Reconfigurability in Wireless Networks: Applications of Machine Learning for User Localization and Intelligent Environment

Chuan Sun
Department of Electrical Engineering and Computer Science
University of Kansas
Email: chuansun@ku.edu

Submitted to the graduate degree program in the Department of Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Chair: Dr. Morteza Hashemi

Dr. Taejoon Kim

Dr. David Johnson

Date Defended: 30 April 2021

The thesis committee for Chuan Sun
certifies that this is the approved version of the following thesis:

Reconfigurability in Wireless Networks: Applications of Machine Learning for User Localization and Intelligent Environment

Chair: Dr. Morteza Hashemi

Co-Chair: Dr. Taejoon Kim

Co-Chair: Dr. David Johnson

Date Approved: 30 April 2021

Abstract

With the rapid development of machine learning (ML) and deep learning (DL) methodologies, the theoretical foundation of leveraging DL in wireless network reconfigurability and channel modeling is studied and summarized. While deep learning based methods have been applied in a few wireless network use cases, there is still much to be explored in many wireless channel modeling scenarios such as predicting channel state information (CSI), and configuring intelligent surface for optimum performance. In this paper, we perform an extensive research on the application of deep learning methods in reconfigurable wireless modeling problems which contains two scenarios. In the first scenario, a user transmitter was moving randomly within a campus area, and at certain spots sending wireless signals that were received by multiple antennas. We constructed an active deep learning architecture to predict user locations from received signals after dimensionality reduction, and analyzed 4 traditional query strategies for active learning to improve the efficiency of utilizing labeled data. We proposed a new location based query strategy that considers both spatial density and model uncertainty when selecting samples to label. We show that the proposed query strategy outperforms all the existing strategies. In the second scenario, a reconfigurable intelligent surface (RIS) containing 4096 tunable cells reflects signals sending from a transmitter to users in an office for better performance. We use the training data of one user's received signals under different configurations to learn the impact behavior of the RIS on the wireless channel. Based on the context and experience from the first scenario, we built a DL neural network that maps RIS configurations to received signal estimations. In a second phase back propagation, the loss function was customized towards our final evaluation formula in order to obtain the optimum configuration array for a user. A further research on the identification of line-of-sight (LOS) and none line-of-sight (NLOS) users has been conducted, which enabled us to prioritize NLOS users over LOS users in our loss function to maximize the final evaluation goal. We built a customized DL pipeline that automatically learns the behavior of RIS on received signals, and generates the optimal RIS configuration array for each of the 50 test users.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Driven by more and more evolving applications and ever-growing user demand, the research on wireless networks has been rapidly expanding with both deeper and broader integration with technologies in various domains like wireless physics, system design, computation, and data science. The ability for wireless networks to handle numerous, changing, and complex user scenarios has become a crucial requirement in both indoor and outdoor user scenarios. Wireless network reconfigurability, which enables wireless networks to reconfigure their hardware and software components, is therefore envisioned to adapt itself to the complex scenarios, enhance network performance and boost user experience.

In order to make full use of wireless network reconfigurability and obtain optimal network configurations, the use of machine learning (ML) technologies has attracted increasing research interest due to its decent performance in learning historical patterns and making future predictions. In recent years, deep learning (DL), one subfield of the ML algorithms, has made breakthrough in the model performance on many complex industry problems. DL based models can be leveraged to model the relation and variation between wireless channel parameters, learn the behavior pattern of reconfigurable wireless networks, and give optimized predictions on network reconfigurability.

In this project, we research and apply the use of ML methodology in modeling wireless channel parameters, and based on that, optimizing the configurations of an intelligent reconfigurable surface (IRS) for best user experience. We work through two concrete scenarios that step-by-step consolidate our system models and materialize our research purposes. The first scenario deals with the user localization problem, whose setup is based on a moving cart inside a campus area. The cart moves randomly along campus roads and transmits signals at certain location spots. The transmitted signals are received by an antenna array placed in different locations inside the campus, and recorded into the training data. Meanwhile, the ground truth of all locations are provided by a differential GPS in the format of x, y, and z coordinates. The goal of the scenario is to predict the coordinate of user locations based on the signals received from the antenna array. In order to predict user localization, we applied deep learning methodologies to build a neural network regression model that learns the pattern between the received high-dimensional signals and user location labels, and generates user localization accordingly on any given signals. Further from that, the problem of high-cost labeling is also researched in our case since it would be difficult and costly to obtain labeled wireless data to train supervised learning models for user localization

problems. We explored the use of active learning on top of our deep learning architecture for a more efficient use of the labeled data. We experimentally compared the effectiveness and performance of 4 commonly used query strategies for active learning in the user localization problem, and proposed a new location based query strategy that takes both spatial density and model uncertainty into consideration when choosing unlabeled samples. It is observed that our proposed query strategy outperforms all the other 4 existing strategies, and is capable of achieving a same level of model performance as fully-trained models using far less labeled samples.

In the second scenario, we move one step further towards wireless network reconfigurability. The setup involves an intelligent reflecting surface (IRS) containing 4096 configurable cells with each one in a state of either -1 or 1. The IRS is installed on a wall and reflects signals transmitted from a base station to a group of users in an office area. The direct path from the base station to all users is blocked, thus the IRS is deployed here to improve user experience. A number of pilot signals are sent from the base station to the users, and are provided as two datasets. Dataset 1 contains all the transmitted signals, IRS configurations, and their corresponding received signals on one single user. While dataset 2 provides all the same metrics on other 50 users. The purpose of the scenario is to find the best IRS configuration for each of the 50 users in dataset 2 that results in a maximum received data rate. To solve this problem, we leveraged the model architecture and training experience from the first scenario to train a generic deep learning based regression model that maps IRS configuration to channel estimation. The generic model is then respectively trained on the data of 50 users and specialized into 50 different models. We managed to obtain the optimal IRS configuration for each user through a back propagation process with customized layer structure and self-defined loss function. Moreover, we further explored the identification of line-of-sight (LOS) and none line-of-sight (NLOS) users in this scenario so as to fine tune the model performance for NLOS users to achieve better final data rate.

Chapter 2

Efficient User Localization Using Active Deep Learning

In this chapter, we investigate the problem of user localization in wireless networks using Active Deep Learning. We constructed a deep neural network to learn the channel state information collected by antennas, and implemented active learning mechanism to improve the efficiency of utilizing labeled data. The performance and behavior of different active learning query strategies are compared and analyzed. We propose a location based query strategy that considers both spatial density and model uncertainty when selecting samples to label. Experimental results show that the active learning methodology is capable of achieving a same level of performance as traditional models using far less labeled samples. In addition, the location based strategy outperforms all the other query strategies for the location prediction problem.

2.1 Introduction

The problem of user localization is increasingly drawing attentions from wireless communication researchers, and its applications span across various scenarios such as navigation [1], surveillance, Internet of Things (IoT) [2, 3], and so on. With the rapid development of machine learning methodology, the theoretical foundation of leveraging machine learning in wireless channel modeling is studied and summarized [4]. Much research efforts have been made on building supervised learning models on the labeled channel state information (CSI) data for various purposes. For instance, neural network has been exploited to predict unobserved wireless channel features [5], and predict wireless channel statistical parameters from the location information between transmitters and receivers [6]. [7] uses regression neural network to learn and estimate wireless channel state information from large datasets. The authors in [8] proposes a method that synthesized multiple machine learning models to enhance the prediction on user locations. [9] compares the performance of 3 neural network models with traditional methods for predicting localization with noisy measurements, and finds that a radial based neural network outperforms all the other models.

In recent years, deep learning [10] has gained significant popularity due to its performance in complex machine learning scenarios such as image and video content extraction, cognitive

modeling, autopilot, etc. The application of deep learning in wireless localization has also prevailed as hot research interests [11]. Different algorithms and model architectures have been explored in both indoor and outdoor localization problems. In [12], received signal strength (RSS) and CSI data are used to train convolutional deep networks that predict indoor object locations. Zhang et al. [13] proposed a deep neural network with a Hidden Markov Model (HMM) based localizer for wireless positioning. The authors in [14] compared the performance of common supervised learning and deep learning models on indoor localization, and found that deep learning model outperforms all the others.

While traditional machine learning models are broadly applied and have achieved state-of-the-art performance in these scenarios, the performance of supervised learning models largely depends on the amount of labeled training data. Large amounts of labeled data are typically required to train a well-fit model, while obtaining labeled data could involve immense manual labor and is therefore a considerably costly task. In addition, since the channel information data in our localization problem is in high dimensions, it could be time consuming to train models on the full dataset over a number of iterations. Active learning [15], as a promising new research topic, has been proposed to improve the efficiency of utilizing labeled data by selecting the most valuable samples to train the model, and the method for determining samples is called query strategy. The statistical basics of active learning are researched and summarized in [16], and it has been already applied in some traditional machine learning models like SVM [17] and decision trees [18] to reduce the number of labels needed, as well as some deep learning structured models for tasks such as image processing [19–21].

In this chapter, we explore the use of active learning methodologies on the localization problem. Our goal is to improve the performance of traditional machine learning approaches in the following ways.

- **Less labeled data needed:** The active learning approaches utilize a query strategy that iteratively determines the most valuable samples that could help improve model performance [22]. These samples are handed to an oracle for labeling and then fed into the model for training. This mechanism boosts the efficiency of utilizing the limited amount of labeled data, and as a result makes it possible to reduce the number of labeled data required to achieve the same performance.

- **Less model training time:** Active learning typically works under scenarios of limited amount of labeled data, and functions by tactically finding the most valuable samples to label and to improve itself on. Thus, active learning models would fit less data samples than traditional models that trains on the entire training dataset at each iteration.

We investigate the user localization problem based on active learning mechanisms. We compare the effectiveness of several commonly used query strategies, and propose a new location based query strategy to further boost the performance of location prediction.

The remainder of this chapter is organized as follows. Section 2.2 provides an overview of the user localization problem definition and the high level structure of our solution model. Section 2.3 takes a deeper look into the active learning mechanism and its query strategies. The experimental setup, results and analysis are presented in Section 3.5. Finally, Section 3.6 concludes the paper.

