Power Level Control in High-Density Cellular Communication Systems: a Deep Learning Based Approach

by

Ramin Hamedi Zaviehgard

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science

Department of Electrical and Computer Engineering
University of Toronto

© Copyright by Ramin Hamedi Zaviehgard 2018
Power Level Control in High-Density Cellular Communication Systems: a Deep Learning Based Approach

Ramin Hamedi Zaviehgard

Master of Applied Science

Department of Electrical and Computer Engineering
University of Toronto

2018

Abstract

Effective power control, while essential for the operation of cellular communication systems, is most challenging in high-density cellular systems as the computational complexity for determining optimal base station transmission power levels grows exponentially with the number of system base stations and channel variation rate. A power level control scheme for high-density cellular systems, based on recent advances in machine learning and neural networks, is proposed in this thesis. Optimal power levels are computed using a greedy search in an offline phase to train a neural network predicting optimal power levels during the online phase. Multiple neural network structures are studied, along with regularization and optimization methods for fast and efficient training, the effect of the training size on the performance and effect of granularity of transmission power levels. Provisions for improving user quality of service are also introduced through a modified optimization objective function targeting minimum user throughput rates.
Acknowledgments

I would like to first and foremost thank my supervisor, Professor Elvino Sousa, for providing the vision and freedom to conduct my research and being a role model for me as a mentor and supervisor. Great thanks to Professor Antonio Capone, and my professors at Politecnico di Milano for training me and making it possible to complete my masters work in Canada. Special thanks to Dr. Ahmed Alsohaily, for his great support, help, and mentorship during my progress.

Words can hardly express my gratitude to my dearest mother, my late father, and my beloved sister, whose continuous and selfless support, encouragement, and advice has made it possible for me to pursue my goals.

I would like to express my gratitude to professor Mahmoud Shahabadi, and Abbas Chimeh, whose mentorship during my undergraduate studies gave me the courage to tackle any problem.

Thanks to my friends at University of Tehran, Politecnico di Milano, and University of Toronto, especially to Marc for his sincere friendship and amazing sense of humor, I am not Tabi Marc! Yes, it is still possible, to Alireza Tajfar, and Uni, for their friendship, and great and kind advice and help over the many years.
# Table of Contents

Acknowledgments..................................................................................................................... iii  

Table of Contents....................................................................................................................... iv  

List of Tables ............................................................................................................................ vii  

List of Figures ........................................................................................................................... viii  

List of Abbreviations and Symbols........................................................................................... xiv  

1 Introduction............................................................................................................................ 1  

1.1 Contributions................................................................................................................... 4  

1.2 Organization.................................................................................................................... 5  

2 Literature Review..................................................................................................................... 6  

2.1 Main Inter-Cell Interference Management Techniques.................................................... 6  

2.1.1 Interference Cancellation ............................................................................................. 6  

2.1.2 Inter-Cell Interference Coordination........................................................................ 7  

2.1.3 Distributed Interference Management........................................................................ 7  

2.1.4 Power Control for Inter-Cell Interference Management............................................. 8  

2.2 Neural Networks Overview ............................................................................................. 8  

2.2.1 Supervised Learning .................................................................................................... 9  

2.2.2 Stochastic Gradient Descent ....................................................................................... 9  

2.2.3 Non-Linear Activity Functions and Deep Architectures............................................ 10
2.2.4 Neural Networks ................................................................. 11
2.2.5 Batch Gradient Descent ....................................................... 18
2.2.6 Training, Validation, and Test Procedure .......................... 18
2.2.7 Overfitting and Early Stopping ............................................ 18
2.2.8 Dropout .............................................................................. 19
2.2.9 Momentum Technique and Adam ....................................... 20
2.2.10 Residual Connections ....................................................... 21
2.2.11 Shared Weights ................................................................. 22

2.3 Related Work ......................................................................... 23

3 System Model, Simulation Assumptions and Related Work ........ 27
3.1 System model ......................................................................... 27
3.2 Simulation Assumptions ........................................................... 32

4 Enhanced Neural Network Design for Power Control .............. 34
4.1 Activation Function ................................................................. 34
4.2 Greedy Search for Transmit Power Level Optimization ........... 35
4.3 Neural Network Training Procedure ...................................... 37
  4.3.1 Effect of Training Data Set Size ......................................... 41
4.4 Choice of Regularization Method .......................................... 42
  4.4.1 Increasing Training Speed Using Adaptive Optimization ...... 47
List of Tables

Table 1: Simulation assumptions for Figure 3.3. User assignment to base stations can be either random or to the base station with maximal channel norm ........................................ 33
List of Figures

Figure 1-1: Densification of cellular systems...............................................................................1

Figure 2-1: Layered structure of neural networks.................................................................11

Figure 2-2: A sigmoidal activation function ............................................................................12

Figure 2-3: A ReLU activation function $f(x) = \max(x, 0)$.....................................................13

Figure 2-4: A fully connected neural network with one input layer (yellow), one hidden layer (blue), and one output layer (orange). ..................................................................................13

Figure 2-5: Residual mapping between X and output of a fully connected layer. ..............22

Figure 2-6: Structure of the proposed convolutional layer..................................................23

Figure 2-7: The system considered in [12].............................................................................24

Figure 2-8: Structure of the neural network based power control scheme presented in [12].....26

Figure 3-1: The considered MIMO high density cellular system.........................................28

