Scalable Multi-Modal Learning for Cross-Link Channel Prediction in Massive IoT Networks

Kun Woo Cho[1], Marco Cominelli[2], Francesco Gringoli[2], Joerg Widmer[3], Kyle Jamieson[1]
Princeton University[1], University of Brescia/CNIT[2], IMDEA Networks[3]

ABSTRACT

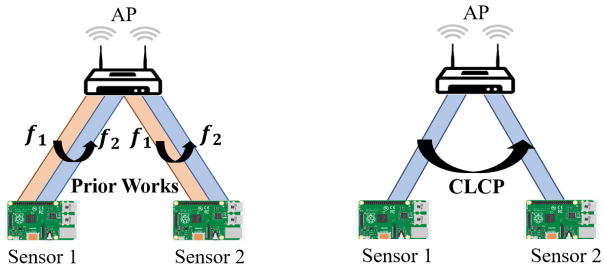
Tomorrow's massive-scale IoT sensor networks are poised to drive uplink traffic demand, especially in areas of dense deployment. To meet this demand, however, network designers leverage tools that often require accurate estimates of Channel State Information (CSI), which incurs a high overhead and thus reduces network throughput. Furthermore, the overhead generally scales with the number of clients, and so is of special concern in such massive IoT sensor networks. While prior work has used transmissions over one frequency band to predict the channel of another frequency band on the same link, this paper takes the next step in the effort to reduce CSI overhead: predict the CSI of a nearby but distinct link. We propose ***Cross-Link Channel Prediction*** (CLCP), a technique that leverages multi-view representation learning to predict the channel response of a large number of users, thereby reducing channel estimation overhead further than previously possible. CLCP's design is highly practical, exploiting existing transmissions rather than dedicated channel sounding or extra pilot signals. We have implemented CLCP for two different Wi-Fi versions, namely 802.11n and 802.11ax, the latter being the leading candidate for future IoT networks. We evaluate CLCP in two large-scale indoor scenarios involving both line-of-sight and non-line-of-sight transmissions with up to 144 different 802.11ax users and four different channel bandwidths, from 20 MHz up to 160 MHz. Our results show that CLCP provides a 2× throughput gain over baseline and a 30% throughput gain over existing prediction algorithms.

CCS CONCEPTS

• **Networks** → **Wireless access networks**; • **Computing methodologies** → **Artificial intelligence**.

KEYWORDS

Channel prediction, multi-modal learning, massive-IoT networks



(a) Cross-band channel pred. **(b)** Cross-link channel pred.

Fig. 1— ***Left:*** Previous work [2, 17] on cross-band channel prediction infers a downlink channel at frequency f_2 using the uplink channel at frequency f_1 on the *same* link. ***Right:*** CLCP infers the channel to Sensor 2 using channel measurements from Sensor 1.

1 INTRODUCTION

Today's wireless IoT sensor networks are changing, scaling up in spectral efficiency, radio count, and traffic volume as never seen before. There are many compelling examples: sensors in smart agriculture, warehouses, and smart-city contexts collect and transmit massive amounts of aggregate data, around the clock. Networks of video cameras (*e.g.*, for surveillance and in cashierless stores) demand large amounts of uplink traffic in a more spatially-concentrated pattern. And in multiple rooms of the home, smart cameras, speakers, and kitchen appliances stream their data continuously.

This sampling of the newest Internet of Things (IoT) applications highlights unprecedented demand for massive IoT device scale, together with ever-increasing data rates. Sending and receiving data to these devices benefits from advanced techniques such as Massive Multi-User MIMO (MU-MIMO) and OFDMA-based channel allocation. The 802.11ax [3] Wi-Fi standard, also known as *Wi-Fi 6*, uses both these techniques for efficient transmission of large numbers of small frames, a good fit for IoT applications. In particular, OFDMA divides the frequency bandwidth into multiple subchannels, allowing simultaneous multi-user transmission.

While such techniques achieve high spectral efficiency, they face a key challenge: they require estimates of channel state information (CSI), a process that hampers overall spectral efficiency. Measuring and propagating CSI, in fact, scales with the product of the number of users, frequency bandwidth, antenna count, and frequency of measurement. Highly-dynamic environments with human and vehicle mobility further exacerbates these challenges, necessitating more frequent CSI measurement. With densely deployed IoT devices, the overhead of collecting CSI from all devices may thus deplete available radio resources [21]. While compressing CSI [13, 21] and/or leveraging channel reciprocity for implicit channel sounding [2, 17] reduces CSI overhead to some degree, users still need to exchange compressed CSI with the Access Point (AP) [21], and implicit sounding relies on extremely regular traffic patterns [2, 17],

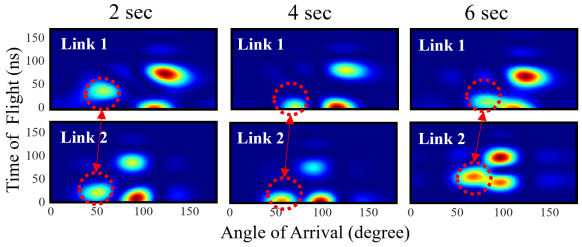


Fig. 2— *CLCP's mechanism:* Time of flight and angle of arrival for two nearby IoT sensors (***upper*** and ***lower***, respectively). While each sensor (link) has a distinct set of static wireless paths, their parameters both indicate reflections off the same moving object, highlighted in red dotted circles.

and so with increasing numbers of clients, antennas, and OFDM subcarriers, CSI overhead remains a significant burden.

In this paper, we take a qualitatively different approach. While conventional wisdom holds that the channels of the nodes that are at least half a wavelength apart are independent due to link-specific signal propagation paths [16], with enough background data and measurements of a wireless environment, we find that it is possible to predict the CSI of a link that has not been recently observed. Fig. 1 illustrates our high-level idea: unlike previous works [2, 17] that use CSI measurements at frequency f_1 to infer CSI at f_2 for a single link, our approach exploits the cross-correlation between different links' wireless channels to leverage traffic on one sensor's link to predict the wireless channel of another.

To support our idea, we measure the wireless channels from two nearby sensors that are roughly 30 cm apart in the presence of a moving human. Figure 2 visualizes these channels using two wireless path parameters, Time-of-Flight (ToF) and Angle-of-Arrival (AoA)[1]. While Sensor 1's channel (*upper*) is independent from Sensor 2's channel (*lower*), ToF and AoA from both reflect the same moving body in the environment, indicated by the dotted red circles, while other major paths remain unchanged. This suggests the existence of a function that correlates the occurrence of wireless links of stationary sensors in the presence of moving reflectors.