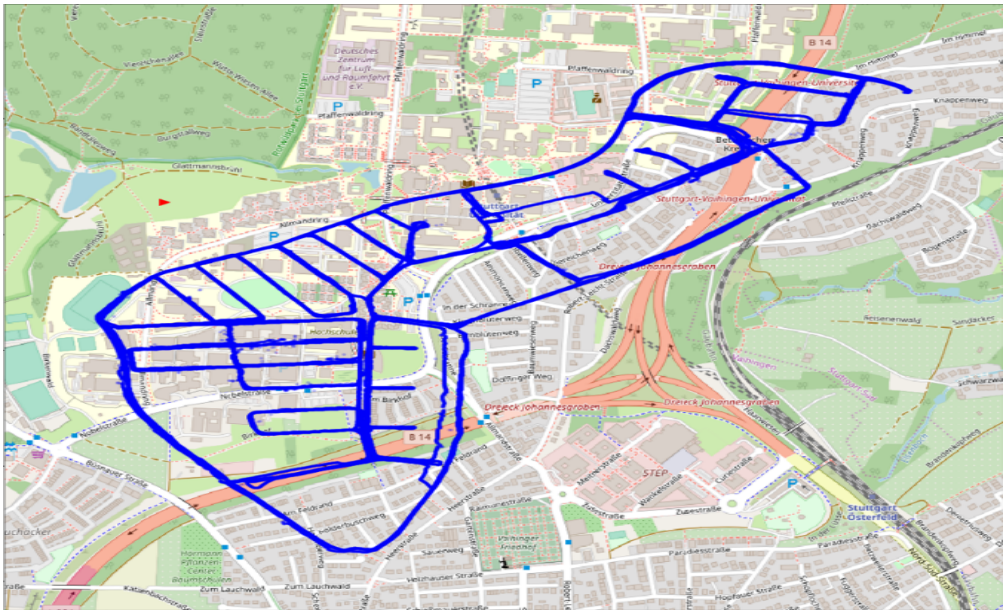

Figure 2.1: Ground truth locations of the moving cart recorded by GPS. The intermittently recorded locations form into a blue route inside the campus. This data is provided in the dataset of [23].

2.2 Problem Statement and System Model

2.2.1 Problem Definition

In our scenario, a transmitter, which is an SDR-equipped cart, moved randomly along campus roads inside a university residential area of several hundred square meters. At certain location spots, the channel responses were collected by an antenna array and recorded in the training data. The ground truth for each location, which is a 3-dimensional coordinate in the format of x, y and z, is provided by a differential GPS. Figure 2.1 shows the ground truth of all locations provided by GPS on a 2-D plane. This measurement and dataset are provided by [23].

2.2.2 Data Sources and Format

The data used in this chapter is from the IEEE CTW 2020 data competition on user localization. The dataset was acquired by the massive MIMO channel sounder [24] measuring outdoor channel responses between a moving transmitter and an 8x8 antenna array (only 56 out of the overall 64 antennas work properly). The channel information data collected for each location spot consists of 924 usable subcarriers, 56 functioning antennas and 5 channel measurements. There were 4979 location samples collected in total, forming into a dataset in the shape of $4979 \times 56 \times 924 \times 5$. Meanwhile, the signal-to-noise ratio data for each location, antenna and measurement was provided in a separate file. A ground truth file containing the x, y and z coordinates of all the 4979 locations was provided as labels.

Given the raw data files, our goal is to train a model that learns the pattern between

the CSI input and location coordinates, and predicts user locations based on incoming CSI data.

2.2.3 System Model

In this chapter, we build a data ingestion and modeling workflow to process the high dimensional CSI input and make location predictions using a deep learning model and active learning query strategies.

Dimensionality Reduction

In order to keep the model training process inside an acceptable period of time, the dimensionality reduction technique is applied to reduce the magnitude of our training data. We implemented an autoencoder [25], which consists of an encoder and a decoder, structured in Figure 2.2.

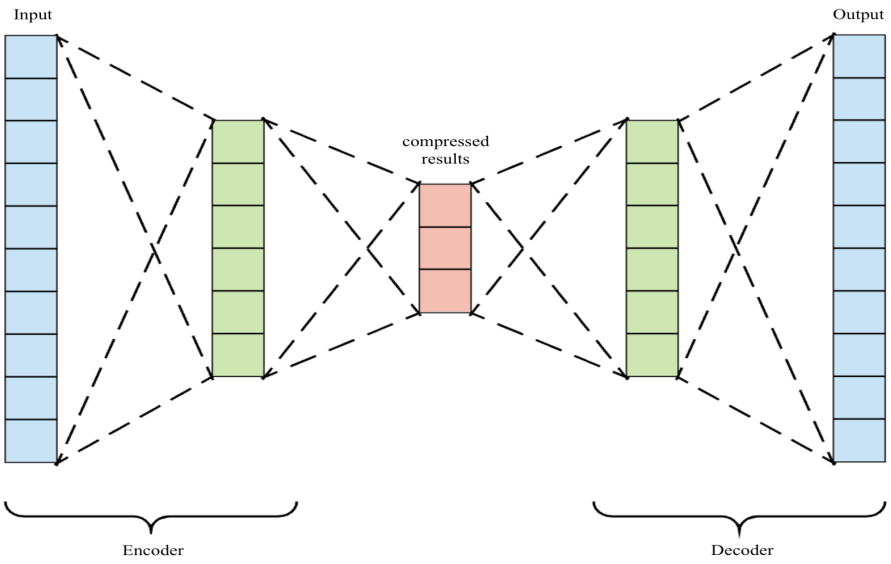

Figure 2.2: The structure of an autoencoder. The autoencoder consists of several encoding layers that gradually decrease the number of neurons, followed by decoding layers that increase the number of neurons back to original.

The encoder is a multi-layered neural network containing 3 convolutional 2D layers that gradually compress the input dimension of subcarriers from 925 to 37. While the subsequent decoder decodes the most compressed layer back to the original dimensions using a reversed network structure. The combined encoder-decoder model was trained to fit the original input data that is labeled by itself, so that the model would minimize the difference between the original input and the one going through our process of both compression and decompression. Finally after the training completes, the encoder part of the model was extracted and used to reduce the dimensionality of our input.

Neural Network Model

In our scenario, the localization problem could be formulated as a regression task based on the CSI input data and the coordinate labels. We adopt a multi-layered neural network as the basic learner of our regressor to fit the high dimensional CSI input, where all of our further active learning work is based.

The architecture of the basic model is illustrated in Figure 2.3, which takes the input data that already has its dimensionality reduced, and consists of 3 convolutional 1D layers and 3 dense layers to accept input and extract high level information. The output of the network consists of 3 nodes, representing the predicted coordinate in x, y and z axes, respectively. Based on the model tuning on our training data, we apply batch normalization for dense layers to stabilize training performance, and adopt ReLU activations for both convolutional and dense layers. The Adam optimizer is chosen as the best algorithm for calculating gradients.

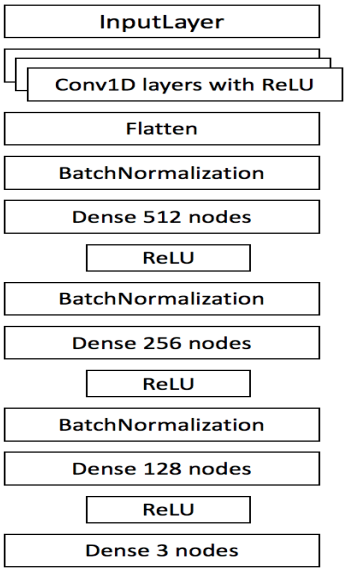

Figure 2.3: The architecture of the basic learner. A convolutional 1D layer receives the input. 3 dense layers with their neurons gradually shrinking broadens the network and connects to a 3-node output layer. ReLU activation is used for each dense layer, and batch normalization is used for each layer except for the last dense layer.

2.3 Active Deep Learning Framework

In this section, we present our active learning framework.

2.3.1 Active Learning Preliminaries

For many real-world machine learning tasks, the number of unlabeled data could easily outmatch that of the labeled ones, while obtaining labels for the unlabeled data could often

be of high cost. In recent years, active learning has evolved as an important branch of supervised learning to deal with scenarios where there are only limited amount of labeled data, while a large number of unlabeled data are available. Active learning improves the efficiency of utilizing labeled data and maximizes model performance by selecting certain samples from the unlabeled data pool to label before training. The selection is based on certain strategies that are called query strategies. The chosen samples are then labeled by an oracle or external human annotators, and fed back to the model to improve its performance. Figure 2.4 demonstrates the process of a typical active learning cycle.

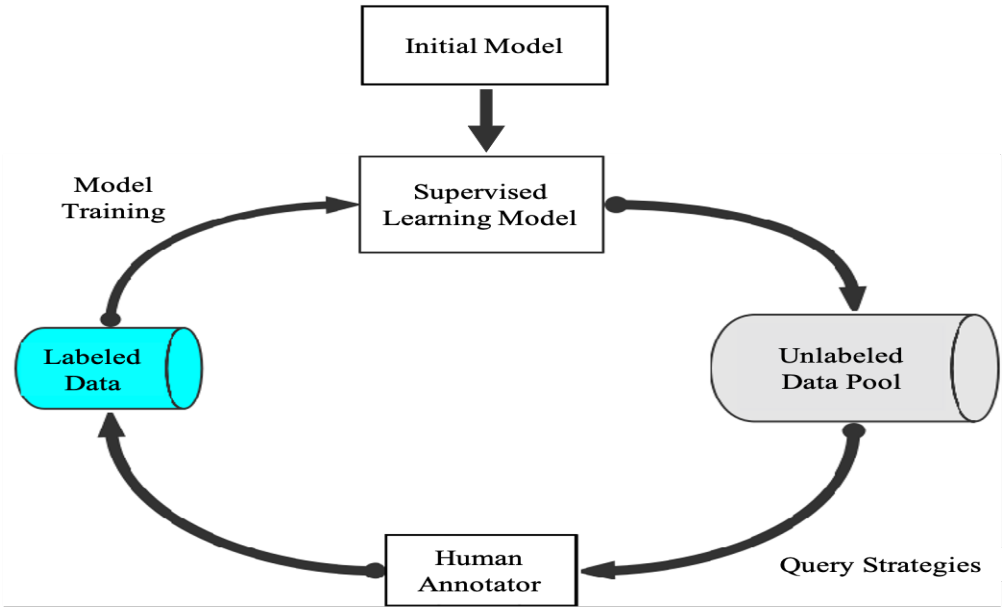

Figure 2.4: The active learning cycle. A query strategy is used to select unlabeled data from the pool for human annotation. The labeled data is then fed into the training of supervised learning models.

For each training iteration, the query strategy determines the most valuable samples for the model to learn, which is a decisive part in the active learning cycle. We will take a deeper look at different query strategies in the next section.

2.3.2 Query Strategies

There are several commonly used query strategies that aim at determining the most valuable samples for a human annotator to label.

Random Sampling

The random query strategy randomly selects samples from the unlabeled data pool without any preference. This strategy could typically serve as a benchmark and owns its advantage of being computationally efficient.

Least Confident

How uncertain the model is on the unlabeled samples could serve as a selection criterion [26]. Different measurements on model uncertainty result in different implementations of uncertainty query strategies. A commonly-used uncertainty measurement metric is based on the model's probability of its output, and selects samples with least confidence, as described in 2.1.

$$x_{LC} = \underset{x}{\operatorname{argmax}}\, 1 - P(\hat{y}|x), \tag{2.1}$$

where $\hat{y}$ is the model's output with highest probability.

