Fast and Efficient Cross Band Channel Prediction Using Machine Learning

Arjun Bakshi
The Ohio State University
bakshi.11@osu.edu

Yifan Mao
The Ohio State University
mao.360@osu.edu

Kannan Srinivasan
The Ohio State University
kannan@cse.ohio-state.edu

Srinivasan Parthasarathy
The Ohio State University
srini@cse.ohio-state.edu

ABSTRACT

Channel information plays an important role in modern wireless communication systems. Systems that use different frequency bands for uplink and downlink communication often need feedback between devices to exchange band specific channel information. The current state-of-the-art approach proposes a way to predict the channel in the downlink based on that of the observed uplink by identifying variables underlying the uplink channel. In this paper we present a solution that greatly reduces the complexity of this task, and is even applicable for single antenna devices. Our approach uses a neural network trained on a standard channel model to generate coarse estimates for the variables underlying the channel. We then use a simple and efficient single antenna optimization framework to get more accurate variable estimates, which can be used for downlink channel prediction. We implement our approach on software defined radios and compare it to the state-of-the-art through experiments and simulations. Results show that our approach reduces the time complexity by at least an order of magnitude (10x), while maintaining similar prediction quality.

CCS CONCEPTS

• **Networks** → *Wireless access networks*; Mobile networks.

KEYWORDS

Machine learning; Wireless networks

MobiCom '19, October 21–25, 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6169-9/19/10…$15.00
https://doi.org/10.1145/3300061.3345438

ACM Reference Format:
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 (MobiCom '19), October 21–25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3300061.3345438

1 INTRODUCTION

In the last two decades, wireless technology has significantly improved user throughput by exploring multi-antenna and multi-user approaches. Very recently, there has been an interest in developing base stations with hundreds of antennas called mega MIMO (multiple input multiple output) systems [16, 18, 19]. These systems can simultaneously talk to multiple clients that each have just a few antennas. For all of these techniques, client or client set selection is critical for performance. These selections, in turn, are based on the channel between the transmitting and receiving antennas across multiple devices.

In application domains like cellular networks, a device receives data in one frequency band, but transmits in another. This is done in order to facilitate simultaneous transmissions in both directions without the use of in-band full duplex techniques. In such applications, the channel observed in the receiving frequency band are different from those in the transmitting frequency band. Therefore, a device can not use the channel it observes in the receiving frequency band to perform tasks like beamforming or rate selection in the transmit frequency band. Traditionally, in cellular systems, channel values are reported back to the base station by the mobile clients, either with or without compression. However, those approaches can be prohibitively expensive or can compromise accuracy [4, 9, 23, 34, 36, 36].

A recently proposed approach, R2F2 [31], aims to eliminate the aforementioned feedback overhead. It enables a base station to derive channel values to a client in the downlink (transmit) frequency band based on the channel it observes from the client in the uplink (receive) frequency band. Their

primary insight is that while the observed channel values between two devices are different in different frequency bands, the physical signal paths underlying the channels remain the same. Thus the channel can be modeled with a set of frequency independent parameters (number of multipath components, individual path lengths, power etc.), and a frequency dependent parameter, which is the frequency band of the signal itself. The goal of R2F2 is to estimate the frequency independent parameters from the observed channel. However, their algorithm can be computationally expensive, depending on the number of channel components, and number of antennas on the device. This prevents R2F2 from being used on devices with low computational power and/or single antenna systems like personal or smart home devices.

While cross band channel estimation is already an important issue in for base stations in large wireless networks, we believe that it can be useful in many other domains. For instance, most traffic in home Wi-Fi networks carries video streaming data [27], which is on the downlink from the Wi-Fi access point to the laptop. With such asymmetry in traffic, one can envision protocols that use different bands and bandwidths for uplink and downlink channels based on the amount of data being transmitted in either direction. In that case, laptops and access points may want to beam data to each other but will need channel feedback. However, laptops may not be able to use R2F2 as their antennas usually don't form a linear array, while an access point may not have enough computational power.

Another application for cross band channel prediction is in smart-home networks where a large number of devices frequently transmit small packets of data. One can envision a scenario where devices operate in a frequency agile manner, changing transmission bands in order to avoid packet collisions with other devices, or use bands with better SNRs.

Motivated by the limitations of the current state-of-the-art and a wide range of applications, we present OptML, a system that leverages machine learning speed up processes involved in channel prediction. OptML reduces the complexity of channel prediction by orders of magnitude compared to R2F2, and is capable of running on single antenna devices. It uses an efficient optimization framework that matches the channel prediction performance of R2F2 in most cases. It leverages a light-weight machine learning (ML) model for generating high quality initial guesses which are used to initialize the optimization framework. Thus, our work is applicable to devices like cellphones, laptops, and IoT nodes, which do not have linear antenna arrays, and have lower computational power.

We implement OptML on USRP radios and compare its performance with R2F2 (current state-of-the-art) in an indoor testbed and in simulations. Our evaluations show that OptML provides beamforming gains similar to those of R2F2 for indoor and simulated environments, for various antenna arrays sizes and multipath channels. Additionally, it is able to do so orders of magnitude faster than R2F2. The contributions of our work are:

- We present a highly efficient and flexible framework for channel prediction that can be used by base stations, user end devices, and smart home devices alike.
- Results show that it is able to provide similar prediction accuracy to the current state-of-the-art approach while providing a speed up of up to 80x.

2 BACKGROUND

2.1 Channel Basics

When a device transmits a signal, the signal gets distorted by the environment that it travels through. The signal undergoes attenuation (a) due to path loss and absorption, and phase changes due to the distance traveled (d) by it, as well as reflections (ϕ). These changes are collectively referred to as the "channel" which affects the signal. For a signal transmitted at a frequency of f_i (wavelength λ_i), the channel, h_i, can be expressed as [28]:

$$h_i = ae^{\frac{-j2\pi d}{\lambda_i} + j\phi} \tag{1}$$

Often times, the signal observed at a receiver is a functional composition of multiple copies of the original signal, where each copy arrives with a slightly different delay, power, and phase. In those cases, the is channel can be represented as the sum of the channels caused be each path, and can be represented as:

$$h_i = \sum_n a_n e^{\frac{-j2\pi d_n}{\lambda_i} + j\phi_n} \tag{2}$$

As seen in Eq. 2, the channel depends on the individual paths through which the signal travels, and the wavelength of the signal itself. For a single antenna, a multipath channel can be described by a set of 3-tuples, $\{(d_i, a_i, \phi_i)\}$. For an antenna array, the channel observed at antenna K_i, for wavelength λ due to N signal paths can be expressed:

$$h_{K_i,\lambda} = \sum_n^N \left(a_n e^{\frac{-j2\pi d_n}{\lambda} + j\phi_n}\right) e^{\frac{-j2\pi i l cos(\theta_n)}{\lambda}} \tag{3}$$

Note that the first term is the same as in Eq. 2. The second term represents the effect of the additional distance the signal travels to a specific antenna, as shown in Figure 1.

2.2 Primer on R2F2

R2F2[31] is the current state-of-the-art method that enables a device with a linear antenna array to predict the channel to a client in a frequency band F_1 based on the channel it observes from the client in a frequency band F_2. Based on the channel model described in Section 2.1 and Eq. 3, R2F2's goal

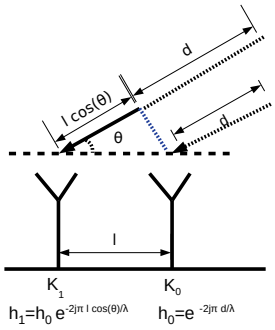


Figure 1: Channels induced by a single signal path at two antennas.

is to identify a number of paths (N) and the corresponding 4-tuple of parameters $\{(a_n, d_n, \phi_n, \theta_n)\}_{n=0}^{N}$ that would induce the channel observed in F_1 at the antenna array. Once the underlying parameters have been estimated, the channel in any other frequency band can be computed.

R2F2 operates in two stages. The first stage generates a set of initial guesses, and the second stage refines them in order to fit the observed channel. To generate the initial guesses, R2F2 computes a likelihood function, $P(d, \theta|h)$, which, given the observed channel h, estimates the likelihood of a signal from a distance d and an angle of arrival θ being a part of the observed channel. It computes the value of $P(d, \theta|h)$ over a range of values, similar to as shown in Figure 2. As seen in Figure 2, multiple regions of the parameter space are highlighted as likely components of the channel.

In the second stage, as the number of multipath components is unknown, R2F2 uses the mostly likely pair (d, θ) to initialize an optimization. The optimization updates $\{(d, \theta)\}$ to minimize the difference between the observed channel and the channel induced by those estimates. If the optimization minimizes the difference below some threshold, then R2F2 assumes that the correct set of $\{(d, \theta)\}$ has been estimated. Otherwise, the next most likely pair of (d, θ) is added to the set of guesses used in the optimization process. This process repeats itself until a good fit is achieved, or the improvement per additional component is below some threshold.

2.3 Limitations of R2F2

While R2F2 can enable channel prediction across frequencies, it is designed for base stations, which have a significant computational resources and linear antenna arrays. This limits the applicability of R2F2 to end-user devices. For example, newer laptops can have up to three antennas, but are scattered along different axes, around the screen and keyboard.

