9.

[26] H. Liu *et al.*, "DNN-based beam and blockage prediction in 3GPP InH scenario," in *2020 Intl. Conf. on Information and Commun. Techno. Convergence (ICTC)*, December 2020, pp. 320–325.

[27] A. Alkhateeb *et al.*, "Machine learning for reliable mmwave systems: Blockage prediction and proactive handoff," *Proc. IEEE Global Conf. on Signal and Information Processing (GlobalSIP)*, pp. 1055–1059, February 2018.

[28] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," *arXiv preprint arXiv:1902.06435*, February 2019.

[29] E. Balevi *et al.*, "High dimensional channel estimation using deep generative networks," *IEEE J. Sel. Areas in Commun.*, vol. 39, no. 1, pp. 18–30, January 2020.

[30] W. Xia *et al.*, "Millimeter wave channel modeling via generative neural networks," *arXiv preprint arXiv:2008.11006*, August 2020.

[31] E. Dahlman *et al.*, *5G NR: The next generation wireless access technology*. Academic Press, 2020.

[32] M. Shafi *et al.*, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Apr. 2017.

[33] M. Giordani *et al.*, "A tutorial on beam management for 3GPP NR at mmWave frequencies," *IEEE Commun. Surveys Tuts.*, Sep. 2018.

[34] T. Bai *et al.*, "Coverage and capacity of millimeter-wave cellular networks," *IEEE Commun. Mag.*, vol. 52, no. 9, pp. 70–77, Sep. 2014.

[35] M. R. Akdeniz *et al.*, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1164–1179, Jun. 2014.

[36] 3GPP, "Requirements for support of radio resource management ," 3GPP TS 38.133 Release 15), Jul. 2019.

[37] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*, 1st ed. Springer Publishing Company, 2011.

[38] N. Abuzainab *et al.*, "Deep learning for THz drones with flying intelligent surfaces: Beam and handoff prediction," *preprint arXiv:2102.11222*, Feb 2021.

[39] W. Sun *et al.*, "LSTM based link quality confidence interval boundary prediction for wireless communication in smart grid," *Computing*, vol. 103, Feb 2021.

[40] X. Xiao and S. Zarar, "Packet loss concealment with recurrent neural networks for wireless inertial pose tracking," in *IEEE 15th Intl. Conf. on Wearable and Implantable Body Sensor Networks (BSN)*, April 2018, pp. 25–29.

[41] H. Cheng *et al.*, "Data prediction model in wireless sensor networks based on bidirectional lstm," *EURASIP J. on Wireless Commun. and Networking*, vol. 2019, Aug 2019.

[42] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," *Proceedings of the IEEE-INNS-ENNS Intl. Joint Conference on Neural Networks. IJCNN. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3, pp. 189–194 vol.3, July 2000.

[43] I. Goodfellow *et al.*, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[44] D. Charte *et al.*, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, January 2018.

[45] Remcom, "Wireless insite." [Online]. Available: http://www.remcom.com/wireless-insite

[46] X. Yang and Y. Lu, "Propagation characteristics of millimeter wave in circular tunnels," in *2006 7th Intl. Symposium on Antennas, Propagation EM Theory*, April 2006, pp. 1–5.

[47] Q. Li *et al.*, "Validation of a geometry-based statistical mmwave channel model using ray-tracing simulation," in *IEEE 81st Vehicular Technol. Conf. (VTC Spring)*, July 2015, pp. 1–5.

[48] S. Wu *et al.*, "Intra-cluster characteristics of 28 GHz wireless channel in urban micro street canyon," in *IEEE GLOBECOM*, February 2016, pp. 1–6.

[49] M. Alrabeiah and A. Alkhateeb, "Deep learning for TDD and FDD massive MIMO: Mapping channels in space and frequency," in *2019 53rd Asilomar Conf. on Signals, Systems, and Computers*, August 2019, pp. 1465–1470.