Margin Sampling

A variation of this strategy, which is called the margin sampling strategy, calculates the margin of each sample as the difference between its top probability and its second top probability, and selects samples with smallest margin as formulated in 2.2.

$$x_{MS} = \underset{x}{\operatorname{argmin}}\, P(\hat{y}_1|x) - P(\hat{y}_2|x), \tag{2.2}$$

where $\hat{y}_1$ and $\hat{y}_2$ are the model's output with top and second top probability respectively.

Entropy Sampling

The entropy based strategy is a more generalized form of measuring uncertainty based on information theory, which calculates impurity in 2.3:

$$x_{ES} = \underset{x}{\operatorname{argmax}} - \sum_i P(\hat{y}_i|x) \log P(\hat{y}_i|x), \tag{2.3}$$

where $\hat{y}_i$ iterates through all possible model outputs.

Location Based Sampling

In this paper, based on the context and pattern of localization raw data, we propose a new query strategy described in 2.4.

$$x_{LB} = \underset{x}{\operatorname{argmax}}(1 - P(\hat{y}|x)) * (\sum_i x - xc_i), \tag{2.4}$$

where xc is the set of all labeled samples, and $P(\hat{y}|x)$ is the model's probability output given an input x. This query strategy takes both model uncertainty and location density into consideration, and selects the sample that maximizes both the model's uncertainty, and the distance from any labeled samples in the space of input dimensions, with its measurement defined by the Euclidean distance. Compared with traditional uncertainty based strategies, the location based strategy adds the capability of selecting spatially balanced samples. This could help boost performance when the samples that the model is most uncertain of agglomerate together and cannot provide more global information.

In this paper, we refer to the least confident query strategy as "uncertainty sampling". We compare the performance of 5 active learning models that adopt these 5 query strategies respectively: random sampling, uncertainty sampling, margin sampling, entropy sampling and location based sampling.

2.4 Numerical Results

2.4.1 Experimental Setup

We conduct data preprocessing, feature engineering and dimensionality reduction on the channel response data, and feed the processed data into our active learning architecture to analyze the effect of different query strategies and the pattern of samples chosen by different strategies.

Data Preprocessing

We calculate the Euclidean norm and the phase of each complex number in the channel response data, and stack them together with the provided signal noise ratio to form into a $4979 \times 56 \times 925 \times 10$ CSI dataset. This dataset then goes through the autoencoder dimensionality reduction process that reduces the dimension of subcarriers from 925 to 37. We split out 10 percent of overall data for test set, and within the training set, we further split out 80% for the labeled query pool that serves as the oracle, as shown in Figure 2.5. This means that the amount of labeled data for active learning is only 20% of that for a traditional supervised learning case.

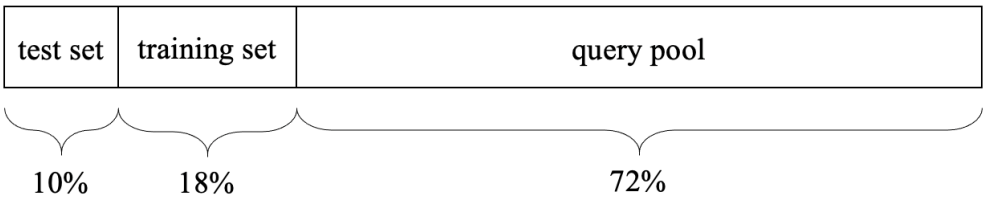

Figure 2.5: The split of the dataset. 10% of total labeled data is used for test set. Among the 90% training data, 80% of that forms a query pool for active learning, and the remaining 20% is used for model training.

Learner Model

The base learner for our user location regression problem is a multi layer convolutional neural network. The architecture of the model is illustrated in Figure 2.3. The mean squared error (MSE) is used to evaluate model performance in our research, which can be denoted as following. Given that an input data has the true label $\mathbf{y}$ and predicted label is $\hat{\mathbf{y}}$, then the MSE is defined as:

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{n} ((\mathbf{y}_x - \hat{\mathbf{y}}_x)^2 + (\mathbf{y}_y - \hat{\mathbf{y}}_y)^2 + (\mathbf{y}_z - \hat{\mathbf{y}}_z)^2), \tag{2.5}$$

where $\mathbf{y_x}$, $\mathbf{y_y}$ and $\mathbf{y_z}$ are the true label coordinates on the x, y and z axis, respectively. n is the size of the test set.

Active Learning Process

We customized an active learning architecture that pre-trains a basic learner using a given amount of labeled data. Then for each training iteration, the active learning mechanism chooses a batch of samples from our query pool according to different query strategies. The query pool comes from our labeled dataset so the samples automatically gets labeled, and the model then re-trains itself given these newly labeled data. Five query strategies are applied and compared here: random sampling, uncertainty sampling, margin sampling, entropy sampling and a proposed location based sampling. The changes of model performance over training epochs under each query strategies were recorded for further analysis.

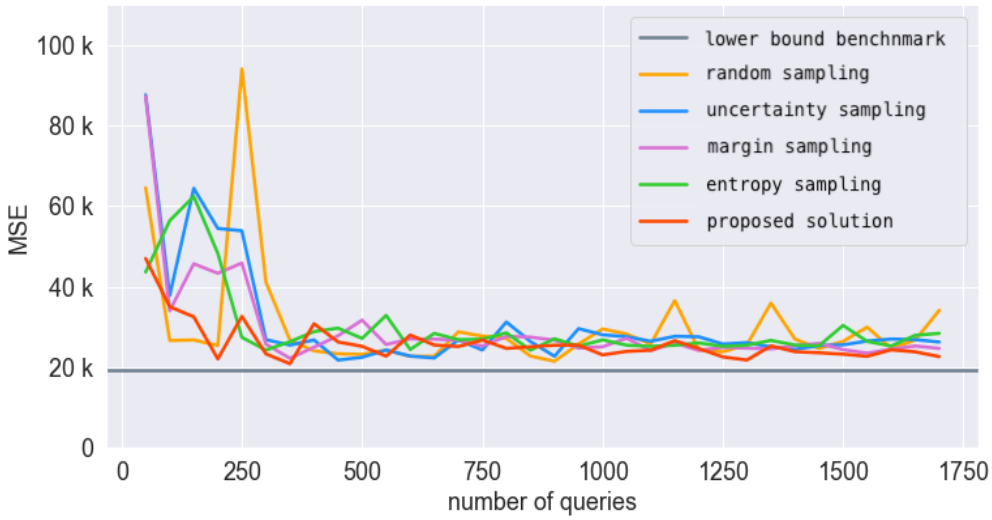

Figure 2.6: MSE change over epochs under different query strategies. The location based query strategy approaches closer to the lower bound benchmark with less training epochs than all the other query strategies.

2.4.2 Performance Results

Figure 2.6 illustrates how the MSE changes over training epochs under all 5 query strategies. The gray horizontal line is the performance of a traditional model trained on the full labeled dataset using same iterations, which is a lower bound benchmark. It could be observed that most strategies tend to converge after around 500 epochs, and the random sampling appears more unstable throughout the training. The proposed location-based strategy shows a fast convergence of 250 queries, with a rather good and stable prediction loss.

In addition, Figure 2.7 plots the cumulative distribution function of the MSE under each query strategy. The location based strategy outperforms all the others with a larger proportion of low MSE values. The random sampling strategy, on the other hand, contains larger amounts of big MSEs which indicates a more unstable fitting process.

Besides model performance, we explored into the samples chosen by each query strategy as well. Figure 2.8 visualizes all the chosen samples on a 2-d plane within the first 250 queries. The larger a dot is plotted, the earlier it is chosen by the strategy.

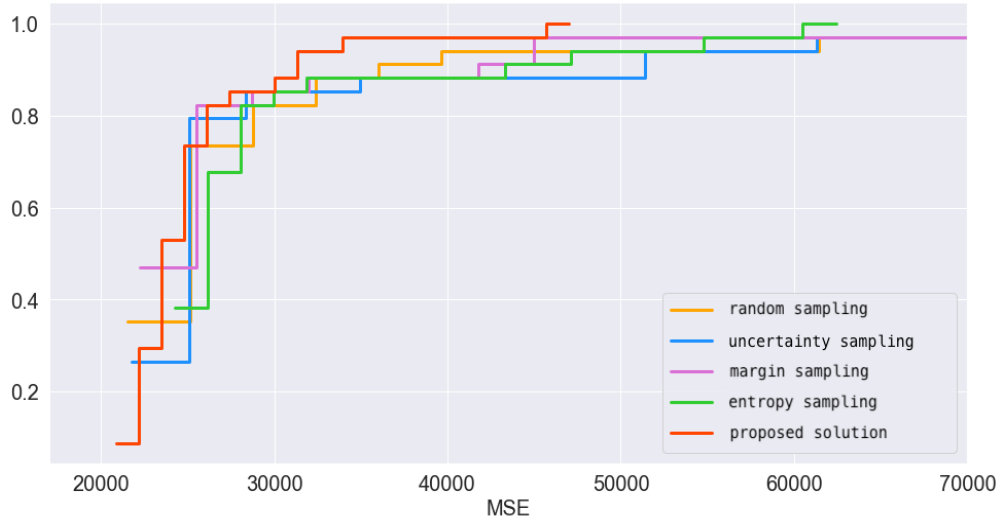

Figure 2.7: The cumulative distribution function plot of MSE. Location based sampling displays a polyline closer to the left, indicating a larger proportion of low MSE values compared with other query strategies.

The margin and entropy sampling strategy display an early preference over the samples on the right and left part of the map respectively. While the early samples chosen by the uncertainty sampling and the proposed location based strategy turn out to fall into both left and right part of the map more evenly, demonstrating a more balanced behavior.

Figure 2.9 demonstrates the prediction results by both active learning and non active learning models after 25 epochs of training. The location based active learning methodology manages to control its prediction scope within a same level of non active learning results, given far less available labeled training samples. Moreover, the active learning model even generates comparatively more scattered predictions than the traditional method, thanks to its consideration for location density in the proposed query strategy.

The kernel density estimate (KDE) on the Euclidean distance between predictions and true labels is plotted in Figure 2.10. From the graph, the proposed location based strategy distributes closer to zero than all the other strategies, indicating the smallest prediction error on the test set.

2.5 Conclusion

In this chapter, we explored the use of active learning methodology in the user localization problem. We processed the raw channel state information data collected by antennas, trained an autoencoder to reduce the dimensionality of the CSI data, and applied active learning methodologies on top of a multi-layered 1D convolutional neural network. We applied several commonly used query strategies, and proposed the location based query strategy for choosing samples to label.