We propose the *Cross-Link Channel Prediction* (CLCP) method, a wireless channel prediction technique that uses multiview representation machine learning to realize this vision. In summary, a CLCP AP uses uplink channels observed from the last transmission to predict a large number of unobserved wireless links. Here, the AP treats each channel reading like a photo of an environment taken at a particular *viewpoint*, and combine multiple different views to form a joint representation of the environment. It then exploits this representation to predict unobserved wireless CSI readings. Since CLCP has no dedicated channel sounding or extra pilot signal, the aggregated overhead no longer scales with the number of radios. However, since traffic patterns are not regular in reality, the number of channels and links observed at the time of prediction is likely to change. Such dynamics in input data often lead to an explosion in the number of trainable parameters, making the learning process intractable [20, 22]. CLCP hence adopts a special model design (§3.1.3) and training paradigm (§3.1.6) that allows it to construct a joint representation from any combination of multiple views. Moreover, we often treat deep learning models as *black boxes* whose inner workings cannot be interpreted. As a result, designers cannot differentiate whether the trained model truly works or is simply overfitting. To understand and validate CLCP's learning mechanism, we visualize a fully trained feature representation and interpret it using the wireless path parameters, ToF and AoA (§3.2).

Our implementation and experimental evaluation using 802.11ax validate the effectiveness of CLCP through microbenchmarks and head-to-head performance comparison against OptML [2] and R2F2 [17] cross-band channel prediction methods (§5). CLCP provides a 2× throughput gain over baseline 802.11ax and a 30% throughput gain over R2F2 and OptML in a 144-link testbed. Moreover, by eliminating channel sounding, it increases sleep time by 65%.

[1]We note that the spectrum representing Link 2 at 6-sec abruptly changes due to a carrier frequency offset resulting from non-synchronized local oscillators. This figure shows the raw observation without any error correction.

2 PRIMER: ML BACKGROUND FOR CLCP

In the context of learning, cross-band channel prediction [2, 17] is a markedly different problem than cross-link channel prediction. In the former, uplink and downlink channels share exactly the same paths in the wireless channel. Therefore, the learning task is simply to map the (complicated) effect of changing from one frequency band to another, given a fixed set of wireless paths. For CLCP, the channels of nearby radios have distinct paths and the the learning task is to elucidate the correlations between the two links.

To correlate the occurrence of distinct wireless links given they share *some* views on the wireless environment, CLCP must **1)** discard radio-specific information from raw observations and extract a feature representation that conveys information on moving reflectors; **2)** integrate the extracted representation with radio-specific properties (*e.g.,* signal paths and noises) to synthesize unobserved channels. However, radio-specific and environment-specific information superimpose in channel readings and thus are not easily separable. We exploit representation learning to achieve these tasks. Specifically, an *encoder* network of our model accomplishes the first task, and a *decoder* network achieves the second task. Before we explain our design in detail, we first provide the background on representation learning.

Variational Autoencoder (VAE). The goal of VAE [10] is to compress input data into a low-dimensional feature while preserving maximum information about the original data. In a nutshell, its *encoder* neural network compresses the input data into a representation z, also known as a *latent variable*, and its *decoder* decompresses z back to the original input. Its loss function, known as an *evidence lower bound* (ELBO), consists of a *reconstruction term* and a *regularization term*. The reconstruction term ensures that the decoder output resembles the encoder input, while the regularization term encourages the latent variable to follow a normal, Gaussian distribution, preventing overfitting and enhancing generalizability beyond the training set [1, 12]. By minimizing this loss, the VAE finds a distribution that best describes the data.

Multiview Representation Learning. *Multiview (multimodal) representation learning* [11, 14, 20] has proven effective in capturing the correlation relationships of information that comes as different *modalities* (distinct data types or data sources). For instance,

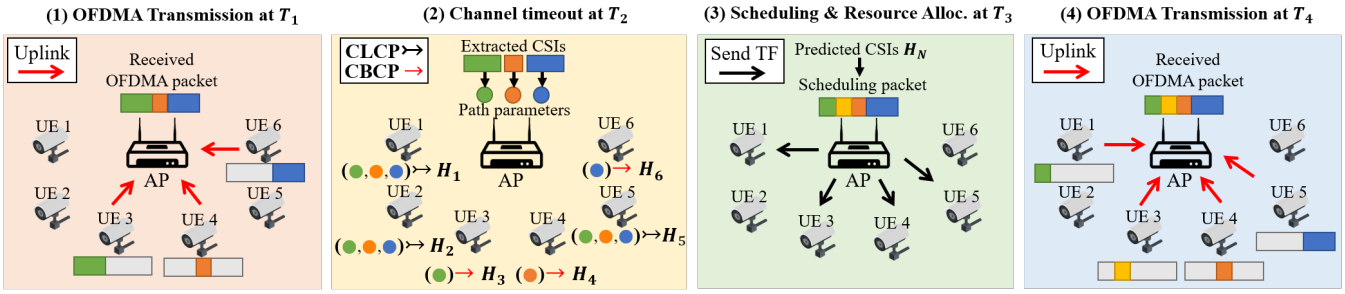


Fig. 3— *System overview for uplink transmission:* (1) an AP receives uplink traffic from multiple users simultaneously. (2) when channels become outdated, the AP extracts the path parameters of partial CSIs estimated from the latest OFDMA packet and predicts unobserved links using CLCP and unobserved bands using CBCP in a server; (3) Then, the AP schedules uplink traffic based on predicted CSIs and triggers the users. (4) Finally, the AP receives the scheduled packet.

different sets of photos of faces, each set having been taken at different angles, could each be considered different modalities. This model learns correlations between different modalities and represents them jointly, such that the model can generate a (missing) instance of one modality given the others. Like VAE, it encodes a primary view into a low-dimensional feature that contains useful information about a scene. However, instead of simply reconstructing the input, multi-view learning decodes this latent variable, which describes the scene, into a *secondary view*.

A more advanced form of multiview learning adopts multiple different views as input data and encodes them into a joint representation. By analyzing multiple information sources simultaneously, we learn a better, comprehensive feature representation. Past works [15] use multiview learning to synthesize unseen images at an arbitrary angle given the images of a scene taken at various angles. Likewise, we treat each wireless link like a photo of a scene taken from a particular viewpoint. We observe wireless links from various viewpoints and combine them to form a joint representation of the channel environment. We then exploit this joint representation to predict the link at unobserved viewpoints.

3 DESIGN

Our system operates in a sequence of steps shown in Fig. 3: to begin with, an AP acquires *channel state information* (CSI) from all users, which it then uses to schedule an uplink OFDMA packet. Once scheduled, the AP triggers the uplink transmission. When the acquired CSIs become outdated, the AP estimates the channels from the most recently received OFDMA packet and uses them to predict (1) the remaining bands of estimated channels using *cross-band channel prediction* (CBCP) and (2) full-band CSIs of unobserved links using *cross-link channel prediction* (CLCP). The AP then schedules the uplink packet using the predicted CSIs and sends a trigger frame (TF) to re-initiate the uplink transmission.