Due to the non-convexity of the objective function used in their optimization, R2F2's end result is sensitive to the quality of the initial guesses. However, the process of generating initial guesses in R2F2 is prone to errors. To illustrate

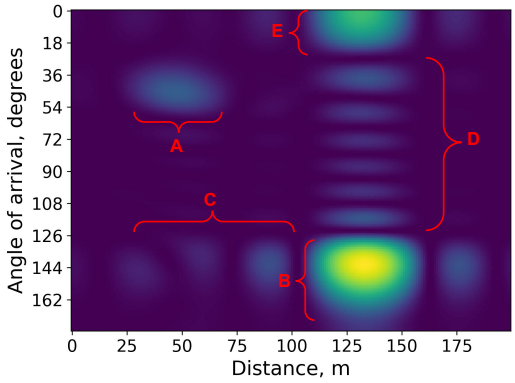


Figure 2: Initial guess generation: Actual channel paths from (130m, ~ 145°) and (40m, ~ 45°) are highlighted in regions marked A and B. However, some other incorrect regions of the parameter space are also highlighted (C, D, E).

this, we generate a (simulated) channel with two multi-path components, and plot the initial guesses for that channel as described in [31]. In Figure 2, the x-axis represents distance, and the y-axis represents the angle of arrival. The value at a position represents the likelihood of a signal from that distance and angle of arrival being a part of the observed channel. The regions marked as B and A represent correct guesses for the channel components. For the stronger component (@130m and ~ 145°), the initialization method suggests paths from multiple angles of arrival. Such "side-lobes" can be stronger than other actual paths. For example, region A is an actual component that has likelihood value lower than or similar to regions C, D and E, which are side-lobes of B. Under some conditions a grating lobe may also manifest (E), which can have power comparable to the main lobe (B). This ambiguity can lead R2F2 to either over estimate the number of paths and incorrectly decomposing the channel, or waste time exploring sub-optimal regions of the parameter space.

3 SCOPE AND CHALLENGES

Our goal is to enable channel prediction on devices with a single antenna, non-linear antenna arrays, or devices with low computational power. While performing channel prediction independently at each antenna would provide a solution for all those cases, certain challenges and limitations of such a solution must be addressed.

3.1 Correctness of decomposition

In order to correctly decompose a channel observed on a single antenna into its underlying set of 3-tuples, a unique mapping or correspondence must exist between a set of 3-tuples and a particular channel on a single antenna. However, such a unique correspondence is not always guaranteed.

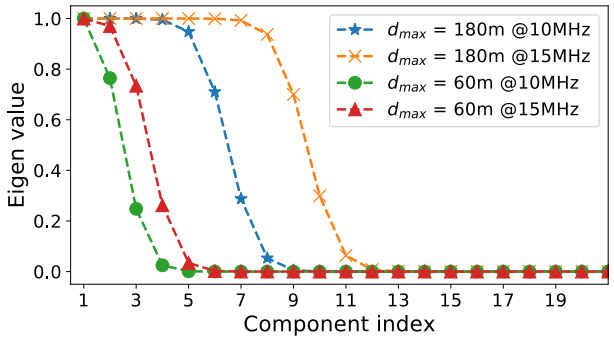


Figure 3: Eigen values of most significant principal components of matrix D. As the bandwidth of the signal is increased, channels get measured on more independent subcarriers, and the number of non-zero components in the PCA increases.

To understand why, consider the following: Construct a matrix D such that $D_{i,n} = e^{-2j\pi d_n/\lambda_i}$, where $d_n \in \{0 \ldots d_{max}\}$, $\lambda_i \in \{\lambda_0 \ldots \lambda_{N_{fft}}\}$, and N_{fft} represents the number of subcarriers in the transmitted signal. That is, each column of D represents the channel for a signal that travels a distance d_n. One can then generate a large number of multipath channels by picking different columns from D, and adding them in different linear (scaled and rotated) combinations.

For example, consider a matrix D_{180} with $d \in \{1, 2 \ldots 180\}$, and wavelengths ($\lambda$) corresponding to a 10MHz bandwidth around the 2.4GHz center frequency. This matrix can be used to generate channels that have multipath components with lengths between 0m and 180m. A multipath channel H which uses the 10^{th} and 75^{th} columns of D can be generated as:

$$H = D\vec{a} \tag{4}$$

where $\vec{a}$ represents a complex valued vector in which the n^{th} element represents the attenuation and phase associated with the channel in column n of D. In our example, only a_{10} and a_{75} have non-zero values which represent the attenuation and phase offsets for paths that travel 10m and 75m. While $\vec{a}$ and D_{180} can be constructed with smaller steps or higher resolution, it is not useful to increase the resolution beyond a certain point, as dictated by the signal bandwidth.

As a part of channel prediction, one would like to estimate $\vec{a}$ when H and D are given. One way to do so is:

$$\vec{a} = D^{+}H \tag{5}$$

where D^{+} represents the inverse of D.

However, Eq. 5 will have a unique solution only when D is invertible. A principal component analysis of D_{180} reveals that it is *rank deficient and thus non-invertible*. Figure 3 shows the eigen values of the most significant components of D_{180}, and D_{60}. It shows that D is rank deficient as number of non-zero eigenvalues is less than number of columns in D, and the determinant of D is zero, as the product of the eigenvalues is zero. *Thus a unique solution to the decomposition may not always be possible.* In other words, a channel measured at a *single antenna* may not contain enough information to be able to identify the correct underlying parameters.

One such example is shown in Figure 4. The channel represented by H_A contains 10 multipath components, and H_B is a channel with 6 components. While the two sets of multipath components are very different, their corresponding channels are very similar in the uplink frequency band. More importantly, while the channels look the same in the uplink band, they are very different in the downlink band. Thus, an algorithm that tries to predict the downlink channel based on a single uplink channel observation will fail in this case.

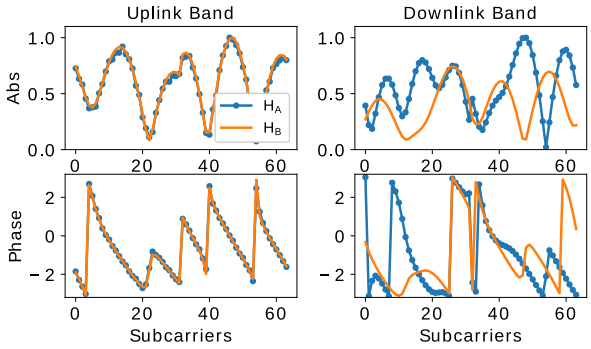


Figure 4: An example of two channels due to different components that are similar in the uplink band, but very different in the downlink band. A correct cross-band prediction is not possible in this case.

In a multi-antenna system, the distances travelled by the components to each antenna is slightly different, which results in linearly independent channels across the antennas. This can enable a more accurate channel decomposition at the cost of higher complexity.

While a correct decomposition may not be always feasible from a single channel observation, if the matrix D is constructed carefully as a part of an optimization problem, as later described in Section 4.2, then it is more likely that the correct solution can be estimated.

3.2 Initialization and Resilience to Noise

This work focuses on single antenna systems and systems with arbitrary antenna arrangements. As such we can not rely on angular resolution to resolve components, or multiple antennas to overcome low SNR. Thus we need a way to generate high quality initial estimates despite the loss of angular resolution and in cases with low SNR.

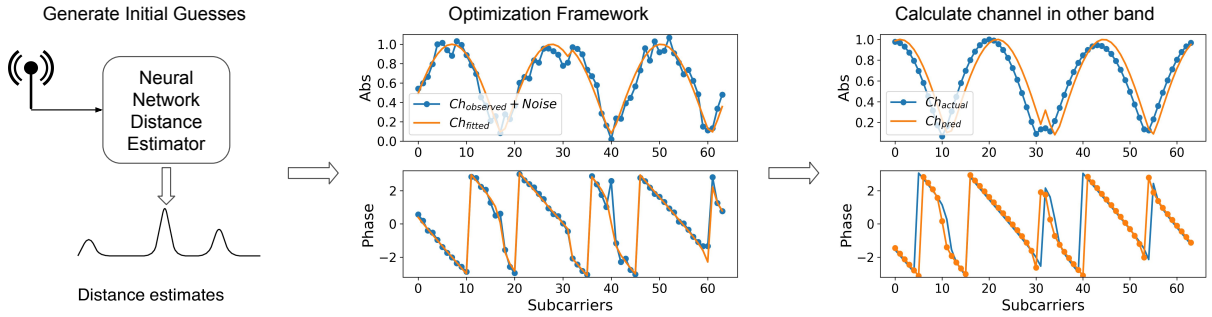


Figure 5: Overview of OptML. A neural network model is used to generate coarse estimates of the lengths of the multipath components in the observed channel. These estimates are then refined by an optimization process. The refined components are then used to calculate the channel in another band.

4 SYSTEM DESIGN

In this section, we present our proposed system, OptML. As shown in in Figure 5, OptML starts by generating estimates of distances travelled by multipath components in a channel (Section 4.1), and then uses an optimization process (Section 4.2) to refine them so that the channel based on the estimates closely fits the observed channel. Once the optimization converges, the final estimates are used to calculate the channel in the downlink using Eq. 2. We also present an algorithm for channel prediction on multi-antenna systems that do not have linear antenna arrays.