[50] A. Taha *et al.*, "Enabling large intelligent surfaces with compressive sensing and deep learning," *IEEE Access*, vol. 9, pp. 44 304–44 321, March 2021.

[51] 3GPP, "TR 38.901, Study on Channel Model for Frequencies From 0.5 to 100 GHz (Release 15) document," Jun. 2018.

[52] J. Jarvelainen *et al.*, "Indoor propagation channel simulations at 60 GHz using point cloud data," *IEEE Trans. on Antennas and Propagation*, vol. 64, pp. 4457–4467, August 2016.

[53] S. Ju *et al.*, "Scattering mechanisms and modeling for terahertz wireless communications," in *IEEE ICC*, May. 2019, pp. 1–7.

[54] MATLAB, "Navigation toolbox." [Online]. Available: https://in.mathworks.com/products/navigation.html

[55] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The Intl. J. of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001. [Online]. Available: https://doi.org/10.1177/02783640122067453

[56] V. Raghavan *et al.*, "Statistical blockage modeling and robustness of beamforming in millimeter-wave systems," *IEEE Trans. on Microwave Theory and Techniques*, vol. 67, no. 7, pp. 3010–3024, July 2019.

[57] V. Raghavan *et al.*, "Millimeter-wave MIMO prototype: Measurements and experimental results," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 202–209, Jan. 2018.

[58] M. R. Akdeniz *et al.*, "Millimeter wave picocellular system evaluation for urban deployments," in *2013 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2013, pp. 105–110.

[59] 3GPP, "TR 38.214,, NR - Physical layer procedures for data - (release 15) document," Jun. 2018.

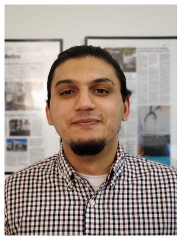
[60] ——, "TR 38.900, study on channel model for frequency spectrum above 6 GHz release," Jun. 2018.

[61] P. Skrimponis *et al.*, "Power consumption analysis for mobile mmwave and sub-THz receivers," in *2020 IEEE 6G Wireless Summit*, Mar. 2020, pp. 1–5.

[62] 3GPP, "TS 38.214 E-UTRA - 5G NR physical layer procedures for data - Release 15," Jan. 2020.

[63] A. Ezen-Can, "A Comparison of LSTM and BERT for Small Corpus," September 2020. [Online]. Available: https://arxiv.org/abs/2009.05451

[64] K. Tran *et al.*, "The importance of being recurrent for modeling hierarchical structure," in *Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 4731–4736. [Online]. Available: https://aclanthology.org/D18-1503



Syed Hashim Ali Shah (S'19) received the B.S. degree in electrical engineering from the Lahore University of Management Sciences (LUMS), Lahore, Pakistan, in 2017. He is currently pursuing the Ph.D. degree with the Tandon School of Engineering, New York University, Brooklyn, NY, USA. His research interests are wireless system design (mmWave and sub-THz) and machine learning for wireless communications.

Sundeep Rangan (S'94−M'98−SM'13−F'16) received the B.A.Sc. at the University of Waterloo, Canada and the M.Sc. and Ph.D. at the University of California, Berkeley, all in Electrical Engineering. He has held postdoctoral appointments at the University of Michigan, Ann Arbor and Bell Labs. In 2000, he co-founded (with four others) Flarion Technologies, a spin-off of Bell Labs, that developed Flash OFDM, the first cellular OFDM data system and pre-cursor to 4G cellular systems including LTE and WiMAX. In 2006, Flarion was acquired by Qualcomm Technologies. Dr. Rangan was a Senior Director of Engineering at Qualcomm involved in OFDM infrastructure products. He joined NYU Tandon (formerly NYU Polytechnic) in 2010 where he is currently a Professor of Electrical and Computer Engineering. He is a Fellow of the IEEE and the Associate Director of NYU WIRELESS, an industry-academic research center on next-generation wireless systems.