(1) Opportunistic Channel Observation. OFDMA divides the entire bandwidth into multiple subchannels known as a resource unit (RU). The AP assigns RUs to individual users, enabling one OFDMA packet to contain channel estimates from multiple users. Our goal is to leverage multiple subchannels already present in existing OFDMA transmissions to predict the channels of *other* users. In Fig. 3, user 3, 4, and 6 simultaneously transmit uplink signals in their dedicated RUs, which adds up to a full bandwidth channel. After the acquired CSIs expire, the AP extracts the subchannels from these users. It then uses this information to predict the remaining channels of these users *and* the whole channels of unobserved users 1, 2, and 5. This way, we completely eliminate the need for channel sounding.

(2) Channel Prediction. Once the AP estimates the subchannels, it directly routes the channels to a backend server through an Ethernet connection. At the server-side, path parameters are extracted from observed channel estimates and are then fed to CLCP for channel prediction (§3.1).

(3) Scheduling and Resource Allocation (SRA). The AP schedules the upcoming OFDMA transmission using predicted CSIs (§3.3). It is worth noting that OFDMA scheduling requires full-bandwidth channels in order to allocate a valid combination of RUs with varying subcarrier sizes [18] and to find a proper modulation and coding scheme (MCS) index for each RU. Moreover, unlike 11n and 11ac, 11ax provides support for uplink MU-MIMO, which requires CSI to find an optimal set of users with low spatial channel correlation and appropriate decoding precedence. After scheduling an uplink packet, the AP encloses these information in a *trigger frame* (TF) and broadcasts the frame.

(4) Uplink data transmission. After receiving the TF, the corresponding users transmit data accordingly.

3.1 CLCP Model Design

Our CLCP ML model is summarized in Fig. 4: there is a *single-view encoder* network $q_\phi(z|h)$ (§3.1.2) dedicated to each single *view* (*i.e.*, channel h of each radio). Every encoder outputs the distribution parameters, μ and σ, and a *multi-view combiner* (§3.1.3) fuses the parameters from all encoders into a joint representation z. If the channel is not observed, we drop its respective encoder network (*e.g.* E_2 in Fig. 4). A decoder network $p_\theta(h|z)$ (§3.1.4), dedicated to each *target* radio whose CSI we seek to synthesize, samples the joint representation z to predict a *cross-link* channel.

A key challenge is that across different prediction instances, the combination of observed channels varies, as we utilize channels from existing OFDMA transmission. In Fig. 4, the input channels consist of two RUs, one from Sensor 1 and the other from Sensor N. In the next prediction instance, the OFDMA packet is likely to contain a different RU combination, possibly from different radios, resulting in inconsistencies that complicate the learning process.

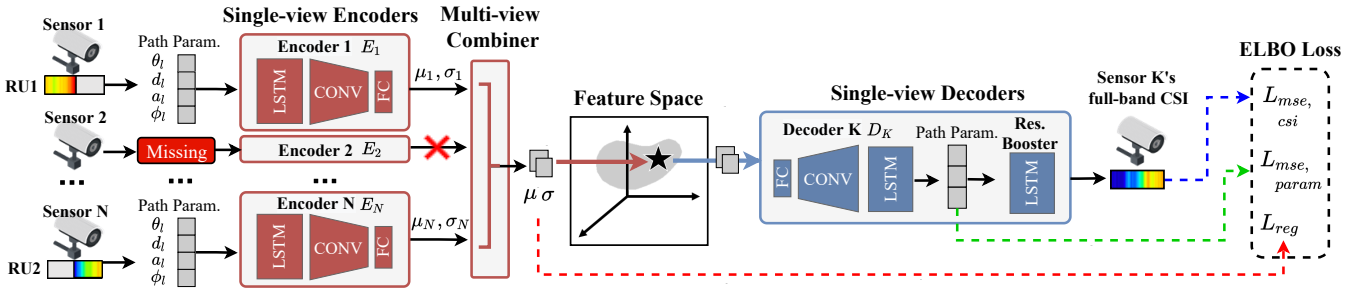


Fig. 4— CLCP ML model with N measured channels, each represented as a set of wireless path parameters $\{\theta_l, d_l, a_l, \phi_l\}_{l=0}^{L}$ with L paths estimated from measured channels. Each set of the parameters is served by a **Single-view Encoder** network E_i ($i \in [1, N]$) that compresses the measured wireless path information of its dedicated radio and outputs variational parameters μ_i and σ_i. The **Multi-view Combiner** integrates all variational parameters into μ and σ, based on which **Single-view Decoder** networks D_K generate a set of path parameters that are unobserved. If any input channel is not observed, CLCP drops the respective encoder network (E_2, for example).

We will address how CLCP is robust against observations that vary in frequency (§3.1.1) and link (§3.1.3).

3.1.1 Path Parameter Estimator. To reduce the learning complexity of CLCP, we extract the geometric information *i.e.,* wireless path parameters, from raw CSIs and use them directly as input data. This approach is particularly effective because the path parameters are *frequency-independent*, which means that CLCP is robust against observations from different frequency bands. We can represent the channel h observed at antenna M_i as a sum of L paths[2], where each path is characterized by an arrival angle θ_l, a time delay d_l, an attenuation a_l, and a reflection ϕ_l:

$$h_{M_i,\lambda} = \sum_{l}^{L} (a_l e^{\frac{-j2\pi d_l}{\lambda} + j\phi_l}) e^{\frac{-j2\pi i k \cos(\theta_l)}{\lambda}} \tag{1}$$

where λ and k are wavelength and antenna distance. To extract the 4-tuple of parameters $\{(\theta_l, d_l, a_l, \phi_l)\}_{l=0}^{L}$, we use maximum likelihood estimation. For simplicity, we now denote the 4-tuple as $\ddot{h}$.

3.1.2 CLCP's Single-View Encoder. Like VAE, a single-view encoder $q_\phi(z_n|\ddot{h}_n)$ compresses its dedicated channel $\ddot{h}_n$ into Gaussian distribution parameters, μ_n and σ_n. We can then compute the distribution that best describes its dedicated viewpoint as $z_n = \mu_n + \sigma_n \odot \epsilon$ where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. Each of the encoders consists of the long short-term memory layer (LSTM) followed by two-layer stacked convolutional layers (CNN) and fully connected (FC) layers. In each layer of CNNs, 1D kernels are used as the filters, followed by a batch norm layer. We learn all layer weights of the encoders and decoders end-to-end through backpropagation. If an encoder has no input due to an unobserved link (*e.g.,* Sensor 2), CLCP drops the respective encoder networks (E_2 in Fig. 4).