4.1 Generating Initial Guesses

A simple way to generate coarse distance estimates (initial guesses) is to compute the similarity of the observed channel and channels associated with signals traveling different distances. For a channel h measured over I sub-carriers, let h_i represent the channel at wavelength λ_i. Then, the likelihood of h containing a path from distance d can be computed as:

$$P(d|h) = ||\sum_{i}^{I} h_i e^{j2\pi d/\lambda_i}||^2 \quad (6)$$

Eq. 6 computes to the similarity between h_i and the channel induced at λ_i due to a multipath component that travels a distance d, and sums it across all sub-carriers. The greater the value, the higher the likelihood of the observed channel containing a component from that distance.

Figure 6 shows the output of such a predictor for a channel with multipath components that travelled 110m, 135m and 167m. The channel due to them is observed at two antennas ~0.24m apart (approx. half wavelength, center frequency 650MHz and bandwidth 10MHz). We see that this predictor generates significantly high peaks around the distances corresponding to these components. However it is unable to distinguish between some paths. Typically, such a predictor

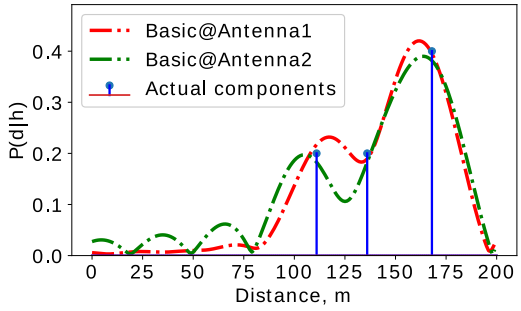


Figure 6: The basic predictor correctly identifies the most significant component, but can not resolve other components, and generates some false peaks/guesses.

can not resolve copies of signal that arrive within one sample duration ($T_s = \frac{1}{bw}$), or when the difference in the distances travelled by two components is less than cT_s , where c is the speed of light. We refer this distance as d_{samp}. Additionally, this estimator generates peaks at distances that do not correspond to any actual multipath component (false positives). These false positives can lead to over estimating the number of multipath components and consequently an incorrect channel prediction. These false positive peaks are akin to the side lobes seen in the initialization method used by R2F2.

4.1.1 Neural Network Based Distance Estimator(NNDE). In order to overcome the aforementioned issues, we propose using a neural network to build a distance estimator. One could consider this as a regression task where a channel is the input, and the output is a vector that represents the likelihood of multipath components from different distances being a part of the channel. For example, if guesses need to be generated for paths that can arrive from any distance between 0 and 200 meters, then the output vector can be

of length 200 with each index corresponding to a particular distance. However, this formulation has the following issues:

- Sparsity: As seen in Figure 6, the ideal output vector (based on actual components) is sparse as only a few non-zero values exist in the output vector. Sparse outputs can be difficult for neural networks to learn.
- Resolution: While the ideal predictor should be able to generate accurate estimates for the distance of the multipath components, a model's ability to resolve two close by components based on the channel is related to the bandwidth over which the channel is observed. A higher bandwidth provides greater resolution.
- Complexity of problem: As the number of multipath distance combinations is extremely large, the task will require an extremely large neural network.

While a target vector with greater sparsity allow for greater resolution between multipath components, it significantly increases the complexity of the machine learning model.

In order to overcome these issues we design the target output for the neural network by convolving the ideal (sparse) output with a Gaussian smoothing function as shown in Figure 7. The extent of smoothing done by a Gaussian filter depends on its σ (standard deviation) parameter. If the value of σ is too low, then the output vector will remain sparse and not be amenable to fitting by a neural network. Conversely, if it is too high, then it may not provide any improvement over the basic predictor. We identify a good value for σ as follows: We first pick a value of σ that results in an output vector with the same resolution as that of the basic predictor. We then train a neural network with that value of σ and measure the quality of fit based on a loss metric (mean squared error etc.). We keep reducing the value of σ and training new networks until the fit of the neural network stops improving. The smallest value of σ for which the neural networks is able to learn a good fit is chosen as the final value of σ.

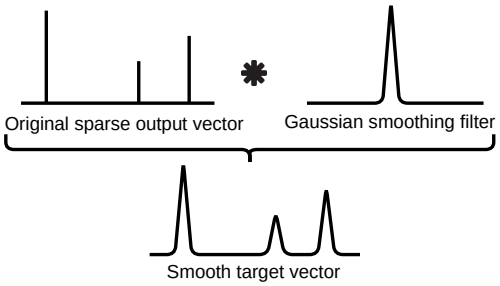


Figure 7: Output vector construction: Convolve the sparse target vector with a Gaussian to get a non-sparse target vector that the NNDE can fit more easily.

A fully connected feed forward neural network architecture is used for building the Neural Network Distance Estimator (NNDE). This choice is motivated by the small number of parameter needed to define a network and the simplicity of the model compared to more sophisticated neural network architectures. We discuss the different parameters involved in the NNDE design in Section 7. The input to the neural network is the frequency domain representation of the channel observed at a single antenna. A channel observed over N_{fft} subcarriers can be represented as a vector of N_{fft} complex numbers. A single channel can be fed to the neural network as a vector of $2N_{fft}$ real values numbers. The network then outputs a vector that represents the likelihood of different multipath components distances in the input channel.

Figure 8 shows the output of the NNDE for the same channel on which the basic predictor was tested. The peaks in the output of the NNDE correspond to the most likely distances for the channel components. It shows that the neural network based predictor is able to generate high quality estimates for the number of multipath components in a channel, their distances and amplitudes. It is also able to clearly resolve between components that are less than d_{samp} apart.

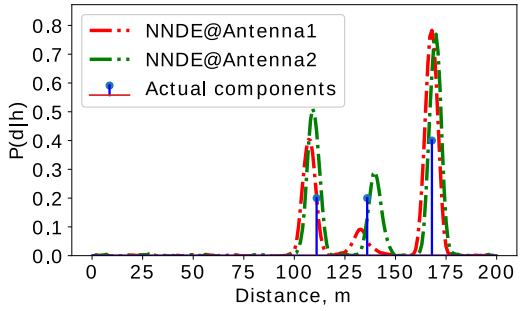


Figure 8: The NNDE correctly identifies components and has no false positives for the same channel that the basic estimator was tested on.

4.2 Optimization Formulation

While the NNDE gives us coarse estimates of the distances of the channel components $\{d_n\}$, they need to be refined before they can be used for channel prediction. This is done through an optimization. While the channel model for a single antenna is described in terms of sets of 3-tuples, $\{(d, a, \theta)\}$, we show how the optimization can be framed in terms of only $\{d_n\}$. Firstly, within Eq. 2 the phase offset term, $e^{-j\phi_n}$, can be combined with the attenuation term, a_n, thus converting a_n from a real valued variable to a complex variable. Furthermore, given a set of distances $\{d_n\}$, one can compute its corresponding solution set $\{a_n\}$ by using a least squares

method. Following the formulation in Section 3.1, let H represent the channel observed on a single antenna, D represent a matrix of size IxN, where I is the number of subcarriers in the signal and N is the number of multipath components estimated by the NNDE, and $\vec{a}$ represents the complex attenuation associated with each path. Then the relationship between H, D and $\vec{a}$ can be represented as follows:

$$H = D\vec{a}$$

For a known H and D, the value of $\vec{a}$ can be estimated using a least squares approach, e.g. pseudo-inverse, in which case $\vec{a} = D^{+}H$. Thus, $\{a_n\}$ can be eliminated from the optimization parameter space. This leaves $\{d_n\}$ as the only set of variables that need to be estimated. Therefore the objective function that needs to be optimized can be written as:

$$O(\{d_n\}_{n=1}^{N}) = ||H - DD^{+}H|| \tag{7}$$

As noted in Section 3.2, the optimization needs to be resilient to noise, and should not try to fit to the noise. We note that fitting noise often results in extremely large value in $\vec{a}$. We therefore use a regularization term (with a scaling α) that takes the amplitude into account, and rewrite the objective function as:

$$O(\{d_n\}_{n=1}^{N}) = ||H - DD^{+}H|| + \alpha \sum |D^{+}H| \tag{8}$$

While this objective function is non-convex, just like the one used in R2F2, it is a much simpler formulation. Additionally, the initial guesses generated by the NNDE increases the likelihood of the optimization finding a good solution.

4.3 Leveraging Multiple Antennas

Multi-antenna devices provide multiple independent channel observations which can be leveraged for getting better guesses. This becomes more important as the number of multipath components in the channel increases. As noted in Section 3, it is difficult to correctly identify the multipath components from the channel of a single antenna. Thus, if the channel has a lot of multipath components, the NNDE may not be able to generate good initial guesses.

Recall that multipath components travel similar distances to all antennas in an antenna array. The extra distance travelled by any component is based on the relative distances between the antennas. For a linear antenna array, this distance is $ilcos(\theta)$, as seen in Figure 1 and Eq. 3. In general, the difference in the distance travelled by a single component to two antennas is bounded by the separation between the two antennas, i.e. $\delta \in [-l, l]$.