Experimental results demonstrate that active learning is capable of approaching a similar performance as traditional supervised learning models using much less labeled data. For

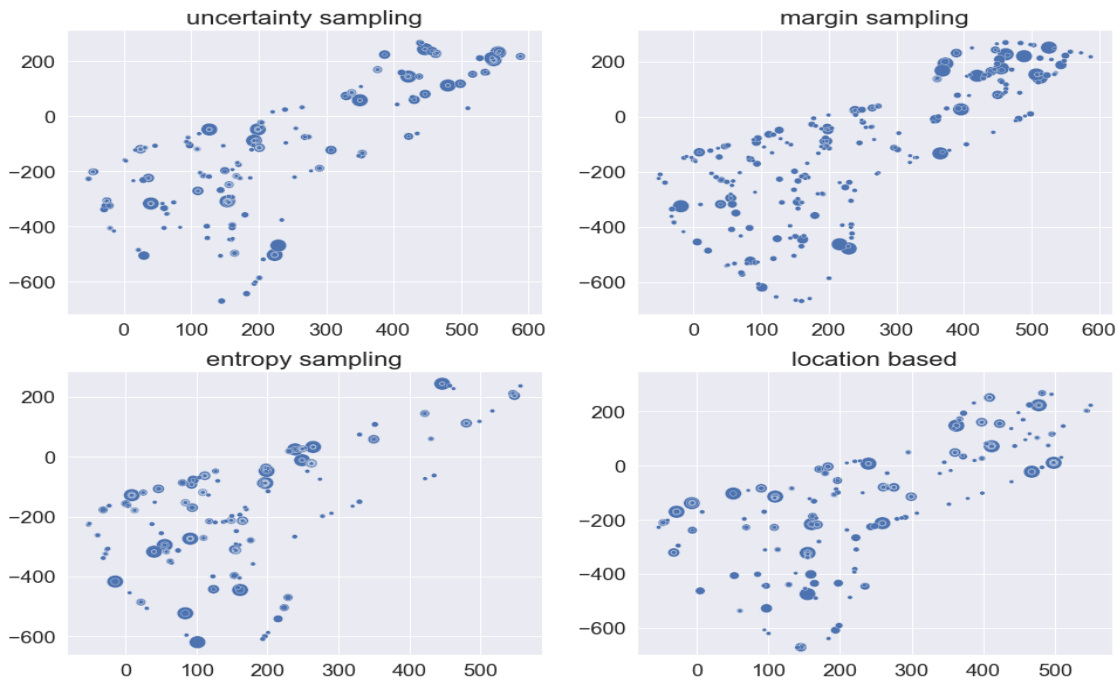

Figure 2.8: The chosen samples by different query strategies on 2-d plain. Larger dot size indicates an earlier selection by the strategy. Location based sampling displays a more balanced large dot distribution, meaning that the strategy is able to quickly feed the model with evenly distributed data.

query strategies, the location based strategy turns out to outperform all the other strategies with fast convergence and the lowest prediction error. We plotted and studied the pattern of chosen samples by each query strategy, which provides useful information on how they correlates with the model's final predictions. Future work includes studying the data behavior in certain areas of the route map to further improve the accuracy of prediction.

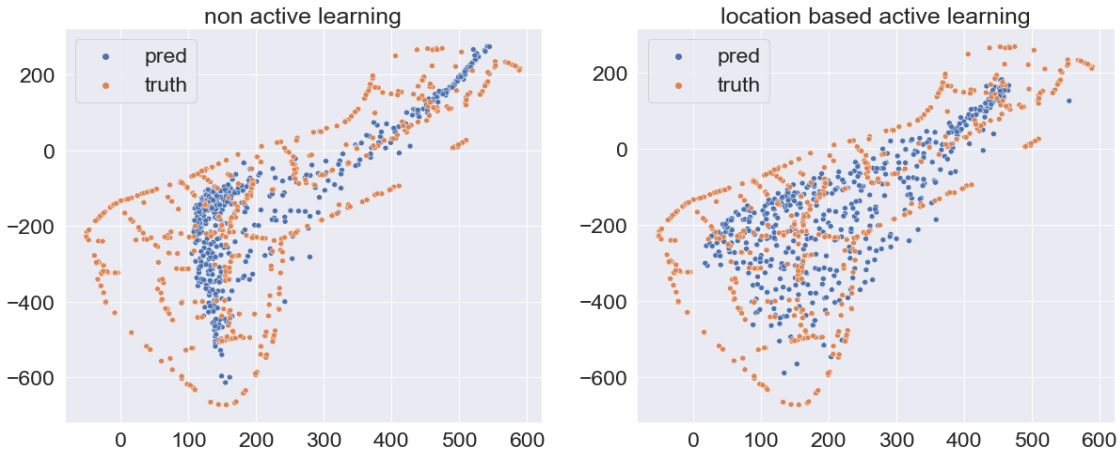

Figure 2.9: Location predictions by both the active learning and non active learning method after 25 training epochs. Compared with traditional supervised learning methods, the location based active learning makes more scattered predictions closer to the ground truths.

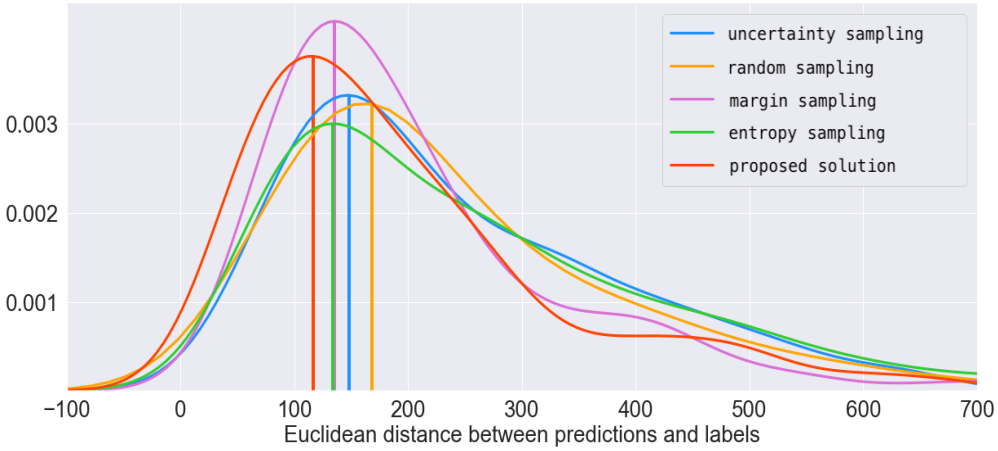

Figure 2.10: KDE plot of the Euclidean distance between predictions and true labels. The location based query strategy outperforms all the other query strategies with smaller prediction error.

Chapter 3

Optimal Configuration of Intelligent Reflecting Surfaces (IRS)

In this chapter, we research and solve the problem of optimizing an intelligent reflecting surface (IRS) for different users using deep learning based methodologies. We trained a regression deep learning network to predict the communication channel parameters given a 4096-element IRS configuration vector. We further re-trained this base model based on the data of 50 different users, and performed a back propagation process using a customized loss function formulated by our final evaluation data rate. We managed to output the optimal IRS configuration for a specific user that maximizes its received data rate, given the received signals of the user.

3.1 Introduction

In wireless network, optimizing the received signals at user side has been an important topic in many practical scenarios. For scenarios where a direct path between the base station and users is not available, the intelligent reflecting surface (IRS) can be utilized to reflect the signals from base stations in a controllable way so that the signals received at user locations are constructively combined and thus providing better user experience. An IRS is a two dimensional array of cells whose material can interact with electromagnetic waves. This interaction can be controlled, for example by tuning the impedance variations over the surface. These tunable surfaces can be utilized to direct wireless signals sent from a transmitter towards receivers.

The study on how to optimally tune the cells in IRS in order to achieve the best user experience (e.g., best receiving data rates) has recently attracted much attention in industry. [27] compared the accuracy, tractability and hardware complexity of 3 different IRS models, and investigated technical challenges of their efficient design respectively. [28] studies the effectiveness of IRS under imperfect channel state information (CSI), and proposes an algorithm that considers both phase shift and amplitude variation to mitigate CSI errors. Going further from these theoretical researches, more and more researches focus their interest on applying IRS into practical scenarios to improve the performance and user experience of wireless networks. [29] studies the rate enhancement on a single-user communication system

after deploying 2 IRSs. [30] explores and enables the use of IRS into integrated air-ground wireless networks by jointly applying IRS and unmanned aerial vehicle (UAV) to enhance the communication performance.

With the fast development and increasing flourish of machine learning (ML), especially deep learning (DL) technology, leveraging ML methods into IRS based setups has drawn huge research interest in scenarios such as predicting channel state, localizing users and devices, and optimizing received signals. [31] synopsizes the use of DL techniques in IRS system design such as signal detection and channel estimation. The authors in [32] explore and develop DL techniques to tune the reflection of IRS elements and maximize receiving rates. [33] proposes a deep reinforcement learning-based methodology to optimize the security rate of a highly dynamic wireless system where legitimate users and eavesdroppers both exist. In these researches, the architecture of deep learning network is utilized to predict or approach ideal IRS parameters to wireless channel users based on some level of constraint.

In the scenario of this chapter, an IRS is installed on a wall which reflects the signals between a base station and the users in an office area, where direct path between the two is not available. The IRS is a 4096-element array with each element being a binary switch. We are provided with the data [34] of pilot signals sent from the base station, the configuration values of the IRS each time a pilot signal is sent, and their corresponding received signal at a specific user.

The problem to solve here is to obtain the best IRS cell configurations for 50 different users given some known configurations and their corresponding received signals on these users. We assemble and propose a DL based methodology that builds a regression DL network to learn the impact of IRS configurations on channel behavior, and then finds the optimal IRS configuration specifically for each user by optimizing a customized loss function.

The remainder of this chapter is organized as follows. Section 3.2 provides an overview of the IRS scenario and formulates the problem we target. The section also describes the detailed structures of the two datasets provided in this scenario. Section 3.3 gives a high level structure of system model solution and different training phases to solve the problem, and the details of each DL training phase is elaborated in Section 3.4. The experimental setup, results and analysis are presented in Section 3.5. Finally, Section 3.6 concludes the paper.

3.2 Problem Statement

In this section we provide a context description on the concrete problem on IRS optimization, as well as the detailed datasets we work on.