3.1.3 CLCP's Multi-view Combiner. A naïve approach to handle a varying number of inputs is to train an encoder network for each combination of input. However, this approach leads to a significant increase in the number of trainable parameters, making it computationally intractable for complex multi-view datasets. Instead, CLCP constructs one encoder per each user and efficiently fuses the output of all encoders into a joint representation. We model this multiview combiner after the *product-of-experts* (PoE) [6, 20] whose core idea is to combine several probability distributions (*experts*), by multiplying their density functions. With any combination of input channels $\mathbf{H} = \{\ \ddot{h}_i \mid \text{channel of } i^{\text{th}} \text{ radio}\ \}$, we formulate the multiview combiner as:

$$q_\phi(z|\mathbf{H}) \propto p(z) \prod_{\ddot{h}_n \in \ddot{H}} \widetilde{q}(z_n|\ddot{h}_n) \tag{2}$$

where $p(z)$ is a Gaussian distribution, and $\widetilde{q}(z_n|\ddot{h}_n)$ is an encoder network dedicated to n^{th} radio. Here, the multiview combiner approximates the joint representation z by multiplying all individual distributions z_n, parameterized by the encoder outputs, μ_n and σ_n.

PoE assumes that multi-view inputs are conditionally independent given the distribution z, which represents dynamics in the wireless environment. The assumption holds because the signal paths of different radios experience independent fading, and allows for the factorization of the encoders in CLCP. With this factorization, we can simply ignore the encoder with no input channel.

3.1.4 CLCP's Single-View Decoder. Our single-view decoder $p_\theta(\ddot{h}_n|z)$ is another DNN, whose input is the joint representation z. The goal of each decoder is to synthesize an accurate channel estimate of its dedicated radio. The decoder architecture is in the exact opposite order of the encoder architecture. The decoder predicts the path parameters of a target radio, which are in turn used to construct a full-band channel based on Eq. (1). In practice, estimating path parameters induces some loss of information. To compensate for this loss, we add an extra neural network layer, a *resolution booster* that generates a final channel estimate h_{pred}.[3]

3.1.5 Objective function. We define our loss function as:

$$\text{ELBO} = \mathbb{E}_{q_\phi(z|\mathbf{H})}\left[\log\ p_\theta(\ddot{h}|z)\right] - \beta D_{\text{KL}}(q_\phi(z|\mathbf{H})||p(z)) \tag{3}$$

where β is a weight for balancing two terms in the ELBO. Like VAE (§2), the second term represents the regularization loss (L_{reg}) that makes the joint distribution z close to a Gaussian distribution $p(z)$. For VAE, the first term represents the reconstruction error. However, instead of simply reconstructing the input, our decoders must synthesize unobserved channels. Hence, our first term consists of a mean squared error between the predicted channel

[2]The number of distinguishable paths is constrained by the numbers of transmitting antennas Mt and receiving antennas Mr. Consequently, we define the number of paths L as min(Mt, Mr).

[3]In the case of adding a new node, once CLCP's state is already constructed, only one decoder needs to be added. Thus, CLCP does not require a fresh training start.

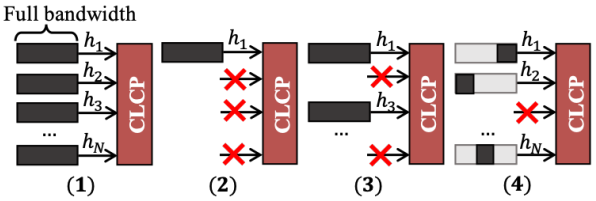


Fig. 5— Multi-step training paradigm.

h_{pred} and ground-truth channel h_{gt} of a target radio: $L_{mse,csi} = \frac{1}{S}\sum_{s=0}^{S}(\|h_{s,gt} - h_{s,pred}\|_2)$ where S is the number of subcarriers. We also include a mean squared error between the predicted and ground-truth path parameters as a part of the first term. Since some paths are stronger than others when superimposed, we weigh the error of each path based on its amplitude a as follow: $L_{mse,param} = \sum_{l=0}^{L}(a_l \|\ddot{h}_{l,gt} - \ddot{h}_{l,pred}\|_2)$. Finally, our first term becomes $L_{mse} = -(\alpha L_{mse,csi} + \eta L_{mse,param})$ where α and η are the weight terms. By maximizing ELBO, we maximize the lower bound of the probability of generating a comprehensive representation of the wireless environment and predicting accurate channels.

3.1.6 Multi-stage training paradigm. If we train all encoder networks altogether, then CLCP cannot generate an accurate prediction when some links are unobserved at the test time. Conversely, if we individually train each encoder network, CLCP fails to capture the relationship across different links. To address this issue, our loss function consists of four ELBOs (see Fig. 5): (1) one from feeding all N full-band channels, (2) the sum of N ELBO terms from feeding each full-band channel at a time, and (3) the sum of k ELBO terms from feeding k randomly chosen subsets of full-band channels (where k is randomly chosen between 2 and N). Lastly, (4) we repeat the first three steps with a random subchannel to mimic channels in the actual OFDMA transmission. We then back-propagate the sum of four ELBOs end-to-end.

3.2 Model interpretability

The proximity of low-dimensional features in the feature space indicates their relevance for a given task. For instance, if we encode two radios' channels into our low-dimensional features and they appear close in the feature space, it suggests that both channels have been affected by the same moving reflectors simultaneously. To understand the learning mechanism of CLCP, we visualize the feature space using t-distributed stochastic neighbor embedding (t-SNE), reducing it to two dimensions for better visibility. We collect channel instances of two radios for three hours and randomly select 250 timestamps, which we then feed into CLCP encoders. Each encoder output μ is represented by a color-coded data point, with red and blue points representing Radio 1 and Radio 2, respectively.

Fig. 6 depicts t-SNE visualizations of the feature space of two clients, before and after training. Prior to training, the encoder outputs form a separate and non-intersecting cluster in the latent space. Post-training, the red and blue data points coalesce into a singular and Gaussian cluster, revealing that the model learned to correlate two distinct links.

Fig. 7 provides a comprehensive analysis of the fully trained feature embedding, utilizing ToF and AoA path parameters. Closely located low-dimensional features for each radio indicate similar

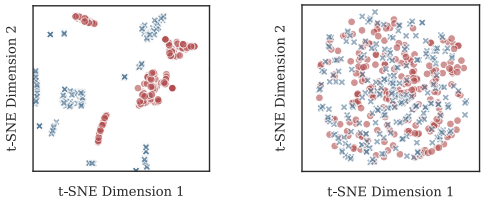


Fig. 6— 2D t-SNE visualization of the latent space of two adjacent clients before (*left*) and after training (*right*).