Consider a channel composed of two components from distances d_1 and d_2. For clarity of explanation, we omit the other parameters (a and ϕ) and limit the example to two paths. However, the following analysis does not depend on this simplification (refer Appendix A for proof). The channels induced by these components at wavelength λ at two antennas, K_1 and K_2, separated by distance l are:

$$h_{K_1} = e^{\frac{-j2\pi d_1}{\lambda}} + e^{\frac{-j2\pi d_2}{\lambda}} \qquad h_{K_2} = e^{\frac{-j2\pi (d_1+\delta_1)}{\lambda}} + e^{\frac{-j2\pi (d_2+\delta_2)}{\lambda}}$$

where $\delta_1, \delta_2 \in [-l, l]$ correspond to the difference in distance travelled for each path to antenna K_2 compared to K_1. The term $e^{\frac{-j2\pi (d_1+\delta_1)}{\lambda}}$ can be treated as the product of $e^{\frac{-j2\pi d_1}{\lambda}}$ and $e^{\frac{-j2\pi \delta_1}{\lambda}}$. Thus, if h_{K_1} is multiplied by $e^{\frac{-j2\pi \delta_1}{\lambda}}$, and then subtracted from h_{K_2}, then the contribution of the signal component due to d_1 will be eliminated. As the result, the number of components in the channel is reduced, and the NNDE can use it to generate good component distance estimates.

However, in practice, the correct value of δ_1 in not known, and even if the exact value of δ_i is known, the multiplication-subtraction operation introduces extra terms. In our example, the suppression process will be as follows, with the extra term written in **bold**:

$$Multiply: h'_{K_1} = h_{K_1} e^{\frac{-j2\pi \delta_1}{\lambda}} = e^{\frac{-j2\pi (d_1+\delta_1)}{\lambda}} + e^{\frac{-j2\pi (d_2+\delta_1)}{\lambda}}$$

$$Subtract: h_{subtr} = h_{K_2} - h'_{K_1} = e^{\frac{-j2\pi (d_2+\delta_2)}{\lambda}} \mathbf{- e^{\frac{-j2\pi (d_2+\delta_1)}{\lambda}}}$$

The analysis in Appendix A assumes that δ_i in unknown, and shows that the extra terms do not change the channel enough to affect the performance of the NNDE. The final result from Appendix A is as follows:

$$h_{subtr} = h_{K_2} - h'_{K_1} = 2jsin(\pi(\delta - \delta_1)/\lambda)\ e^{\frac{-2j\pi}{\lambda}\left(d_1 + \frac{\delta_1+\delta}{2}\right)} + 2jsin(\pi(\delta - \delta_2)/\lambda)\ e^{\frac{-2j\pi}{\lambda}\left(d_2 + \frac{\delta_2+\delta}{2}\right)} \tag{9}$$

The exponential terms in Equation 9 represents channel components very similar to the original components, and not some new or spurious component. More importantly, the $jsin()$ terms *scale* each exponential term. If the δ term is close or equal to either δ_1 or δ_2, then the $sin()$ term will be close to zero and effectively suppress that channel component. Since we know that $\delta_i \in [-l, l]$, one can sample values of δ from within that range with the goal of suppressing different components each time. Each suppressed channel is fed into the NNDE to generate an output vector g_{subtr}, similar to that seen in Figure 8. The final set of initial guesses is generated by adding up the output vectors $\{g_{subtr}\}$ for the inputs $\{h_{subtr}\}$ and detecting significant peaks in that vector.

4.4 Channel Prediction Algorithms

We now describe how the optimization and NNDE fit together for devices with single or multiple antennas. Table 1 contains a list of terms used in this section.

4.4.1 Single Antenna Devices. The algorithm for predicting the channel using a single antennas is described in Algorithm 1. The algorithm starts by using the uplink channel, H_{UL}, and the NNDE to generate initial guesses for the multipath components (line 1). Peak detection is used on the output of the NNDE to identify the most likely distances. We refer to this set of initial guesses as $\{d\}_{init}$. The initial guesses can be used to initialize the objective function described in Section 4.2. However, as the objective function is non-convex, it is possible that the optimization can move far form the region described by the initial guesses, and move away from the optimal solution. This can happen if the point described by the initial guesses is at a cusp. For that reason, we restrict the optimization to a region close to the initial guesses. This has the added benefit of limiting the search space, and making a more exhaustive search also possible. In our implementation, the initial guesses $\{d_1, d_2, ...\}$ are replaced with bounded variables $\{(d_1 \pm b), (d_2 \pm b), ...\}$, or $\{d\}_{bounded}$ (line 2). A differential evolution algorithm is used to find the best solution within the region described by these bounds (line 3). Once the solution $\{d_1, d_2, ...\}_{final}$ with the least error has been identified, the corresponding $\vec{a}$ term is computed (lines 4-5). Finally, the channel in the downlink frequency band is constructed (line 6-7).

Algorithm 1 Single Antenna Channel Prediction

Input: $NNDE$, λ_{UL} , λ_{DL} , H_{UL} , b
Output: H_{DL}
1: $\{d\}_{init} = NNDE(H_{UL})$
2: $\{d\}_{bounded} = get\_bounded(\{d\}_{init}, b)$
3: $\{d\}_{final}, error = Optimize(H_{UL}, \{d\}_{bounded})$
4: $D_{UL} = get\_matrix(\{d\}_{final}, \lambda_{UL})$
5: $\vec{a} = D_{UL}^{+} H_{UL}$
6: $D_{DL} = get\_matrix(\{d\}_{final}, \lambda_{DL})$
7: $H_{DL} = D_{DL}\vec{a}$

4.4.2 Multi-Antenna Devices. For a multi-antenna device, the channels measured over K antennas $H_{UL,K}$, is used to generate the set of initial guesses, $\{d_n\}_{init}$, based on the process described in Section 4.3 (line 1). These initial guesses are bounded and used to find the best solution for each antenna independently based on the channel $H_{UL,k}$ observed at that antenna k (lines 2-5). The error for each solution is compared across all antennas, and the antenna and solution corresponding to the least error are noted as K_{ref} and $\{d\}_{ref}$ (lines 6-12). In the final stage of the algorithm, $\{d\}_{ref}$ is used to initialize the optimization problem at all other antennas. However, this time the bounds are set to be much tighter than before. The bounds are based on the distances between K_{ref} and the antenna for which the optimization is being solved, k. That is, if the distance between antenna k and K_{ref} is l, then the final optimization at antenna k is restricted to the region described by $\{d \pm l\}_{ref}$ (lines 14-16). The optimizations are once again solved independently at each antenna which identify the final set of distances $\{d\}_{final}$ for each antenna. The corresponding set of $\vec{a}$ are found for each antenna (lines 17-18). Finally, the channel at antenna k in the downlink band is calculated (line 19-20).

Term	Definition
K	Number of Antennas
λ_{UL}	Uplink (UL) subcarrier wavelengths
λ_{DL}	Downlink (DL) subcarriers wavelengths
b	Bound use to limit optimization region
$\{d\}$	Path lengths for channel components
$H_{UL,K}, H_{DL,K}$	Channel matrix over K antennas
D_{UL}, D_{DL}	Matrix as described in Section 3.1

Table 1: Table of notations

Algorithm 2 Multi-antenna Channel Prediction

Input: $NNDE$, K , λ_{UL} , λ_{DL} , $H_{UL,K}$, b
Output: $H_{DL,K}$
1: $\{d_{init}\} = NNDE\_Subtr\_Guesses(H_{UL,K})$
2: $\{d_{bounded}\} = Update(\{d_{init}\}, b)$
3: $\{d\}_{ref} = \emptyset$, $error_{min} = \infty$, $K_{ref} = 0$, $k = 0$
4: **while** $k < K$ **do**
5: $\{d\}, error = Optimize(H_{UL,k}, \{d\}_{bounded})$
6: **if** $error < error_{min}$ **then**
7: $error_{min} = error$
8: $\{d\}_{ref} = \{d\}$
9: $K_{ref} = k$
10: **end if**
11: $k = k + 1$
12: **end while**
13: **while** k < K **do**
14: $l = Separation(k, K_{ref})$
15: $\{d\}_{bounded} = Update(\{d\}_{ref}, l)$
16: $\{d\}_{final}, error = Optimize(H_{UL,k}, \{d\}_{bounded})$
17: $D_{UL,k} = get\_matrix(\{d\}_{final}, \lambda_{UL})$
18: $\vec{a} = D_{UL,k}^{+} H_{UL,k}$
19: $D_{DL,k} = get\_matrix(\{d\}_{final}, \lambda_{DL})$
20: $H_{DL,k} = D_{DL,k}\vec{a}$
21: **end while**

4.5 Generating Sufficient Training Data

The variables for generating the NNDE training data are:

- RF parameters: center frequency and bandwidth
- Maximum number of multipath components, N_{max}
- Component distances, $d \in [d_{min}, d_{max}]$
- Attenuation, $a \in (0, 1]$

- Phase rotation, $\phi \in [-\pi, \pi]$

While the RF parameters can be fixed for a certain application, and most variables have a small range, the range of distances travelled by each components can be quite large. This presents an issue for generating an extensive dataset, as it increases the number of multipaths possible, and therefore the training dataset size.