3.2.1 Problem Definition

In this wireless communication setup, a single-antenna base station serves user devices located in an office area [34], as shown in Figure 3.1. The direct path between the base station and the users is blocked by a wall. To improve the signal propagation, an IRS was deployed at a fixed location on a wall, which consists of 4096 unit cells. The IRS cells do not process or amplify signals, but only diffusely reflect radio waves coming from the base station. Each

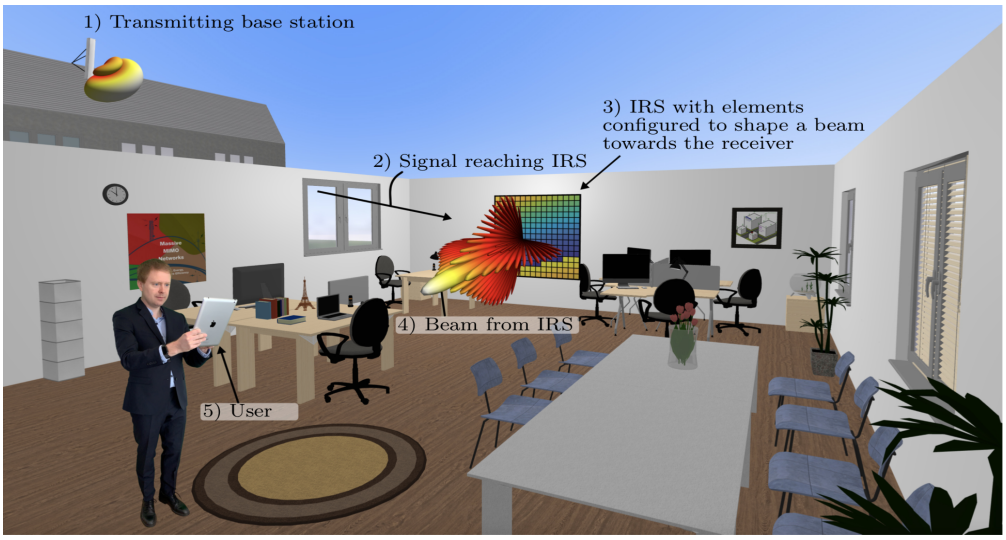

Figure 3.1: Wireless network setup [34] of the IRS problem. An IRS on the wall reflects the signals sent from the base station to users in an office. Since the direct path from the base station and users is blocked, IRS is used for improving user experience.

unit cell can be tuned by an impedance controlled by a two-state switch, meaning that each cell can tuned to be either -1 or 1. The general purpose of the scenario is to tune the 4096 cell vector in such a way that it results in an optimal signal propagation to users. The evaluation criterion for a user is based on the received data rates that this user achieves with his specific θ configurations, computed as:

$$R = \frac{B}{K+M-1} \sum_{v=0}^{K-1} log_2 \left(1 + \frac{P|\bar{h}_\theta[v]|^2}{BN_0}\right), \tag{3.1}$$

where P is the signal power, B is the bandwidth, M is the number of taps, K is the number of OFDM subcarriers, and N_0 is the noise power spectral density. In this problem, P, B, M and K are known constants, while N_0 is unknown.

There are 50 test users in total, and it is most likely that each user has a specific IRS configuration. In this scenario, there are two user categories: line-of-sight (LOS) users and non-line-of-sight (NLOS) users. The weight for NLOS users will be doubled at final evaluation since they would typically have low rates, but the problem doesn't disclose which users belong to which category. To summarize and formulate our problem, we need to find 50 best θ configurations, one for each user, so as to maximize 3.2.

$$R_{\text{weighted average}} = \frac{1}{50}\Big[\sum_{LOS\ users} R_{user} + 2 \sum_{NLOS\ users} R_{user}\Big], \tag{3.2}$$

where R_{user} is the rate for each user as defined in 3.1.

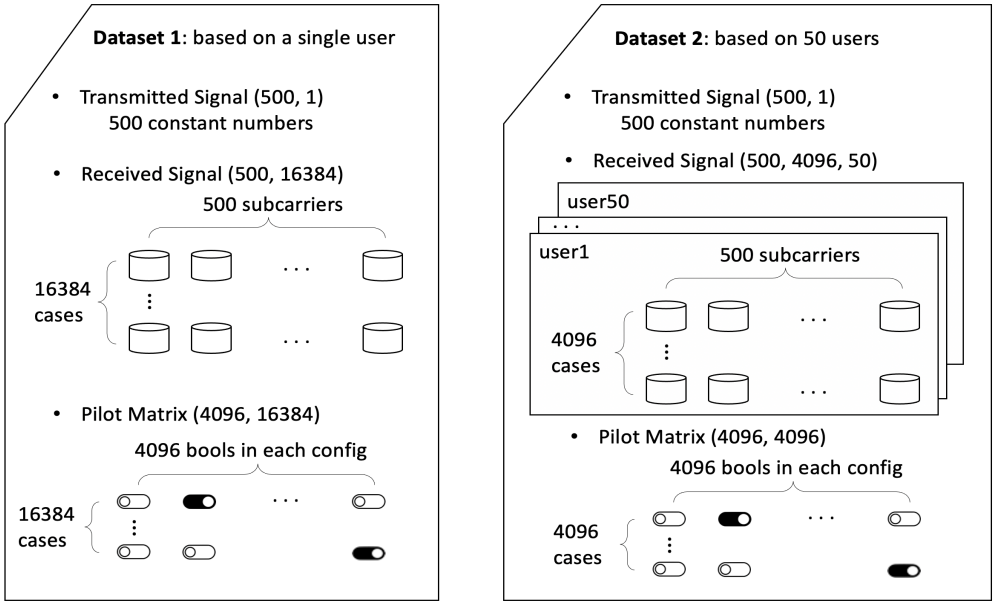

Figure 3.2: Structures of the 2 datasets provided in [34]. Dataset 1 contains 16384 samples of transmitted signals, θ configurations and received signals on a single user. Dataset 2 provides 4096 same metrics on 50 different users each.

3.2.2 Data Sources and Format

The data source of the setup is officially provided by IEEE Signal Processing Cup 2021 [34], which consists of two separate data files, as shown in Figure 3.2. Dataset 1 provides all the data of a single user scenario, including 16384 different IRS configuration values and their corresponding received signals at a fixed user side. Dataset 2 provides info on 50 different users, where 4096 different IRS configurations and their corresponding received signals are provided for each of the 50 users. Each IRS configuration is in the form of a 4096-element array with each one being a binary number (-1 or 1), and all the received signals are in complex number form. The transmitted signals in both dataset 1 and 2 are unchanged constants.

3.3 System Model

The relation between the transmitted signals and the received signals in time domain can be denoted as:

$$y[k] = \sum_{l=0}^{M-1} h_\theta[l]x[k-l] + w[k], \tag{3.3}$$

where M is the number of channel taps, which equals to 20 in our problem, $h_\theta[l]$ is the finite impulse response (FIR) filter describing the communication channel, and $w[k]$ is the receiver noise. The relation between the input and received signals can be written in short

form as:

$$\underbrace{\begin{bmatrix} \bar{y}[0] \\ \vdots \\ \bar{y}[K-1] \end{bmatrix}}_{=\bar{y}} = \underbrace{\begin{bmatrix} \bar{h_\theta}[0] \\ \vdots \\ \bar{h_\theta}[K-1] \end{bmatrix}}_{=\bar{h}_\theta} \odot \underbrace{\begin{bmatrix} \bar{x}[0] \\ \vdots \\ \bar{x}[K-1] \end{bmatrix}}_{=\bar{x}} + \underbrace{\begin{bmatrix} \bar{\omega}[0] \\ \vdots \\ \bar{\omega}[K-1] \end{bmatrix}}_{=\bar{\omega}}, \tag{3.4}$$

where K is the number of subcarriers, which is 500 in our case, $\bar{x}$ is the transmitted discrete-time signal, $\bar{h}_\theta$ is the finite impulse response (FIR) filter describing the communication channel, $\bar{\omega}$ is the receiver noise, $\odot$ denotes the Hadamard product, and $\bar{y}$ is the received discrete-time signal. This relation can be written in short as:

$$\bar{y} = \bar{h}_\theta \ \odot \ \bar{x} \ + \ \bar{\omega}. \tag{3.5}$$

Different IRS configuration θ could result in different channel vectors $\bar{h}_\theta$, and consequently affecting the signals received at user side. To estimate the impacts of channel and IRS configurations on received signals, a known pilot signal vector $\bar{x}$ is transmitted, and the $\bar{h}_\theta$ estimations can be extracted from the noisy received signal $\bar{y}$. Here, the values of $\bar{x}$, θ and $\bar{y}$ are provided in dataset 1 on a single user case (16384 observations), as well as in dataset 2 for each of the 50 users (4096 observations each).

However $\bar{h}_\theta$ depends on both the configuration θ and the propagation channel. To model the relation between these two, we constructed a deep learning network that utilizes dataset 1 to learn the behavior and impact of θ configurations on vector $\bar{h}_\theta$. Then dataset 2 is used to do customized training based on the θ and $\bar{h}_\theta$ from each of the 50 users. Finally, a further round of back propagation using a customized loss function generates the optimal configuration for each user.

Since the final evaluation of the problem doubles the rate for NLOS users, we explored around our data to identify NLOS users from the mixed 50 users, and worked on an additional round of model tuning for NLOS users to boost their performance.

3.4 Deep Learning Framework

Several deep learning neural networks are trained in this paper, assembled as different components of our entire solution pipeline depicted in Figure 3.3. They will be elaborated respectively in this section.

Received Signal Denoising

According to Eq. 3.5, the received signal $\bar{y}$ contains a noise term $\bar{\omega}$. So given an array $\bar{y}$, we need a denoising process to approximate the value of $\bar{h}_\theta \ \odot \ \bar{x}$. In this paper, we applied a denoising process that first adds Gaussian noise to our $\bar{h}_\theta$ vectors, then trains an autoencoder regression network that minimizes the mean squared error (MSE) between the noised $\bar{h}_\theta$ and the original $\bar{h}_\theta$. The trained autoencoder can then be used to denoise all the $\bar{h}_\theta$ given in both dataset 1 and 2. The autoencoder network consists of 6 dense layers, utilizing a symmetric two-part architecture, with the first part gradually shrinking in layer size, and the second part increasing its size back to original. Each dense layer is followed by a Rectified Linear

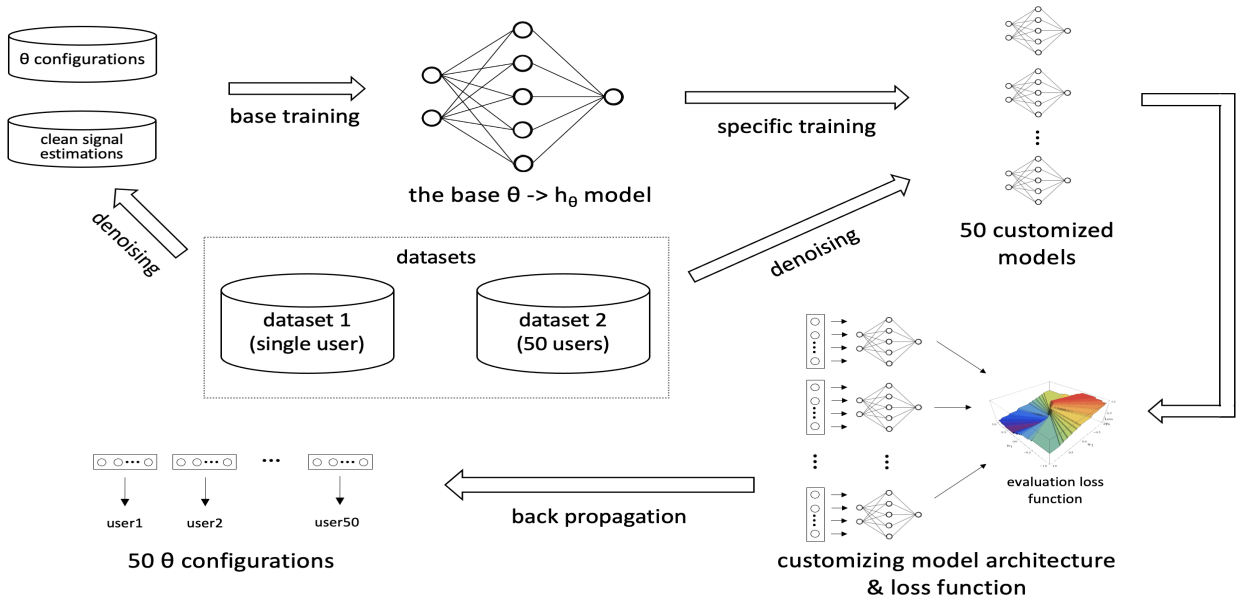

Figure 3.3: The proposed deep learning solution pipeline. We use dataset 1 to train a generic model that maps θ to $\bar{h}_\theta$, and use dataset 2 for specific model training and tuning. A final round of back propagation is conducted to obtain the optimal θ for each user.