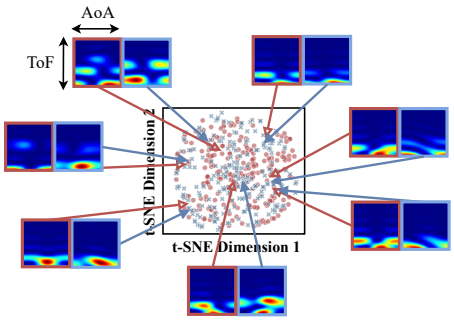


Fig. 7— A fully trained latent space example with ToF on the y-axis and AoA on the x-axis. Encoded CSI instances of Radio 1 and 2 are highlighted in red and blue, respectively.

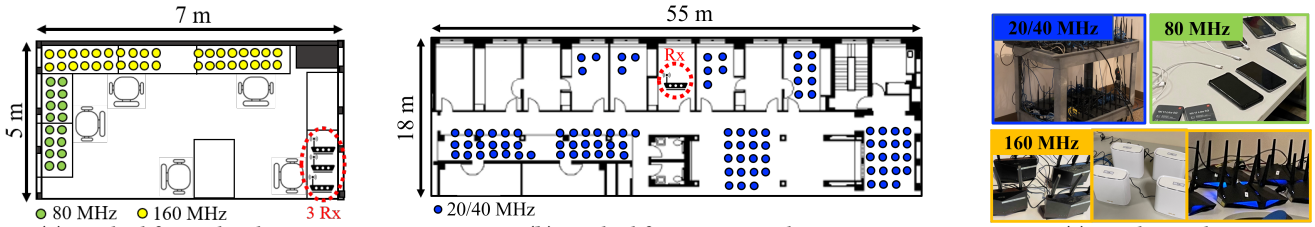
path parameters as shown in the spectra on right. Furthermore, a pattern across different radios is discernible, where the number of strong reflectors in Radio 1 and Radio 2's spectra are similar when their low-dimensional features are in close proximity. For instance, both Radio 1 and 2 exhibit many reflectors in the upper-left spectrum, and only one in the bottom-left spectrum. This reveals the CLCP's proficiency in encoding wireless channels and distributing encoded features based on reflector movement. Also, CLCP can efficiently locate encoded features in the latent space, making accurate generalizations when presented with previously unseen channels.

3.3 Scheduling and Resource Allocation

Our scheduling algorithm exploits both channel conditions and buffer status to compute an optimal user schedule. In OFDMA, the channel bandwidth is divided into different RUs with various sizes ranging from the smallest 26 tones (2 MHz) up to 996 tones (77.8 MHz). The size and the locations of RUs are defined for channels of 20, 40, 80, and 160 MHz. Our goal is to select a set of RUs that covers the entire channel bandwidth and maximizes the total channel capacity. We also need to consider the buffer status of all users to allocate large RUs for high-data devices (*e.g.,* streaming video) and smaller RUs for low-data devices. Scheduling is challenging as the search space increases exponentially with the number of users and RU granularity.

To compute the optimal set of RUs efficiently, we adopt a divide-and-conquer algorithm [19] and constrain the user assignment for each RU based on user buffers. Fig. 9 shows the RU abstraction. First, we search for a user that has low buffer occupancy and maximizes channel capacity for each of the two 26-tone RUs[4]. Next, we select the best user with more buffer occupancy for the 52-tone RU and compare its channel capacity with the sum of two 26-tone RUs.

[4]As per 802.11ax standard, each user can only be allocated one RU, so when a user is chosen for the first 26-tone RU, they are no longer considered for the second RU.



(a) Testbed for cashierless store. **(b)** Testbed for smart warehouse. **(c)** Hardware devices.

Fig. 8— CLCP preliminary experimental testbed floor plans and radio hardware with different operating bandwidths.

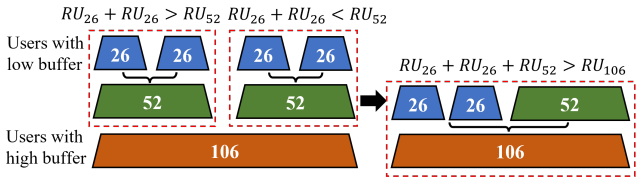


Fig. 9— Our scheduling and resource allocation algorithm.

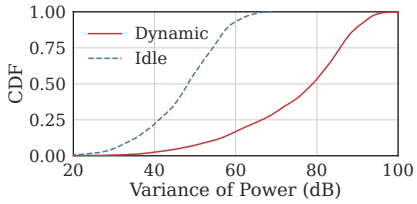


Fig. 10— Channel variability. We indicate our channel data as a red line and idle channels as a dotted blue line.

This process is repeated for subsequent resource blocks, and the RU combination with higher capacity is chosen and compared with larger RUs until the combination covers the full bandwidth.

4 IMPLEMENTATION

We conduct an experimental study on cross-link channel prediction in a large indoor lab (Fig. 8(a)) and an entire floor (Fig. 8(b)). Fig. 8(a) is equipped with high-bandwidth 802.11ax commodity radios with three APs highlighted in red - Asus RT-AX86U APs that support 802.11ax, 4x4 MIMO operation, and 160-MHz bandwidth (2048 subcarriers per spatial stream) at 5 GHz. The users (Fig. 8(c)) include the Asus RT-AX82U router (four antennas), Asus ZenWifi XT8 routers (four antennas), and the smartphones, such as the Samsung A52S (single-antenna) and Xiaomi Mi 11 (two antennas). While the bandwidth of the routers is 160 MHz, the smartphones use 80 MHz bandwidth. We identify a total of 144 separate links (considering each spatial stream as a separate link). To extract CSIs from commodity 11ax devices, we used the AX-CSI extraction tool [4]. Fig. 8(b) is equipped with low-bandwidth 802.11n commodity radios. We use 11n WPJ558 with the Atheros QCA9558 chipset (three antennas) for both the AP and users. The users are placed in 95 different locations in NLoS settings, and we extract traces using Atheros CSI Tool. All routers and phones together are generating traffic constantly using `iperf`. Two people (Fig. 8(a)) and 8 people (Fig. 8(b)) moved at a constant walking speed of one to two meters per second. Although we did not record a walking trajectory for each individual, we directed them to traverse all areas of the room or hallway. Since commodity 11ax devices do not allow OFDMA scheduling on the user side, we run a trace-driven simulation using a software-defined simulator. We implement CLCP using Pytorch, and the model parameters[5] include batch size of 16 and learning rate of $5e^{-6}$. We employ Adam for the learning rate optimization.