However, we note that the range for d can be reduced to the maximum delay spread of the environment by using a channel modification process similar to the one used in Section 4.3. Specifically consider an NNDE model trained to identify components in the range of [0m, 200m], and a channel that contains components that travel d_1=1250m and d_2=1300m. Let h represent the channel, where:

$$h = a_1 e^{\frac{-j2\pi d_1}{\lambda_i}} + a_2 e^{\frac{-j2\pi d_2}{\lambda_i}}$$

To change the channel into something the NNDE has trained on, we first define a shifting vector as follows:

$$h_{shift} = e^{\frac{j2\pi d_{shift}}{\lambda_i}}$$

Note the positive exponent term. The term d_{shift} represents a distance value that can be subtracted from d_1 and d_2 to bring both of them into the range[0,200]. Let us set d_{shift} to 1200. We then multiply the channel h with the shifting vector as follows:

$$\begin{aligned} h' = h \;\; e^{\frac{j2\pi d_{shift}}{\lambda}} &= a_1 e^{\frac{-j2\pi(d_1 - d_{shift})}{\lambda}} + a_2 e^{\frac{-j2\pi(d_2 - d_{shift})}{\lambda}} \\ &= a_1 e^{\frac{-j2\pi d_1'}{\lambda}} + a_2 e^{\frac{-j2\pi d_2'}{\lambda}} \end{aligned}$$

Where d_1' and d_2' will be equal to 50m and 100m in our example. This channel can now be fed into the NNDE to get distance estimates. Once the estimates have been generated, d_{shift} can be added to the estimates to get the actual distance values that will be used by the optimization algorithm.

A good value for d_{shift} can be estimated from the channel's impulse response. The impulse response of a channel measured in the frequency domain can be computed through its IFFT. Figure 9 shows the impulse response of our example channel (d_1=1250m, d_2=1300m, 10MHz bandwidth). The plot on top shows the IFFT of the channel while the lower one shows the region around the peak. The bin index of the peak can be used to get a coarse estimate of the distance. For a signal with a bandwidth of 10MHz, each bin corresponds to a distance of 30m (c/BW). Based on the significantly non-zero value in the 40^{th} bin, we get an estimate of 1200m for d_{shift}.

This process allows us to limit the training to the NNDE to the delay spread and not the maximum communication range of the network. As the delay spread can be an order of magnitude smaller than communication ranges [8], this greatly reduces the amount of data needed to be generated.

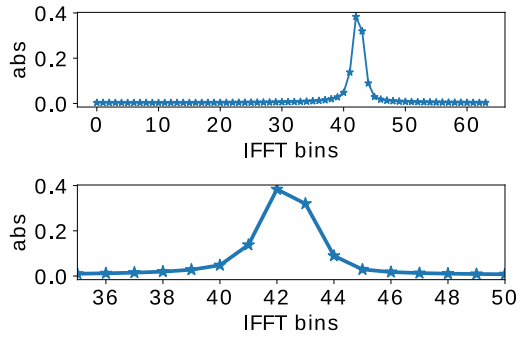


Figure 9: The location of the peak in the channel impulse response provides a coarse estimate of d_{shift}.

5 EXPERIMENTS AND SIMULATIONS

We implemented OptML and R2F2 using the USRP N210s software defined radio platform. We implemented a 3-antenna base station and a single antenna client that are synchronized via an external clock. Experiments were conducted in a large indoor lab space (Figure 10). The uplink and downlink frequencies are separated by 30 MHz and have a bandwidth of 10MHz. We used the 2.4GHz ISM band in our experiments as it allowed for license free operation. While this frequency band is different from the one used in [31], it does not affect the performance of either algorithm. We also run simulations in the same cellular network frequency band used in [31]. The NNDE is implemented using Keras [5], and the optimization in Python [12, 32]. The NNDE used in this paper is composed of 10 hidden layers, each with 200 neurons with the Exponential Linear Unit (ELU) activation function. The model was trained on 1.5 million data points, each of which represents a channel and target vector. The training channels were generated with varying SNR and up to 6 multipath components. We limit the number of multipath components to 6 based on the discussion in Section 3. The trained model takes less than 6MB of disk space.

5.1 Beamforming results

Beamforming is one of the most common use cases of channel values in MIMO systems. For that reason, we evaluate the performance of our system by using the predicted channels to beamform to a client from the base station. We measure the SNR of the signal at the receiver when beamforming is done using the actual channels in that frequency band, channels predicted by OptML and R2F2, as well as when no beamforming is done. We refer to the transmit beamforming done by using the actual/ground truth channels in the downlink band as the "optimal" beamforming. Figure 10 shows the layout of the room used for this experiment. While we

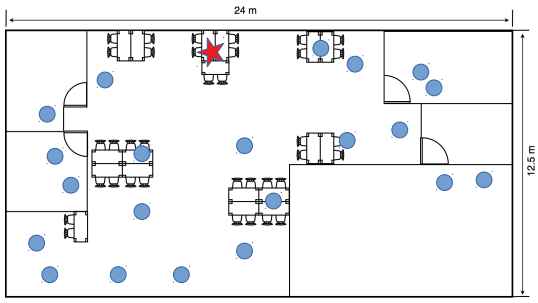


Figure 10: Layout of the indoor lab space used for experiments. The red star represents the location of the base station, while the blue dots represent some of the locations for the client node.

collected data more densely throughout the room, only a few locations have been highlighted as examples for clarity.

Figure 11 shows the beamforming performance of R2F2 and OptML in the indoor testbed. We see that the performance of OptML closely follows that of R2F2, and is within 1dB of the optimal beamforming performance. We also see that the median gain over no beamforming is approximately 4dB, which is close to the theoretical maximum for a 3-antenna transmitter.

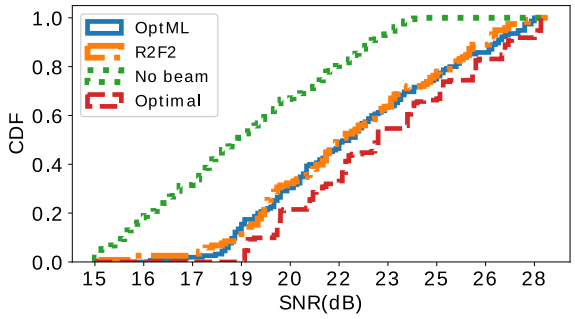


Figure 11: Beamforming gains (indoor testbed): The beamforming gains from channels predicted by OptML are similar to those of R2F2, and is within 1dB of the optimal performance based on the actual downlink channels.

We perform simulations in order to test the performance of OptML in environments where we can increase the delay spread, and finely control the multipath. A testbed with the same parameters used in [31] is simulated, where each component can travel distances between [0m, 200m]. The attenuations $a \in (0, 1]$, angles of arrival θ and phase offsets $\phi \in [-\pi, \pi]$ are chosen uniformly at random from within their respective ranges. Simulations are done with different number of antennas in the array (3-10 antennas), with the inter antenna separation set to half the wavelength of the uplink frequency. The uplink and downlink frequencies are 650MHz and 680MHz, respectively. The bandwidth is set to 10MHz in each band. The maximum number of components is 6, based on the discussion in Section 3.

Figure 12 shows the CDF of the beamforming gains for OptML and R2F2 for channels with up to 6 multipath components for a base station with different number of antennas (K). It shows that the beamforming gains of OptML match those of R2F2 for all antenna arrays tested. Also, the gains increase as the number of antennas in the array increase, which is in line with the expected theoretical gains.

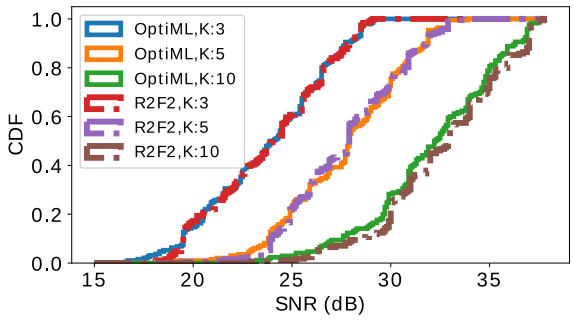


Figure 12: Beamforming gain (simulations): For antenna arrays of different sizes (K) with channel with up to 6 multipath components.

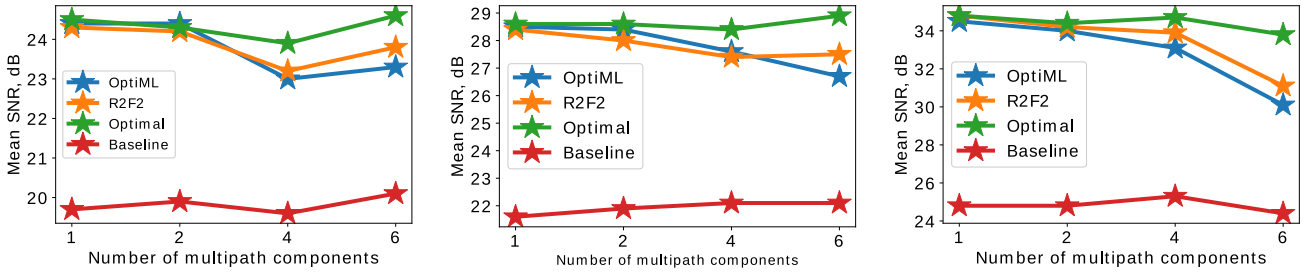
Figure 13 shows a breakdown of how each approach performs for different number of multipath components and antennas. When the channel contains fewer components, OptML performs almost optimally. However, as the number of components increase, the performance of both R2F2 and OptML degrades. R2F2 is able to perform slightly better as it can resolve components with similar distances in the angular domain, while OptML can not.