Unit (ReLU) activation and a batch normalization (BN) to make the fitting process more stable. The architecture of the autoencoder is illustrated in Figure 3.4.

Regression DL Network

To model the mapping from θ configurations to $\bar{h}_\theta$ vectors, we constructed a DL regression network which inherits a similar architecture as the deep learning model used in the user localization scenario. The training contains two phases. For the first phase, we train a generic regression network from dataset 1. The inputs of the generic model are in the shape of a 4096-element θ configuration vector. The inputs go through several dense layers with their size shrinking from 512 to 64, together with a ReLU activation and batch normalization. A final 500-neuron dense layer outputs the squared magnitudes of each $\bar{h}_\theta$ estimation for all 500 subcarriers as a vector. Figure 3.5 displays the architecture of the DL network.

We copied the weights of the generic model for each of the 50 users, and conduct a second phase of precise training based on the specific data for each user in dataset 2. After both training phases, we get 50 different models for 50 users, which are ready for the next optimization phase.

Customized Back Propagation

The 50 DL networks trained above enables us to generate $\bar{h}_\theta$ vector estimations given any θ configuration. While our goal is to find out which θ best maximizes our predefined evaluation formula and lands the best data rate R for each user. To achieve this, we built a third training phase on each of the 50 models, with several differences listed below:

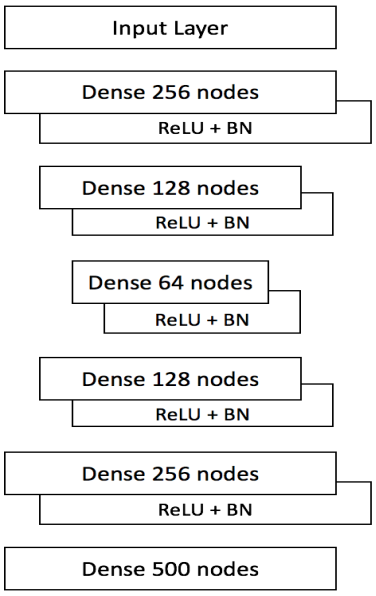

Figure 3.4: The architecture of the autoencoder network. This is a similar autoencoder structure as described in figure 2.2 with a size-shrinking encoder and a size-growing decoder.

- A new layer containing 4096 neurons is added before the entire network, with all its weights constrained between -1 and 1. This layer represents the θ configuration we would like to get as our final result.
- All the other layers of the network is fixed and non-trainable, so that we are simply utilizing the mapping trained from the second phase, and only optimizing the first layer.
- The loss function of the network is customized into Eq. 3.1, with a negative sign assigned ahead so that minimizing the loss function actually maximizes Eq. 3.1. The back propagation process ensures that the weights of the added layer converge towards the minimum loss, and thus the maximum data rate.

When the third phase is trained through back propagation, the weights of the first added layer gradually changes towards the direction that makes formula 3.1 larger. In this way we are getting better evaluation metric values with the weights evolving. Since the 4096 weights are limited between -1 and 1, we could finally obtain the θ configuration by thresholding the weight values into either -1 or 1.

3.5 Numerical Results

Data Preprocessing

For both datasets 1 and 2, the vectors of $\bar{x}$ and $\bar{y}$ are extracted, and a noised estimation of $\bar{h}_\theta$ could be obtained from 3.5, as:

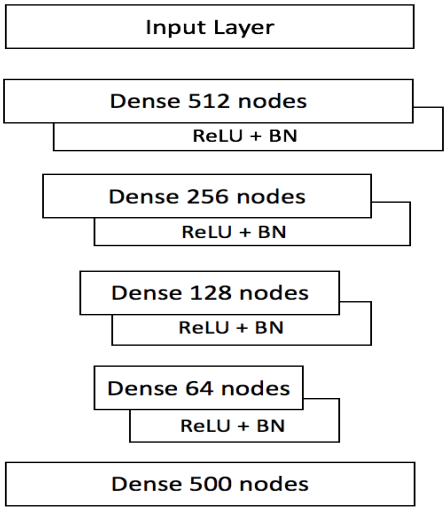

Figure 3.5: The architecture of the deep learning regression network. 4 dense layers gradually decrease their neuron size with a ReLU activation and batch normalization applied. A final output layer of 500 neurons generates the squared magnitude array of a given θ over 500 subcarriers.

$$(\bar{h}_\theta)_{noised} = diag(\bar{x})^{-1} \cdot \bar{y} \tag{3.6}$$

The noised estimations of $\bar{h}_\theta$ are then denoised using the autoencoder network mentioned above. All the 16394 θ configurations and their corresponding denoised $\bar{h}_\theta$ vectors are train-test splitted with a test size ratio of 0.2, and then fed into the 3-phase deep learning training pipeline.

Deep Learning Results

The deep learning regression model accepts θ configurations as inputs and outputs the squared magnitudes of the estimated $\bar{h}_\theta$. Figure 3.6 displays the fitting process of the generic regression model in phase 1. It could be observed that the model smoothly converges after around 50 - 60 epochs without over-fitting.

This model represents the relation between θ and $\bar{h}_\theta$ estimations trained on the single user that dataset 1 provides. To further get a more precise relation between θ and $\bar{h}_\theta$ of the test 50 users, we copied this base model for each of the 50 users, and further trained these 50 models on the specific data provided in dataset 2. Figure 3.7 displays the second training process on 9 randomly picked users. To fully approach the best performance for each user, we explored different neural network optimizer methods respectively for every user. Based on our experiment results, the convergence process of each user's network optimization varies in terms of its speed and lowest loss value. Each model will gradually converge to its best loss value without over-fitting.

Now that we are able to predict $\bar{h}_\theta$ given any θ configuration, we conduct a final round of customized back propagation to obtain the best θ configuration that generates the largest data rates for each user. Figure 3.8 depicts the back propagation process on the same users

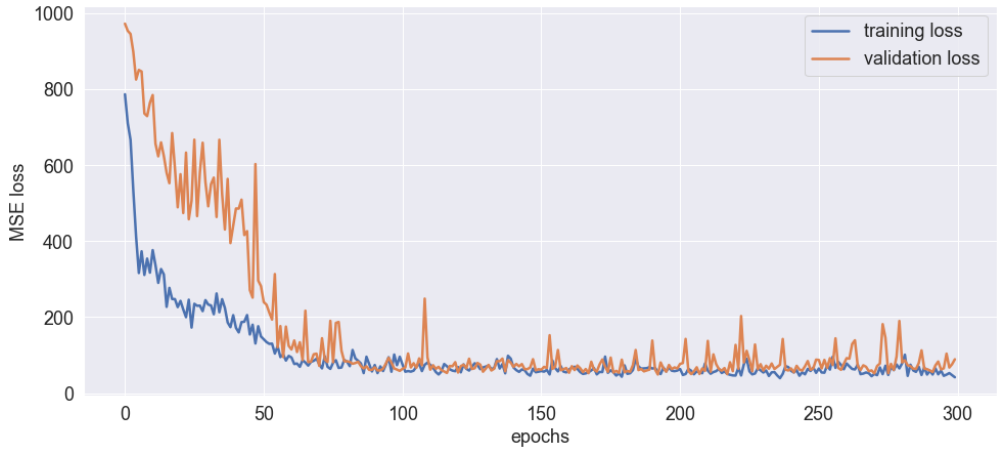

Figure 3.6: The fitting process of the deep learning regression model. The validation loss decreases quickly and finally converges with the training loss, demonstrating a good data fitting process without over-fitting.

as in Figure 3.7. This round of back propagation quickly approaches the best MSE loss, which is the negative of our final evaluation data rate, on most of the users. Both training loss and validation loss decline to the same level of loss values, enabling us to approach the highest data rates for each user.

Figure 3.9 plots the histogram of the weight distribution across all 50 users. The back propagation process changes and optimizes the weight of the added 4096-neuron layer towards the minimum loss, and we can observe that the weight constraint preset between -1 and 1 on this layer nicely forces most of the weights to be closely distributed either near -1 or +1, which is very close to the binary results we're targeting in our problem. This is a result of the weight constraint set at the first layer of the back propagation process. Based on these weight values, we applied a mapping rule that maps negative weights to -1, and positive weights to 1 to form our final θ configuration values.

The calculated optimal rates are compared with several benchmark rates as below:

- A randomly generated θ configuration where all the values follow the uniform distribution.
- The best θ configuration detected in the given dataset 2 after exhaustive search using our regression model and Eq. 3.1, denoted as θ_{ds2}.
- The configuration with all one values.
- The configuration with all -1 values.