Channel measurement error. To minimize the packet boundary delay during OFDM symbol boundary estimation, the AP averages the phases of multiple CSIs within the channel coherence time. Then, to compensate for the amplitude offset due to the power control uncertainty error, the AP obtains the Received Signal Strength Indicator (RSSI), reported alongside CSI in the feedback, detects the RSSI outliers, and discards the associated packet. Lastly, to minimize carrier frequency offset due to non-synchronized local oscillators, the AP subtracts the phase constant of the first receiving antenna across all antennas. Since phase constant subtraction does not alter a relative phase change across antennas and subcarriers, the signal path information is preserved in CSI.

5 EVALUATION

We begin by presenting the methodology for our experimental evaluation (§5.1), followed by the end-to-end performance on throughput and power consumption (§5.2). Lastly, we present a microbenchmark on CLCP (§5.3).

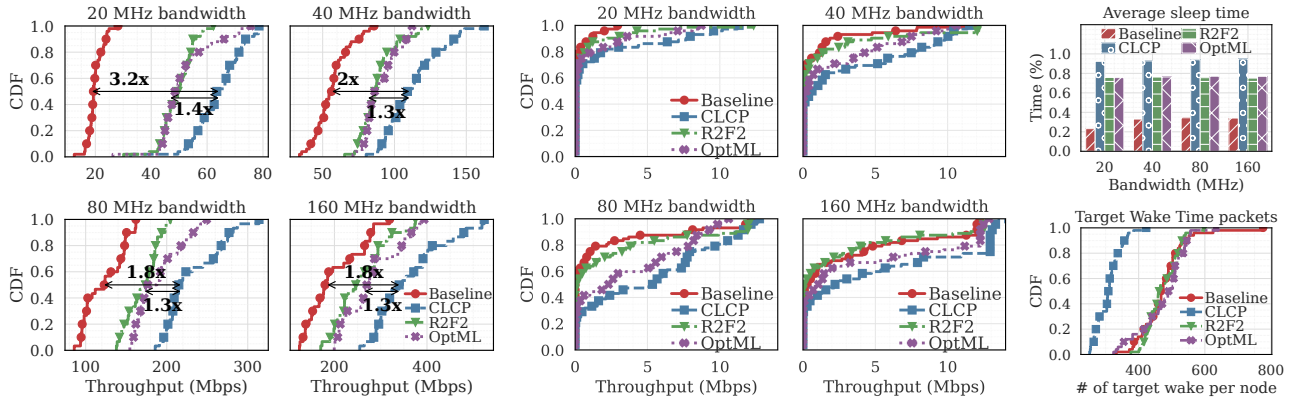
5.1 Experimental methodology

Use cases. We evaluate CLCP in two use case scenarios: a cashierless store (Fig. 8(a)) and a smart warehouse (Fig. 8(b)). **Cashierless stores** typically experience a high demand of data traffic as densely deployed video cameras continuously stream their data to the AP for product and customer monitoring. To reflect a realistic cashierless store application, we configure all users to continuously deliver standard quality video of 1080p using UDP protocol for trace-driven simulation. Also, we leverage 80 and 160 MHz bandwidth for every uplink OFDMA packet. In **smart warehouses**, IoT devices transmit relatively little data traffic and are widely and sparsely deployed compared to the cashierless store use cases. Hence, each uplink packet has 20 and 40 MHz bandwidth in NLoS settings. CLCP is scoped for static sensor deployments with environmental dynamics, and the movement of the sensor[6] is beyond the scope of our work.

Evaluation metrics. To quantify the network performance, we calculate uplink throughput as the total data bits transmitted to the

[5]We used a grid search for hyperparameter tuning. Here, we divided the domain of the hyperparameters, including alpha and beta, into a discrete grid and tried a combination of values of this grid until we find a point that minimizes the loss.

[6]In case of sensor mobility, a system that uses CLCP can gracefully degrade to the status quo for mobile sensors (*i.e.,* mobile nodes use standard channel estimation).



(a) Throughput performance across time. **(b)** Throughput performance across users. **(c)** Power consumption.

Fig. 11— End-to-end performance on throughput and power consumption: (a) aggregated throughput across time for every 500 ms, (b) throughput across users for 20, 40, 80, and 160 MHz bandwidth, and (c) device sleep time over the entire transmission duration and the total number of Target Wake Time (TWT) triggered on every user.

AP divided by the duration, measured every 500 ms. Also, we report the total number of Target Wake Time (TWT) packets to evaluate power consumption. TWT is an 11ax mechanism for power-saving, where user devices sleep between AP beacons and only wake up to transmit signals, such as uplink data transmission and channel reports. If a user is not scheduled for uplink transmission or reporting CSIs to the AP, it is not triggered for a TWT packet, increasing device sleep time and conserving IoT device battery life.

Baselines. Our baseline follows sounding protocols in which the AP periodically requests Buffer Status Reports (BSRs) and CSIs from all users. Upon receiving the Neighbor Discovery Protocol (NDP) from the AP, all users calculate the feedback matrix for each OFDM subcarrier as follows [8]:

$$\frac{\text{CSI tones} \times \text{CSI bits} \times \text{TxAntenna} \times \text{RxAntenna} \times T_c}{\text{Subcarrier Group} \times \text{Feedback Period}} \quad (4)$$

where T_c signifies the wireless channel coherence time. We use 8-bit CSI quantization, a channel coherence time of 15 ms, and a subcarrier grouping of 4. The other control protocols we consider are BSR report (32 bytes), BSR poll (21 bytes), CSI poll (21 bytes), MU-RTS (20 bytes), CTS (14 bytes), TF ($28 + (5 \times K)$ bytes), and BlockAck/BA ($22 + (5 \times K)$ bytes), where K denotes the number of users. SIFS takes $10us$. We note that BSRs and CSIs are delivered to the AP via OFDMA transmission to minimize the overhead.

Algorithms. We compare CLCP to the following algorithms which collectively represent the state-of-the-art in channel prediction: (1) R2F2 [17] infers downlink CSI for a certain LTE mobile-base station link based on the path parameters of uplink CSI readings for that *same* link, using an optimization method in lieu of an ML-based approach; (2) OptML [2] leverages a neural network to estimate path parameters, which, in turn, are used to predict across frequency bands. Both algorithms predict downlink based on uplink channels in LTE scenarios, where the frequencies for downlink and uplink are different. To adopt these algorithms into our OFDMA scheduling problem, the AP triggers all users to transmit pilot signals simultaneously in the 242-subcarrier RUs, and these algorithms predict a full-band channel (2048 subcarriers) of each user based on the received RU. We use a maximum likelihood approach for fast path parameter estimation.