In addition to matching the performance of R2F2 in most cases, a major advantage of OptML is its low computational complexity, which is detailed in the following section.

5.2 Run time

One of our main goals in designing OptML was to reduce the time taken to generate a channel prediction so that the system is more suitable for machines with limited computational power. Figure 14 shows the run time distribution for OptML and R2F2 from experiments in the indoor testbed. It shows that OptML provides ~8x reduction in runtime, which includes time taken to generate initial guesses and the optimization stage for both approaches.

In order to study the performance of both systems in more challenging conditions, we ran extensive simulations with the same parameters as earlier. Figure 15 shows the runtimes of OptML and R2F2 for the 3-antenna base station setup.



(a) Beamforming with a 3 antenna base station (b) Beamforming with a 5 antenna base station (c) Beamforming with a 10 antenna base station

Figure 13: Beamforming(simulation): OptML and R2F2 closely follow the optimal beamforming performance for all array sizes when the number of multipath components is low. Even when the number of components increase, they provide significant gains over the baseline.

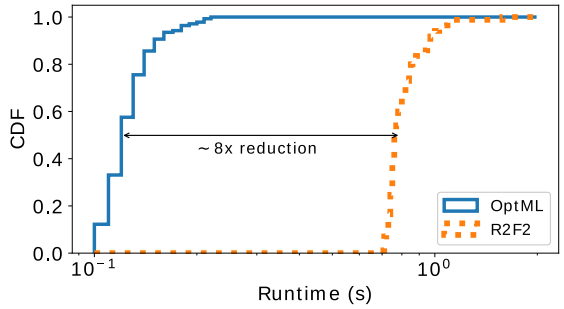


Figure 14: Runtimes (indoor testbed). OptML shows an order of magnitude reduction in the median time taken to generate a prediction compared to R2F2.

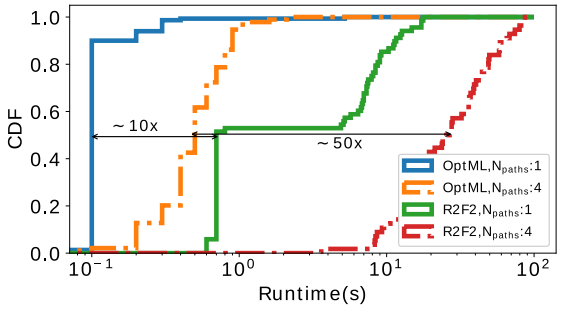


Figure 15: Run times (simulations) for a 3 antenna system. The runtime gain of OptML over R2F2 increases with the number of multipath components.

It shows that for a single component channel, OptML provides a 10x speedup. For more complex channels with 4 components, it provides a 50x speedup over R2F2. This factor increases as the number of antennas and multipath components increase (up to 80x for a 10-antenna system). The reason R2F2 takes this long is partially because its method for generating initial guesses produces a lot of false positives, which results in it exploring sub-optimal regions of the parameter space. In contrast, the NNDE is less likely to generate false positives.

Figure 16 shows a breakdown of the runtimes of OptML and R2F2 for various antenna array sizes and number of multipath components. Each point represents the median runtime for an approach for a channel with a certain number of multipath components, and a fixed number of antennas. In general, we see that both approach take longer to generate a prediction as the number of multipath components increases, or as the number of antennas increases. However, in all cases, the median runtime of OptML is much lower than that of R2F2. In fact, the speedup is higher when the number of multipath components or number of antennas increases.

The increase in R2F2's runtime as the number of components increases is partially due to its iterative design. It starts off by assuming the observed channel can be fit using one component, and then increases that number if the optimization can not find a good fit with the current number of components. As a result, it has to run for much longer before it ends up using enough components to fit the observed channel. In contrast, OptML uses all the components suggested by the NNDE to model the channel at once, and as the NNDE is less likely to generate false positives, the search space of the optimization is not unnecessarily increased.

5.3 Estimating Number of Components

In this section we compare the number of components in the channel to the number estimated by OptML and R2F2. We do so by simulating channels with known number of components under the same conditions as described in Section 5.1, and then using OptML and R2F2 to estimate the number of components in the channel. However, it should be noted that such a comparison comes with a caveat. Specifically, that though a channel may have N components, some of them

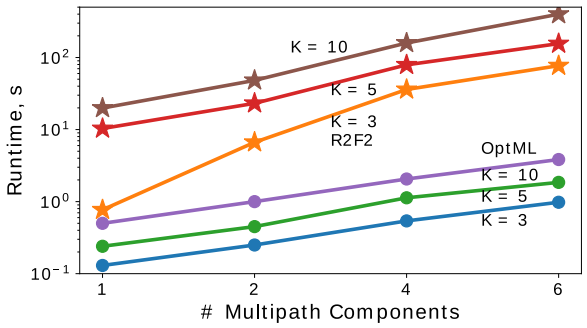


Figure 16: Speedup: The median runtime of both approaches increases as the number of multipath components increase. However, the runtime for OptML grows more slowly compared to R2F2.

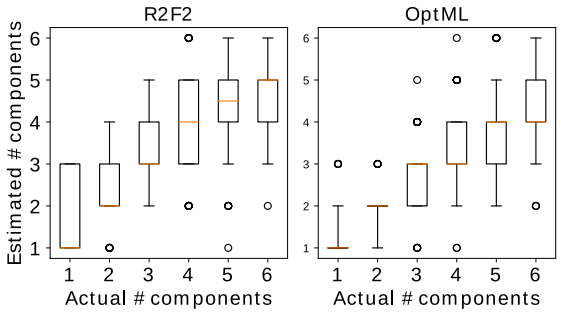


Figure 17: Actual vs estimated number of components: R2F2 has a tendency of over estimating the number of components, even when the channel has only one component. OptML generates more accurate estimates when the channel has fewer components.

may have very low amplitude. In those cases, the channel can be modeled very accurately with fewer than N components. *Thus, estimating fewer than the actual number of components is less harmful to a prediction than over-estimating the number of components.*

With that caveat in mind, we use box-plots in Figure 17 to compare the actual and estimated number of components by each approach. The actual number of components are marked on the x-axis, and the estimates are along the Y axis. The median is drawn in orange within the box, and the whiskers represent the 5^{th} and 95^{th} percentiles. Outliers are plotted as circles, where darker circles represent multiple outliers at that point. The figure shows that R2F2 has a tendency of overestimating the number of components, especially when the number of components are low. In contrast, the NNDE used in OptML provides more accurate guesses when the number of actual components are low. This is because when fewer components are in the channel, the channels due to each individual component is more likely to be independent or dissimilar. As the number of components increases, the probability of two components traveling similar distances increases. This becomes an issue for the NNDE as it can not leverage angular resolution to separate them. However, as seen from the results in Section 5.1, OptML's performance does not degrade a lot. This reinforces our intuition about under estimating the number of components being less harmful than over estimating them.

5.4 Single Antenna Evaluations

In this section we evaluate the performance of OptML for a single antenna device. Consider a scenario where devices in a smart home can transmit in one of many bands, similar to the 11 bands or channels in WiFi at 2.4GHz. Under this setup, a device may choose to a particular transmission band based on whether it is free of interference, or its SNR. In this scenario, a device can use OptML and a single beacon from the AP in any band to estimate channel in any other band.

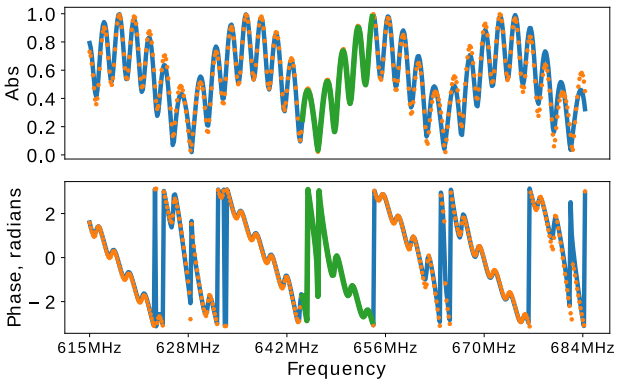


Figure 18: Channel values predicted in ±30MHz band around input channel band by a single antenna device using OptML. The actual channel values are in blue, the predicted in orange, and the input channel band is plotted in green.

We use simulations to test OptML in such a scenario. For this test, we measure the channel in a 10MHz band and predict it in the ±30MHz band around the measured channel. The same parameters as the previous simulations are used here. A sample prediction by OptML for a multitap channel is shown in Figure 18.