Figure 3.10 illustrates the performance of the best θ configuration we obtain, together with the benchmarks mentioned above. It could be observed that the best θ configuration obtained by our pipeline significantly outperformed randomly generated configurations and the configurations with all 1 and -1 values. It also outperforms the detected θ_{ds2} on almost every one of the 50 user scenarios. Furthermore, Table 3.1 provides a quantified comparison

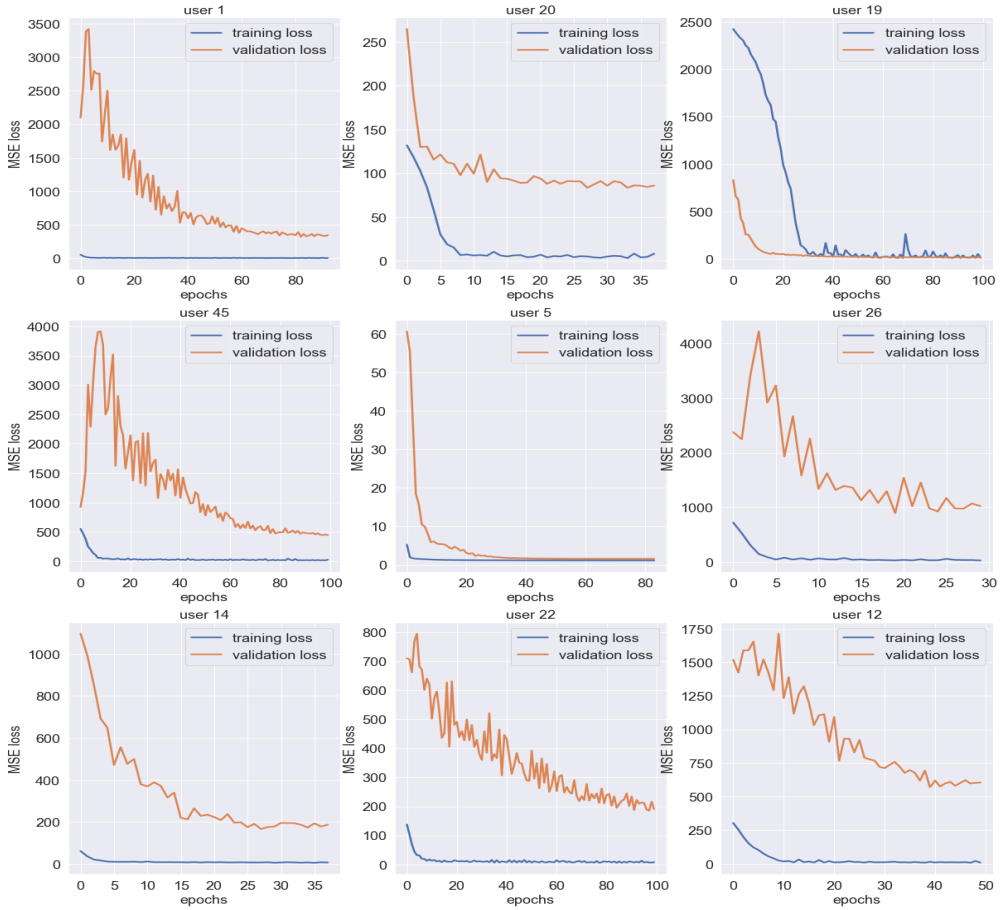

Figure 3.7: The specific fitting processes for 9 randomly selected users. This process further improves model performance by fitting the data of 50 specific users. Models of most users display a smooth converging process, with validation loss decreasing and approaching training loss without over-fitting.

on the average final rate across all users achieved by different θ configurations. Here we set θ_{ds2} as the comparison baseline. Experimental data shows that the best average rate by our method achieves a performance boost of 5.22% against the baseline.

The rate results also give us some insights on the information of LOS/NLOS users. The blue line in figure 3.10 clearly demonstrates two different value patterns. Peaks can be observed such as for user 1, 4, 5, etc. which indicate a high data rate for LOS users. On the other hand, some users such as 3, 6, 7, etc. display approximately a half rate value, which indicates a NLOS scenario.

NLOS Identification

Since the NLOS users will have a doubled weight over LOS users during data rate evaluation, it is necessary to identify NLOS users from all the 50 users. To verify our previous findings and identify the pattern of NLOS users, we further explored the signals received by each user. The magnitude and angle of the received signals of all 50 users are analyzed, and we

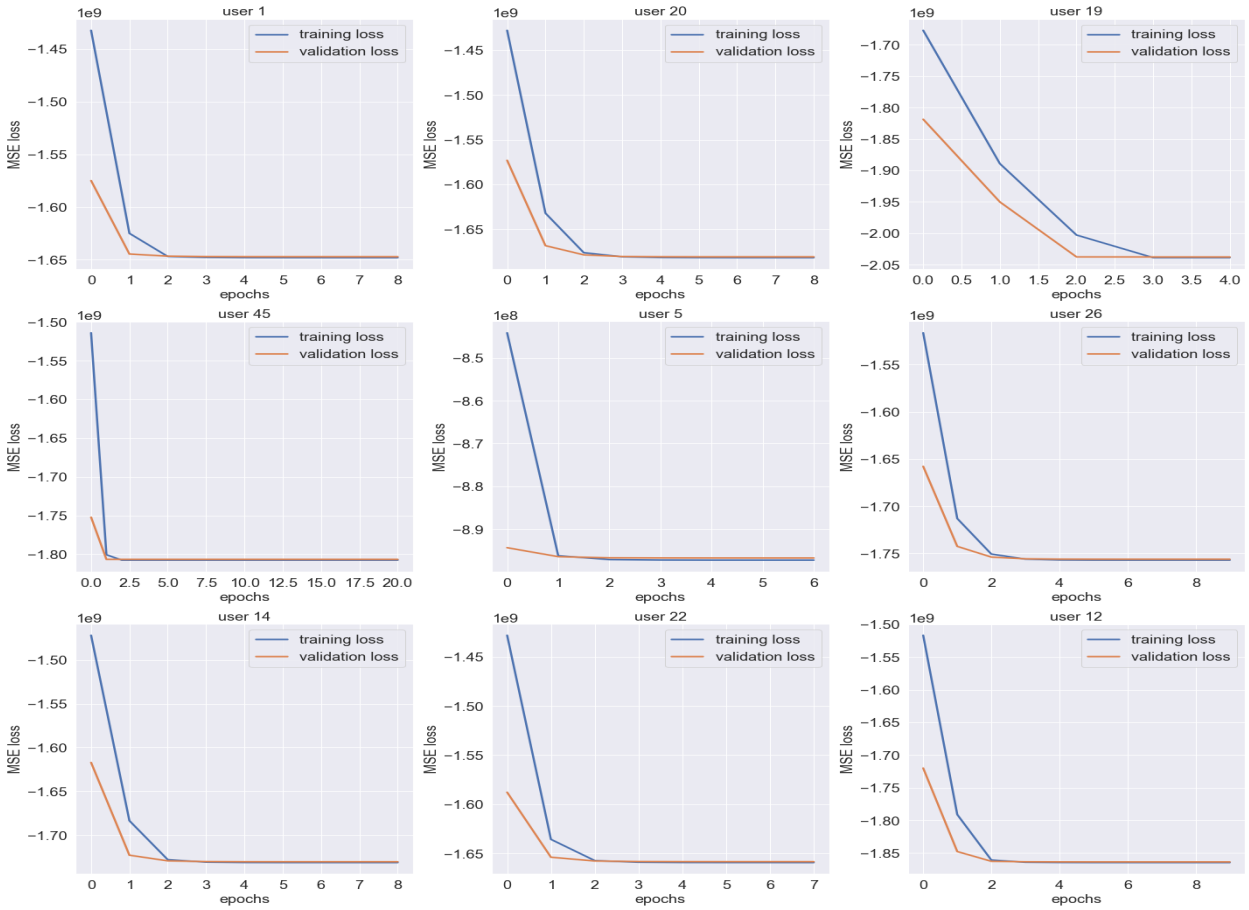

Figure 3.8: The customized back propagation process on 9 randomly selected users. The process finds the optimal θ configuration for each user. Figures show that the validation loss of users models are able to achieve the same minimum value of training loss, which verifies the efficacy of our back propagation process for finding the optimum.

observe that they fall into 2 different behavior categories, as plotted in Figure 3.11. The left plot shows the group of users where signal magnitudes are quite small and don not vary much across different angle phases. While the users in the right group tend to have a much larger magnitude variance compared with the left group.

We find that the users that appear in the left part of Figure 3.11 exactly match the low-rate users detected in Figure 3.10. This verifies our NLOS finding in the final evaluation rates, and drives us towards further model tuning and future improvements designed and aimed for these NLOS users.

3.6 Conclusion

In this chapter, we applied the deep learning methodologies to solve the problem of optimizing IRS configurations for specific users. We assembled and proposed a solution that consists of 3 phases: an initial training of a generic deep learning regression model that maps θ

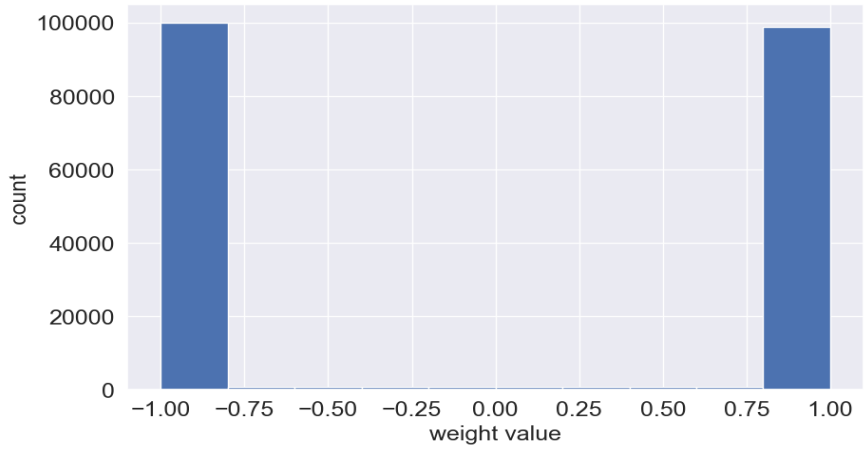

Figure 3.9: The distribution histogram of the weights across all users. Most values distributes very close to either -1 or 1, which is a result of our weight constraint during the back propagation process.

θ Value	Avg Rate(Mbps)	Weighted Avg Rate(Mbps)	WAR % Against θ_{ds2}
all -1s	32.34	40.04	-61.51%
all 1s	31.21	39.47	-62.07%
randomly generated	35.49	43.07	-58.60%
θ_{ds2}	90.66	104.04	—
proposed solution	95.39	110.49	6.21%

Table 3.1: Performance comparison between the θ obtained from our proposed solution and from other benchmarks. The θ configuration obtained by our proposed solution outperformed the one obtained from exhaustive search of dataset 2 by 6.21% in terms of the weighted average rate.

configurations to $\bar{h}_\theta$ estimations; a customized re-training process on each of the 50 users; and the final back propagation process using a customized loss function to obtain the optimal θ configuration.

The first and second phase aim at generating $\bar{h}_\theta$ estimations given any θ configuration, while the third phase enables us to find the optimal θ to maximize our final evaluation data rate for each of the 50 users. In this way, we are able to give the best θ configuration vector for a specific user, as long as its received signals are provided. Moreover, we conduct analysis on the NLOS identification problem, and verifies the detected NLOS users in our final obtained rates. This enlightens our future research work of further performance boost on NLOS user patterns so as to achieve better overall rate and better user experience.