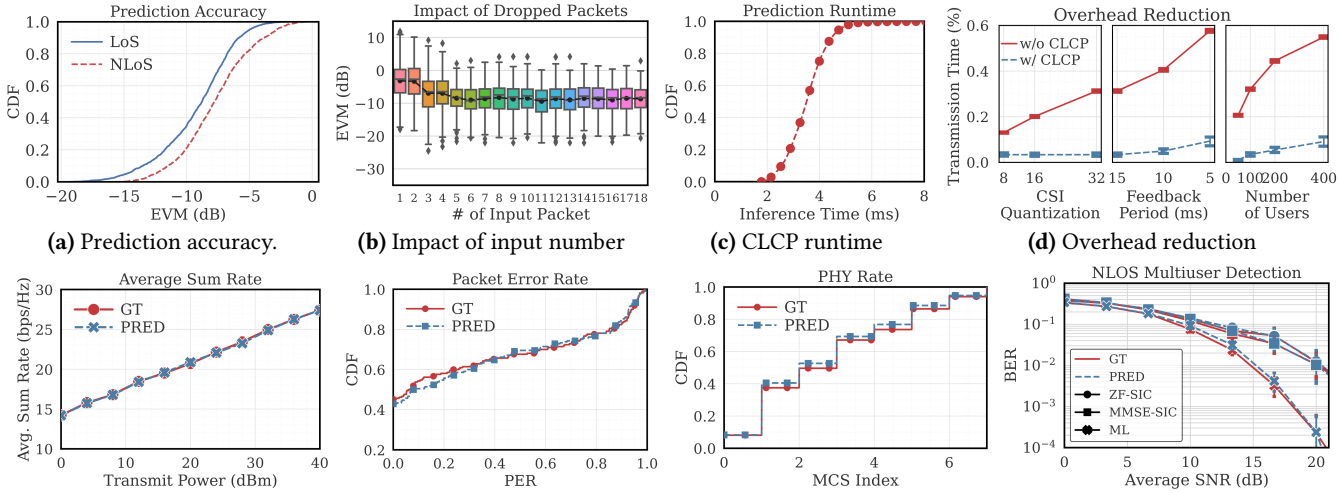
Channel variability. It is important to take into account environmental dynamics in the training dataset since training in a static environment does not generalize well. Fig. 10 demonstrates the variability of idle channels without human mobility (a dotted blue line) and our channel environment affected by moving reflectors (a red line). Here, we calculate the power variance of channels over one second (across subcarriers and links) and generate the corresponding variance distribution. In Fig. 10, the power variance of our channel data is about 30 dB higher than that of idle channel data, suggesting that our links are not idle and there is environment variability caused by moving reflectors.

5.2 End-to-end performance

We evaluate CLCP's end-to-end throughput against R2F2 and OptML, and compare its power consumption performance

Significant throughput improvement. The end-to-end throughput performance of CLCP is evaluated and compared to the baseline, R2F2, and OptML across time and users. Fig. 11(a) summarizes the results for 20, 40, 80, and 160 MHz bandwidth channels. Here, each data point of the curves indicates an aggregated uplink throughput within 500 ms duration. With 20 MHz bandwidth, CLCP shows a 3.2x improvement in throughput compared to the baseline and a 1.4x improvement over R2F2 and OptML. Similarly, CLCP provides a $1.9x$ to $2x$ improvement over the baseline for 40, 80, and 160 MHz channels and a $1.3x$ improvement over R2F2 and OptML. Particularly, CLCP improves spectral efficiency for smaller bandwidths. While existing cross-band prediction algorithms need a dedicated pilot signal for channel sounding from all users, CLCP uses the channel estimates from existing transmissions, avoiding additional signal transmissions with the corresponding control beacons.

In Fig. 11(b), we present the end-to-end throughput performance across users. Here, each data indicates the throughput of one user for 10 seconds of uplink traffic. It is worth noting that as the bandwidth increases, more users get an opportunity to send their data. For 20 MHz bandwidth, only 20% to 40% of users can send their



(a) Prediction accuracy. **(b)** Impact of input number **(c)** CLCP runtime **(d)** Overhead reduction

(e) Channel capacity. **(f)** PER distribution. **(g)** PHY rate distribution. **(h)** BER distribution.

Fig. 12— Microbenchmark on the prediction accuracy, overhead reduction, and scheduling performance.

data, while for 160 MHz bandwidth, more than 50% to 70% of users communicate with the AP. For all bandwidths, we observe that CLCP allows 15% to 20% more users to deliver their data.

Increasing device sleep time. TWT reduces the power consumption by letting users sleep and only waking up when they need to transmit their packets. Fig. 11(c) (*upper*) shows the average sleep time of all users over the entire measurement duration. While the users sleep slightly over 25% of the time using the baseline, CLCP enables 90% of users to remain in sleep state, which is roughly 65% and 15% longer than the baseline and cross-band prediction algorithms, respectively. It is worth noting that the sleep time of the cross-band algorithms is longer than the baseline because each user sends at least 260 byte of its channel information for the baseline while users simply transmit a pilot signal for R2F2 and OptML. However, CLCP does not need bulky feedbacks nor pilot signals, minimizing contention between users and increasing the duration of power save mode. Fig. 11(c) (*lower*) shows how many TWT packets are exchanged when 150 MB data are delivered to the AP. (*i.e.,* how frequently each user wakes up to participate in channel sounding and data transmission). Here, CLCP has significantly less TWT counts because the users stay idle during channel acquisition. In contrast, all users are forced to transmit a signal with the baseline, R2F2, and OptML. Given that the average wake power and transmit power is 600 uW and 135 mW, respectively, we can infer that CLCP's power consumption is significantly less.

5.3 Microbenchmark

We present a microbenchmark on the prediction accuracy, channel capacity, packet error rate, and PHY-layer bit rates[7].

Prediction accuracy. We use an *error vector magnitude* (EVM) to represent how far a predicted channel H deviates from a ground-truth H_{gt}: EVM $= 10\log_{10}\left(|H - H_{\text{gt}}|^2/|H_{\text{gt}}|^2\right)$. According to 802.11 spec. [7, 8], BPSK modulation requires an EVM between -5 and -10 dB, while QPSK from -10 to -13 dB. In Fig. 12(a), CLCP provides an EVM of approximately -8 dB. Compared to an LoS setting, an NLoS setting shows lower EVMs across different users due to multiple wall blockages and long signal propagation distance.

Impact of the number of observed channels. We evaluate the prediction accuracy with varying number of observed channels. As shown in Fig. 12(b), the prediction performance improves when there are more than two input users. However, further increasing the number of input channels does not greatly enhance CLCP's prediction accuracy. This result suggests that CLCP can predict channels correctly even when there are many unobserved channels.

Overhead reduction. Fig. 12(c) shows that CLCP achieves only about 4 ms of inference time, which comply with the runtime of other VAE-based models [9]. Next, we present the overhead reduction with varying parameters in Fig. 12(d). We define the overhead as the percentage of CSI transmission time over the total traffic time. In the absence of CLCP, a short feedback period, an increase in the number of users, and a greater number of sub-carriers result in a larger CSI overhead. In a densely deployed scenario, our CLCP notably reduces the overhead. Fig. 12(d) (*right*) shows that with 400 users, CLCP can free up more than 40% overhead.