To get an broader view of the quality of the predictions, the correlation of the predicted and actual channel's amplitude is measured and its distribution is shown in Figure 19. As a baseline, we compute the correlation between a flat channel and the actual channel over the 70MHz. The median correlation between the actual channel and the baseline is zero,

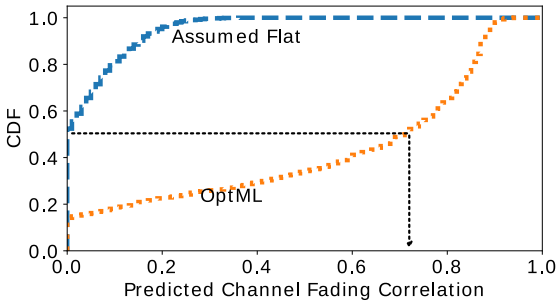


Figure 19: Correlation of the amplitude of the actual channel with channel predicted by OptML, and flat channel. While the flat channel has a median correlation of zero, the predictions of OptML have very high correlation with the actual channels.

while that of OptML is close to 0.7. Thus OptML can provide useful guidance for band selection. Since OptML provides the full CSI within each band, and not just a mean SNR value for the band, it can also be used to perform power allocation across the subcarriers.

6 RELATED WORK

As comparisons between OptML and R2F2 have been done throughout the paper, this section covers other related work.

6.1 Cross Band Channel Prediction

The problem of dealing with the overhead of channel feedback has also been addressed in ways compress or reduce the amount of feedback needed [7, 13]. The clear advantage of our approach over such techniques is the complete elimination of the feedback overhead. The authors of [26] address the issue of feedback in 60GHz systems by using the channel in 2.4GHz to help select the analog beamforming code in 60GHz. The approach works by using the Wi-Fi channel to localize the client, and then picking a code from a mmWave beamforming codebook which will point a 60HGz beam in the direction of the client. While this approach does provide cross frequency band guidance for beamforming, it only works for line-of-sight channel scenarios.

In [1, 20, 25], the authors propose deep learning frameworks for channel prediction. These approaches differ from OptML as they use neural networks to learn an end-to-end prediction between the uplink and downlink channel. That is, the output of the neural network is the channel in the target frequency band, as opposed to estimates of the multipath components. While such systems can further reduce the time required to generate a prediction, the size of the neural networks used are much larger. Additionally, these models were evaluated on different channel models (VehA, SUI-5, 3GPP) to the one considered here. Those channel models often have a dominant path (LoS or NLoS) or have well separated components which makes channel prediction easier, as a unique mapping is more likely to exist.

6.2 Machine Learning in Wireless Networks

There has been considerable work in the application of machine learning to wireless networks [11, 21, 38]. In [38], the authors explore the use of deep neural networks for estimating the signal detection and decoding. In [21], the goal is to use support vector machines to estimate the linear and nonlinear components of the channel from the signal in order to improve the BER. However, these approaches are aimed at estimating the in-band channel, and not predicting the channel across different frequency bands. There has been a lot of work in the application of ML techniques to predict channel occupancy in the TV white-space [3, 10, 29]. While one could try to predict the channel in the downlink over time based on previous downlink or uplink channel information, most existing work does not predict full CSI information, and is limited to coarse SNR or occupancy information.

The task of the NNDE can be thought of as a 1-dimensional deconvolution of the channel into its constituent components. While a lot of work exists on convolutional neural networks (CNNs) [33, 37] that can be used to improve the NNDE, the goal of this paper is to show how neural networks can be used to detect multipath components.

6.3 Super-resolution and Localization

As a part of OptML, we present the NNDE which can provide some form of super-resolution or component localization. Indoor localization is a well-studied problem that often deals with multipath and super-resolution [14, 15, 22, 24, 35]. However, most systems require an array of antennas on a device for them to work [14, 24, 30], or require cooperation between multiple devices in order to localize the target[35]. As our goal is to build a system that can work with even a single antenna, we can not use those approaches for our problem. Most localization work focuses on line-of-sight scenarios where the goal is to get very accurate estimates of the time-of-flight of the signal to multiple nodes or antennas. While OptML could be used to facilitate localization, we consider such an application as out of the scope of this work.

7 DISCUSSION

7.1 NNDE: Architecture

For the purposes of this work, we used a fully connected neural network architecture for the NNDE. As noted earlier, this choice is motivated by the small number of hyper parameters

compared to a convolutional neural network or other more sophisticated architectures. The number of neurons per layer and number of layers are the primary hyper parameters for this architecture, and need to be estimated through trial and error, or a grid search. The problem of finding the correct hyper parameters for a neural network is a research problem in itself [2, 6, 17]. However, as this search is done before the model is deployed to the device, its cost does not affect the runtime of the system.

7.2 NNDE: Performance Variables

We now discuss the variables that affect the accuracy of the estimates generated by the NNDE.

Signal bandwidth or sampling rate: Sampling rate has a direct relationship with the ability of a device to resolve two signals arriving at slightly different times. In general, as the sampling rate increases, two components can be separated more finely in the time domain. The analog in the frequency domain is that increasing the bandwidth increases the range of frequencies over which the signal is observed, which provides more independent observations for how the two signal components are adding up. This can also be seen in Figure 3 where, for a given delay spread, the number of non-zero eigenvalues (matrix rank) increases with the signal bandwidth. With more independent observations, we can resolve the two components more accurately.

Delay spread of channels: The delay spread of the channel represents the difference between the shortest and longest path length in the channel. In other words, it represents the range of distances that a signal may travel to reach the receiver. This in turn corresponds to the size of the input and output space for the neural network in the NNDE. That is, if the delay spread is increased, then the number of possible channels (input space) increases, and the length of the target vector (output space) also increases. The size of the input and output space affects the size of the neural network needed to learn the mapping between them, and the amount of data needed for training the network. The larger the input and output space, the larger the neural network needs to be to learn the mapping, and the larger the training dataset needs to be to train it.

Gaussian filter: The standard deviation of the Gaussian filter σ controls the sparsity of the target or output vector for the NNDE. The sparsity of the output, in addition to the signal bandwidth, controls the ability of the NNDE to resolve to channel components with similar path lengths. As noted in Section 4.1.1, σ needs to be chosen carefully so as to provide greater resolution, while still allowing the NNDE to fit a good model.

Number of components: In our experiments, we limit the number of components to 6 based on the discussion in Section 3.1, and our evaluation parameters (bandwidth and maximum delay spread). The ability of the NNDE to resolve components depends on the sampling rate. If more components are packed into the a small delay spread, then the components can not be resolved unless the bandwidth or sampling rate of the signal is increased.

While the effects of each variable is easier to understand, there is also some important interplay between these variables. In particular, larger delay spreads can introduce sparsity issues for the output vector if the Gaussian filter is not chosen correctly. The target vector can become sparse when the length of the target vector is increased due to the larger delay spread, and the Gaussian filter is kept small or narrow in order to provide greater resolution. In such cases, even if the bandwidth of the signal is high enough to resolve components from similar distances, the NNDE may not be able to learn an accurate model.

7.3 Hardware Based Channel Asymmetry

The channel observed at a device can also be affected by imperfections in the hardware of the transmitter and receiver. For example, band-pass filters have non-ideal characteristics at the edges of their pass band. These imperfections can be considered as the baseband channel, and can cause problems in channel prediction. While accounting for the baseband channel is outside the scope of our work, a possible solution to this is to construct a database which contains baseband channel information for each transmitter in the network. This information can be used to account for the baseband channel in the channel estimation step before prediction.

8 CONCLUSION

This paper presents an efficient and flexible way to predict channels across frequency bands that can be used by devices with a wide range of antenna arrangements and computational power, ranging from cellphones and laptops to smart home IoT devices. We evaluated and compared OptML to the current state-of-the-art approach through experiments and extensive simulations. The proposed technique is able to match the performance of the current state-of-the-art approach for beamforming in most cases, while providing a order of magnitude reduction in runtime (10-100x) and complexity. To our knowledge, this is the first paper that applies machine learning to the problem of estimating component distances in a multipath rich channel. Given the flexible nature of the approach, single antenna operation, and low complexity, we envision this approach being extremely useful in next generation networks.

Acknowledgement: This work is supported by The National Science Foundation grants 1254032, 1514260, 1547306, 1629548, 1747447, and The Office of Naval Research grant N00014-17-1-2412.