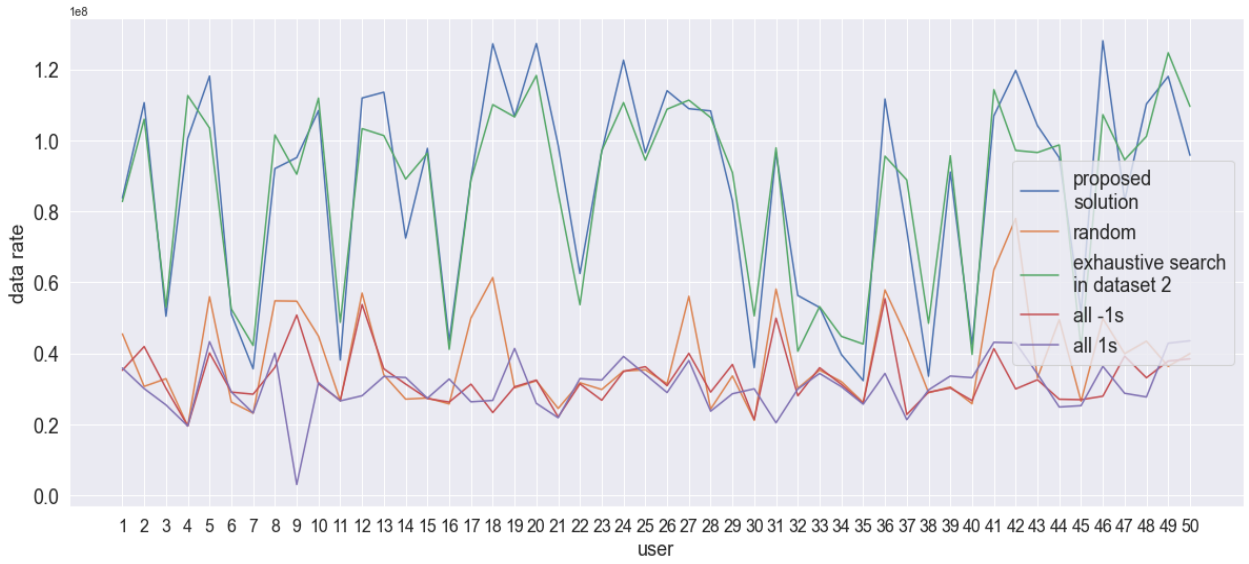

Figure 3.10: The final data rates of different θ configurations. The blue line is the θ obtained by our DL pipeline, and the green line is the θ_{ds2} baseline. The other 3 benchmark rates are also plotted in the figure. From the results, we observe that our solution significantly outperforms the 3 benchmark rates, and achieves better rates than the baseline θ_{ds2} in almost every user scenario.

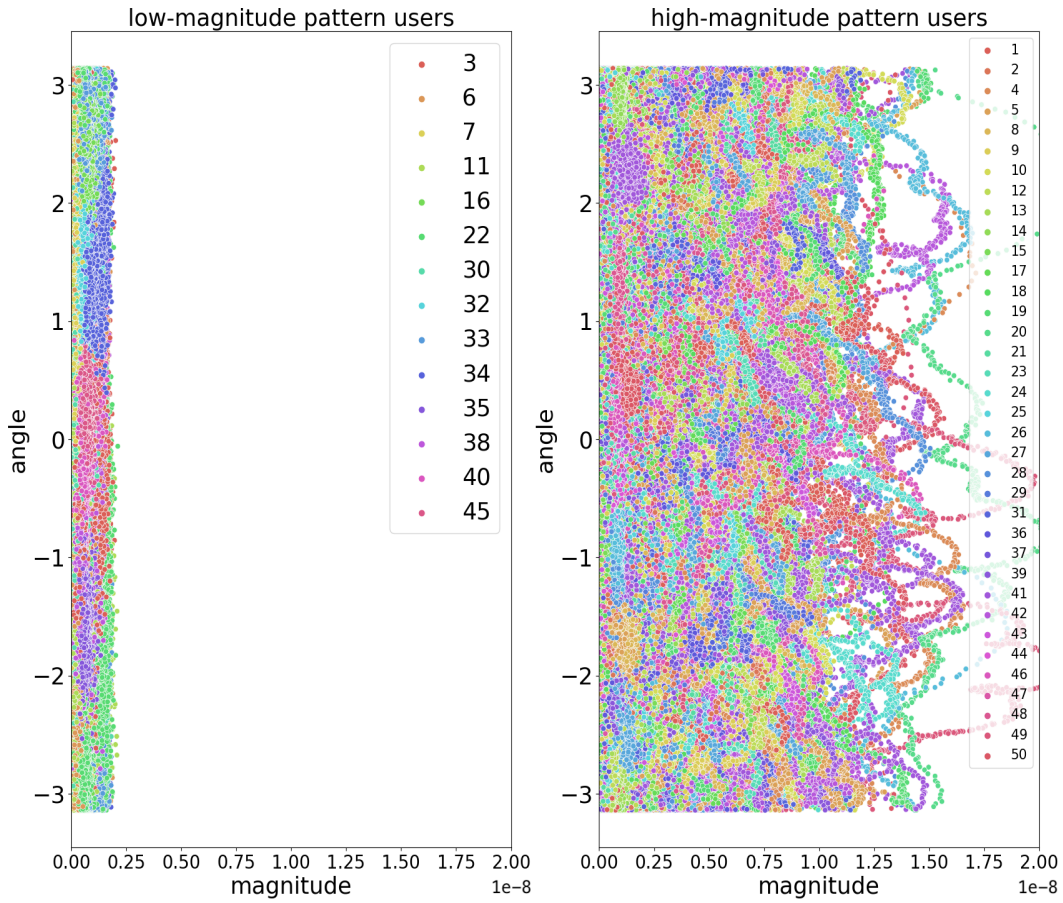

Figure 3.11: Two different pattern groups of user received signals. Patterns on the left demonstrate a small magnitude and barely vary with angle. Patterns on the right display much larger magnitudes and much more active variations across angles.

Bibliography

[1] R. Ayyalasomayajula, A. Arun, C. Wu, S. Sharma, A. R. Sethi, D. Vasisht, and D. Bharadia, *Deep Learning Based Wireless Localization for Indoor Navigation*. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3372224.3380894

[2] F. Khelifi, A. Bradai, A. Benslimane, P. Rawat, and M. Atri, "A survey of localization systems in internet of things," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 761–785, 2019.

[3] S. Sadowski and P. Spachos, "Rssi-based indoor localization with the internet of things," *IEEE Access*, vol. 6, pp. 30 149–30 161, 2018.

[4] S. Aldossari and K.-C. Chen, "Machine learning for wireless communication channel modeling: An overview," *Wireless Personal Communications*, vol. 106, 05 2019.

[5] S. Navabi, C. Wang, O. Y. Bursalioglu, and H. Papadopoulos, "Predicting wireless channel features using neural networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[6] L. Bai, C.-X. Wang, J. Huang, Q. Xu, Y. Yang, G. Goussetis, J. Sun, and W. Zhang, "Predicting wireless mmwave massive mimo channel characteristics using machine learning algorithms," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[7] T. O'Shea, K. Karra, and T. C. Clancy, "Learning approximate neural estimators for wireless channel state information," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017, pp. 1–7.

[8] L. Li, X. Guo, and N. Ansari, "Smartloc: Smart wireless indoor localization empowered by machine learning," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 8, pp. 6883–6893, 2020.

[9] A. Shareef, Y. Zhu, and M. Musavi, "Localization using neural networks in wireless sensor networks," in *Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*. Citeseer, 2008, pp. 1–7.

[10] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135

[11] Y.-X. Ye, A.-N. Lu, M.-Y. You, K. Huang, and B. Jiang, "Wireless localization based on deep learning: State of art and challenges," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[12] C. Hsieh, J. Chen, and B. Nien, "Deep learning-based indoor localization using received signal strength and channel state information," *IEEE Access*, vol. 7, pp. 33 256–33 267, 2019.

[13] W. Zhang, K. Liu, W. Zhang, Y. Zhang, and J. Gu, "Deep neural networks for wireless localization in indoor and outdoor environments," *Neurocomputing*, vol. 194, pp. 279–287, 2016.

[14] Z. Turgut, S. Üstebay, G. Z. G. Aydın, and A. Sertbaş, "Deep learning in indoor localization using wifi," in *International Telecommunications Conference*. Springer, 2019, pp. 101–110.

[15] B. Settles, "Active learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 6, no. 1, pp. 1–114, 2012.

[16] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.

[17] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of machine learning research*, vol. 2, no. Nov, pp. 45–66, 2001.

[18] Y. Gu, D. Zydek, and Z. Jin, "Active learning based on random forest and its application to terrain classification," in *Progress in Systems Engineering*. Springer, 2015, pp. 273–278.

[19] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1183–1192.

[20] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin, "Cost-effective active learning for deep image classification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2591–2600, 2017.

[21] L. Yang, Y. Zhang, J. Chen, S. Zhang, and D. Z. Chen, "Suggestive annotation: A deep active learning framework for biomedical image segmentation," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2017, pp. 399–407.

[22] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A survey of deep active learning," 2020.

[23] I. C. Data Competition, "Self-supervised learning for user localization," IEEE Communications Theory Workshop, 2020.

[24] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An overview of massive mimo: Benefits and challenges," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 742–758, 2014.

[25] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232 – 242, 2016, roLoD: Robust Local Descriptors for Computer Vision 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231215017671

[26] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 1070–1079.

[27] X. Yu, V. Jamali, D. Xu, D. W. K. Ng, and R. Schober, "Smart and reconfigurable wireless communications: From irs modeling to algorithm design," 2021.

[28] M. M. Zhao, Q. Wu, M. J. Zhao, and R. Zhang, "Irs-aided wireless communication with imperfect csi: Is amplitude control helpful or not?" in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[29] C. You, B. Zheng, and R. Zhang, "Wireless communication via double irs: Channel estimation and passive beamforming designs," *IEEE Wireless Communications Letters*, vol. 10, no. 2, pp. 431–435, 2021.

[30] C. You, Z. Kang, Y. Zeng, and R. Zhang, "Enabling smart reflection in integrated air-ground wireless network: Irs meets uav," 2021.

[31] A. M. Elbir and K. V. Mishra, "A survey of deep learning architectures for intelligent reflecting surfaces," 2020.

[32] Y. Song, M. R. A. Khandaker, F. Tariq, K.-K. Wong, and A. Toding, "Truly intelligent reflecting surface-aided secure communication using deep learning," 2021.

[33] H. Yang, Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu, "Deep reinforcement learning-based intelligent reflecting surface for secure wireless communications," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 375–388, 2021.

[34] E. Björnson, "Configuring an intelligent reflecting surface for wireless communications," 2021.