Channel capacity. Fig. 12(e) evaluates the channel capacity of OFDMA packets that are scheduled using predicted channels. Channel capacity is defined as the sum of achieved rates at each subcarrier s, that is $R_{\text{capacity}}(RU_i) = \sum_{s\in RU_i} R_{\text{capacity}}(s)$ where RU_i is RU at i-th location. Then, we define the capacity of a final user schedule g as:

$$\sum_j R_{\text{capacity}}(p_j, u_j) = \sum_j \sum_{s\in p_j} \sum_{u\in u_j} \log_2(1 + P_{u,s}) \tag{5}$$

where $P_{u,s}$ denotes a transmit power for user u and subcarrier s. Fig. 12(e) shows that the channel capacity of packets scheduled based on predicted CSIs is nearly identical to that of ground-truth CSIs. These results indicate that our predicted channels are accurate enough for OFDMA scheduling.

PER distribution. PER distributions of packets scheduled using predicted CSIs are shown in Fig. 12(f). Even if PER is high, packets scheduled with ground-truth CSIs and predicted CSIs share similar

[7] Microbenchmark results are obtained under settings in Fig. 8(b)

PER distributions. We conclude that even if the channel condition is bad, CLCP still provides good channel prediction.

PHY rate distribution. 11ax enables each RU to have its own MCS index, which is calculated based on its channel condition. Therefore, accurate channel estimates are essential for rate adaptation. In Fig. 12(g), we present the PHY rate distributions calculated using an effective SNR (ESNR)-based rate adaptation [5]. This algorithm leverages channel estimates to find a proper MCS index. The results show that the PHY rate distributions of both ground-truth and predicted channel are similar.

Multiuser detection. 11ax uses multi-user detection algorithms to separate uplink streams from multiple users. For uplink MU-MIMO, it is crucial to select a subset of users with low spatial channel correlation *and* determine an appropriate decoding precedence. Here, we deploy several multiuser detection algorithms, such as zero-forcing (ZF) and minimum mean squared error (MMSE), that are integrated with a successive interference cancellation (SIC) and the optimal maximum-likelihood (ML) decoder. Fig. 12(h) shows a BER of packets that are scheduled with ground-truth CSIs and predicted CSIs. We decode these packets using ZF-SIC, MMSE-SIC, or ML techniques across different SNR values. We observe that the BER of packets from predicted CSIs is slightly higher than that from ground-truth CSIs for ML decoder when SNR ranges from 10 to 16 dB. In contrast, BER with ZF- and MMSE-SIC decoder shows no difference, indicating that CLCP's prediction is accurate for ZF-SIC and MMSE-SIC decoders.

6 CONCLUSION

This paper presents the first study to explore cross-link channel prediction for the scheduling and resource allocation algorithm in the context of 11ax. We have evaluated CLCP using 802.11ax, validating the effectiveness of our system through microbenchmarks and end-to-end performance.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. AST-2232457, by the European Office of Aerospace Research & Development (EOARD) under award number FA8655-22-1-7017, and projects TSI-063000-2021-59 RISC-6G and TSI-063000-2021-63 MAP-6G funded by the Spanish Ministry of Economic Affairs and Digital Transformation under the European Union NextGeneration-EU plan. We gratefully acknowledge the support of a gift from the Amateur Radio Digital Communications Foundation and a grant from the Princeton University School of Engineering and Applied Science Innovation Fund.

REFERENCES

[1] Hareesh Bahuleyan, Lili Mou, Olga Vechtomova, and Pascal Poupart. 2017. Variational attention for sequence-to-sequence models. *arXiv preprint arXiv:1712.08207* (2017).

[2] Arjun Bakshi, Yifan Mao, Kannan Srinivasan, and Srinivasan Parthasarathy. 2019. Fast and efficient cross band channel prediction using machine learning. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.

[3] Boris Bellalta. 2016. IEEE 802.11 ax: High-efficiency WLANs. *IEEE wireless communications* 23, 1 (2016), 38–46.

[4] Francesco Gringoli, Marco Cominelli, Alejandro Blanco, and Joerg Widmer. 2022. AX-CSI: Enabling CSI extraction on commercial 802.11 ax Wi-Fi platforms. In *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization*. 46–53.

[5] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. 2010. Predictable 802.11 packet delivery from wireless channel measurements. *ACM SIGCOMM computer communication review* 40, 4 (2010), 159–170.

[6] Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14, 8 (2002), 1771–1800.

[7] IEEE 2010. IEEE Standard Part 11: 802.11p, Amendment 6: Wireless Access in Vehicular Environments.

[8] IEEE 2019. IEEE Standard Part 11: P802.11ax/D4.0. , 746 pages.

[9] Minyoung Kim and Vladimir Pavlovic. 2020. Recursive inference for variational autoencoders. *Advances in Neural Information Processing Systems* 33 (2020), 19632–19641.

[10] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[11] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. 2016. Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*. 2352–2360.

[12] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. 2019. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8247–8255.

[13] Souvik Sen, Bozidar Radunovic, Jeongkeun Lee, and Kyu-Han Kim. 2013. Cspy: Finding the best quality channel without probing. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. 267–278.

[14] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. 2018. Cross-modal deep variational hand pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 89–98.

[15] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. 2018. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *Proceedings of the European Conference on Computer Vision*. 155–171.

[16] D Tse and P Viswanath. 2005. *Fundamentals of Wireless Communication*. Cambridge University Press.

[17] Deepak Vasisht, Swarun Kumar, Hariharan Rahul, and Dina Katabi. 2016. Eliminating channel feedback in next-generation cellular networks. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 398–411.

[18] Jing Wang, Jian Tang, Zhiyuan Xu, Yanzhi Wang, Guoliang Xue, Xing Zhang, and Dejun Yang. 2017. Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach. In *IEEE INFOCOM 2017-IEEE conference on computer communications*. IEEE, 1–9.

[19] Kaidong Wang and Konstantinos Psounis. 2018. Scheduling and resource allocation in 802.11 ax. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 279–287.

[20] Mike Wu and Noah Goodman. 2018. Multimodal generative models for scalable weakly-supervised learning. In *Advances in Neural Information Processing Systems*, Vol. 31. 5575–5585.

[21] Xiufeng Xie, Xinyu Zhang, and Karthikeyan Sundaresan. 2013. Adaptive feedback compression for MIMO networks. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. 477–488.

[22] Yan Yang and Hao Wang. 2018. Multi-view clustering: A survey. *Big Data Mining and Analytics* 1, 2 (2018), 83–107.