Appendix A CHANNEL SUBTRACTION

Recall that the channel at two antennas K_1 and K_2 are defined as:

$$h_{K_1} = e^{\frac{-j2\pi d_1}{\lambda}} + e^{\frac{-j2\pi d_2}{\lambda}}, h_{K_2} = e^{\frac{-j2\pi (d_1+\delta_1)}{\lambda}} + e^{\frac{-j2\pi (d_2+\delta_2)}{\lambda}}$$

In the following analysis we work under the condition where the ideal value of δ is unknown. We pick a random δ for h'_{K_1} as:

$$h'_{K_1} = h_{K_1} e^{\frac{-j2\pi\delta}{\lambda}} = e^{\frac{-j2\pi (d_1+\delta)}{\lambda}} + e^{\frac{-j2\pi (d_2+\delta)}{\lambda}}$$

We make the following substitutions for brevity:

$$a'_1 = \frac{-2\pi(d_1+\delta)}{\lambda} \qquad b'_1 = \frac{-2\pi(d_2+\delta)}{\lambda}$$
$$a_2 = \frac{-2\pi(d_1+\delta_1)}{\lambda} \qquad b_2 = \frac{-2\pi(d_2+\delta_2)}{\lambda}$$

and rewrite h'_{K_1} and h_{K_2} as:

$$h'_{K_1} = e^{ja'_1} + e^{jb'_1}, \quad h_{K_2} = e^{ja_2} + e^{jb_2}$$

Upon using Euler's formula we get:

$$h'_{K_1} = cos(a'_1) + jsin(a'_1) + cos(b'_1) + jsin(b'_1)$$
$$h_{K_2} = cos(a_2) + jsin(a_2) + cos(b_2) + jsin(b_2)$$

We now subtract h'_{K_1} from h_{K_2}:

$$h_{K_2} - h'_{K_1} = cos(a_2) - cos(a'_1) + jsin(a_2) - jsin(a'_1) + cos(b_2) - cos(b'_1) + jsin(b_2) - jsin(b'_1)$$

After applying the sum-to-product identities, simplifying, and re-applying Euler's formula, we get,

$$= 2jsin\left(\frac{a_2 - a'_1}{2}\right) e^{\frac{a_2+a'_1}{2}} + 2jsin\left(\frac{b_2 - b'_1}{2}\right) e^{\frac{b_2+b'_1}{2}}$$

After replacing a'_1, a_2, b'_1, b_2 with the initial values, and simplifying, we get:

$$h_{subtr} = h_{K_2} - h'_{K_1} = 2jsin(\pi(\delta-\delta_1)/\lambda)\ e^{\frac{-2j\pi}{\lambda}\left(d_1+\frac{\delta_1+\delta}{2}\right)} + 2jsin(\pi(\delta-\delta_2)/\lambda)\ e^{\frac{-2j\pi}{\lambda}\left(d_2+\frac{\delta_2+\delta}{2}\right)} \quad (10)$$

Note that the variables for different paths are not combined with each other in any stage, and are treated independently at every step. The combinations happen between terms corresponding to the same paths at different antennas (a'_1 with a_2, b'_1 with b_2). Thus the analysis holds for any number of paths in the channel.

REFERENCES

[1] Maximilian Arnold, Sebastian Dörner, Sebastian Cammerer, Sarah Yan, Jakob Hoydis, and Stephan ten Brink. 2019. Enabling FDD Massive MIMO through Deep Learning-based Channel Prediction. *arXiv preprint arXiv:1901.03664* (2019).

[2] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2546–2554. http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf

[3] Mario Bkassiny, Yang Li, and Sudharman K Jayaweera. 2013. A survey on machine-learning techniques in cognitive radios. *IEEE Communications Surveys & Tutorials* 15, 3 (2013), 1136–1159.

[4] Junil Choi, David J Love, and Patrick Bidigare. 2014. Downlink training techniques for FDD massive MIMO systems: Open-loop and closed-loop training with memory. *IEEE Journal of Selected Topics in Signal Processing* 8, 5 (2014), 802–814.

[5] François Chollet and others. 2015. Keras. https://keras.io. (2015).

[6] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. 2015. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[7] Yu Han, Tien-Hao Hsu, Chao-Kai Wen, Kai-Kit Wong, and Shi Jin. 2018. Efficient downlink channel reconstruction for FDD transmission systems. In *2018 27th Wireless and Optical Communication Conference (WOCC)*. IEEE, 1–5.

[8] Telesystem Innovations. 2010. LTE in a Nutshell. *White paper* (2010).

[9] Ralf Irmer, Heinz Droste, Patrick Marsch, Michael Grieger, Gerhard Fettweis, Stefan Brueck, Hans-Peter Mayer, Lars Thiele, and Volker Jungnickel. 2011. Coordinated multipoint: Concepts, performance, and field trial results. *IEEE Communications Magazine* 49, 2 (2011), 102–111.

[10] Sweta Jain, Apurva Goel, and Prachi Arora. 2019. Spectrum Prediction Using Time Delay Neural Network in Cognitive Radio Network. In *Smart Innovations in Communication and Computational Sciences*. Springer, 257–269.

[11] Chunxiao Jiang, Haijun Zhang, Yong Ren, Zhu Han, Kwang-Cheng Chen, and Lajos Hanzo. 2017. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications* 24, 2 (2017), 98–105.

[12] Eric Jones, Travis Oliphant, and Pearu Peterson. 2014. {SciPy}: open source scientific tools for {Python}. (2014).

[13] Mahdi Barzegar Khalilsarai, Saeid Haghighatshoar, Xinping Yi, and Giuseppe Caire. 2018. FDD massive MIMO: Efficient downlink probing and uplink feedback via active channel sparsification. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[14] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 2015. SpotFi: Decimeter Level Localization Using WiFi. In *Proceedings of the 2015 ACM SIGCOMM Conference*. ACM, 269–282.

[15] Xinrong Li and Kaveh Pahlavan. 2004. Super-Resolution TOA Estimation With Diversity for Indoor Geolocation. *IEEE Transactions on Wireless Communications* 3, 1 (2004), 224–234.

[16] Le Liang, Wei Xu, and Xiaodai Dong. 2014. Low-complexity hybrid precoding in massive multiuser MIMO systems. *IEEE Wireless Communications Letters* 3, 6 (2014), 653–656.

[17] Ilya Loshchilov and Frank Hutter. 2016. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269* (2016).

[18] Lu Lu, Geoffrey Ye Li, A Lee Swindlehurst, Alexei Ashikhmin, and Rui Zhang. 2014. An overview of massive MIMO: Benefits and challenges. *IEEE journal of selected topics in signal processing* 8, 5 (2014), 742–758.

[19] Hariharan Rahul, Swarun Suresh Kumar, and Dina Katabi. 2012. Megamimo: Scaling wireless capacity with user demand. In *Proc. ACM SIGCOMM*, Vol. 4. 1.

[20] Mohammad Sadegh Safari and Vahid Pourahmadi. 2018. Deep UL2DL: Channel Knowledge Transfer from Uplink to Downlink. *arXiv preprint arXiv:1812.07518* (2018).

[21] Matilde Sánchez-Fernández, Mario de Prado-Cumplido, Jerónimo Arenas-García, and Fernando Pérez-Cruz. 2004. SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE transactions on signal processing* 52, 8 (2004), 2298–2307.

[22] Ralph Schmidt. 1986. Multiple emitter location and signal parameter estimation. *IEEE transactions on antennas and propagation* 34, 3 (1986), 276–280.

[23] Tan Shuang, Tommi Koivisto, Helka-Liina Maattanen, Kari Pietikainen, Timo Roman, and Mihai Enescu. 2011. Design and evaluation of LTE-Advanced double codebook. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*. IEEE, 1–5.

[24] Elahe Soltanaghaei, Avinash Kalyanaraman, and Kamin Whitehouse. 2018. Multipath Triangulation: Decimeter-level WiFi Localization and Orientation with a Single Unaided Receiver. In *Proceedings of the 16th Annual International Conference on Mobile Systems , Applications, and Services*. ACM, 376–388.

[25] Mehran Soltani, Vahid Pourahmadi, Ali Mirzaei, and Hamid Sheikhzadeh. 2019. Deep Learning-Based Channel Estimation. *IEEE Communications Letters* (2019).

[26] Sanjib Sur, Ioannis Pefkianakis, Xinyu Zhang, and Kyu-Han Kim. 2017. WiFi-assisted 60 GHz wireless networks. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. ACM, 28–41.

[27] Cisco Systems. 2018. Cisco Visual Networking Index: Forecast and Trends, 2017-2022. *White paper* (2018).

[28] David Tse and Pramod Viswanath. 2005. *Fundamentals of wireless communication*. Cambridge university press.

[29] Vamsi Krishna Tumuluru, Ping Wang, and Dusit Niyato. 2010. A neural network based spectrum prediction scheme for cognitive radio. In *2010 IEEE International Conference on Communications*. IEEE, 1–5.

[30] Deepak Vasisht, Swarun Kumar, and Dina Katabi. 2016. Decimeter-level localization with a single WiFi access point. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*. 165–178.

[31] 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*. ACM, 398–411.

[32] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* 13, 2 (2011), 22–30.

[33] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2019. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* 119 (2019), 3–11.

[34] Lu Wu, Jinhui Chen, Hongwei Yang, and Di Lu. 2011. Codebook design for cross-polarized linear antenna array in LTE-A downlink system. In *Vehicular Technology Conference (VTC Fall), 2011 IEEE*. IEEE, 1–5.

[35] Jie Xiong, Karthikeyan Sundaresan, and Kyle Jamieson. 2015. ToneTrack: Leveraging Frequency-Agile Radios for Time-Based Indoor Wireless Localization. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 537–549.

[36] Yi Xu, Guosen Yue, and Shiwen Mao. 2014. User grouping for massive MIMO in FDD systems: New design methods and analysis. *IEEE Access* 2 (2014), 947–959.

[37] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[38] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Juang. 2018. Power of deep learning for channel estimation and signal detection in OFDM systems. *IEEE Wireless Communications Letters* 7, 1 (2018), 114–117.