Figure 3-2: Base stations and user terminals randomly scattered in a $30m \times 30m$ area. The blue dots represent the base stations and the red dots represent the users.................................................................29

Figure 3-3: Average capacity for greedy search vs the number of power levels for a simulation environment stated in table 3.1 ..........................................................................................33

Figure 4-1: Histogram of the norm of channel matrix values. .............................................35
Figure 4-2: Validation accuracy vs Training set size (3000 vs 1000 training samples). ............42

Figure 4-3: Validation loss vs Training set size (3000 vs 1000 training samples). ...............42

Figure 4-4: Training and validation accuracies for the SGD optimizer.................................43

Figure 4-5: Training and validation losses for the SGD optimizer........................................43

Figure 4-6: Training and validation accuracy with and without LII regularization.................44

Figure 4-7: Training and validation losses with LII regularization........................................44

Figure 4-8: Training and validation accuracies with dropout, LII, LII and dropout, and with no
regularizer...............................................................................................................................45

Figure 4-9: Train and validation loss for dropout and LII as a regularizer ............................45

Figure 4-10: Validation accuracy, dropout only and dropout and LII regularized....................46

Figure 4-11: Validation losses-dropout regularized.................................................................46

Figure 4-12: Validation accuracies for Adam vs SGD. ........................................................47

Figure 4-13: Training and validation loss for Adam optimizer (with dropout).......................48

Figure 4-14: Validation accuracy with different number of hidden units............................49

Figure 4-15: Training and validation loss with different number of hidden units...............49

Figure 4-16: Training and validation accuracies for different Adam learning rates..............50
Figure 4-17: Validation losses for different Adam learning rates. ..............................................50

Figure 5-1: Structure of the neural network proposed for power level assignment......................52

Figure 5-2: Training and validation accuracies, two predictions with shared representation. .....53

Figure 5-3: Training and validation losses, two predictions with shared representation.............53

Figure 5-4: Prediction accuracy for the convolution.................................................................55

Figure 5-5: Learning rate for the convolution. ...........................................................................56

Figure 5-6: One hidden layer performs better with shared representations. .........................57

Figure 5-7: Validation accuracy for residual mappings............................................................57

Figure 5-8: Number of hidden units needed for shared representation.....................................58

Figure 5-9: Validation accuracy, ReLU vs Sigmoid in the second layer....................................59

Figure 5-10: Effect of mini-batch size on performance.............................................................60

Figure 5-11: CDF of system capacity with varying number of power levels. .........................62

Figure 5-12: Histogram of number optimal power levels for a base station with greedy search. (index 0 corresponds to power level -10 dBm and index 4 to +10 dBm). .................................62

Figure 5-13: CDF of average and minimum rate for greedy search (maximizing average rate) and maximum power. Greedy search with average rate as objective deteriorates the minimum rate when compared to maximum power transmission.................................................................65
Figure 5-14: Training and validation losses, maximizing the minimum rate (90000 training samples).....................................................................................................................................................................................66

Figure 5-15: Training and validation accuracies, maximizing minimum rate (90000 samples)...66

Figure 5-16: Overall system performance, maximizing the minimum rate. .................................67

Figure 5-17: Performance of the overall system, proposed structure vs the baseline (single hidden layer with sigmoidal activation). ..............................................................................................................................................69

Figure 5-18: Validation accuracy for five users, using shared weights vs only using fully connected layers..............................................................................................................................................70

Figure 5-19: Training and validation losses for the convolutional structure.................................70

Figure 5-20: Overall performance of convolutional structure, five users and five base stations. 71

Figure 5-21: training and validation accuracy for round-robin, five users and five base stations. 72

Figure 5-22: Overall performance for round-robin, five users and five base stations. .................72

Figure 5-23: Training and validation accuracy, one neural network per base station, reducing the number of fully connected neurons..................................................................................................................73

Figure 5-24: Training and validation accuracies, multiple ConvNets (18000 training samples). 74

Figure 5-25: Training and validation losses, multiple ConvNets (18000 training samples).........74
Figure 5-26: Performance of overall system, Multiple ConvNets vs Single ConvNet, multiple smaller ConvNets require less computations for training and forward pass and less training samples. ........................................................................................................................................75

Figure 5-27: Training and validation accuracies, one ConvNet per prediction, (9000 training samples). .......................................................................................................................................76

Figure 5-28: Training and validation losses, one ConvNet per prediction, 9000 training samples. ........................................................................................................................................76

Figure 5-29: Performance of overall system for round-robin, one ConvNet per prediction, (9000 training samples vs 18000 training samples). .............................................................................77

Figure 5-30: Training and validation loss for different convolutional layer depths. ................78

Figure 5-31: Training and validation accuracies for different convolutional layer depths. ....78

Figure 5-32: Prediction accuracy for dropout rate of 0.1 vs no dropout..............................79

Figure 5-33: Validation loss for dropout rate of 0.1 vs no dropout .....................................80

Figure 5-34: Validation loss for dropout rate of 0.1 vs no dropout .....................................80

Figure 5-35: Training and validation losses, random base station assignment, five users and five base stations, 9000 training set size. ..................................................................................................................81

Figure 5-36: Training and validation accuracies, random base station assignment, five users and five base stations. ..................................................................................................................82
Figure 5-37: Overall system performance, random base station assignment, five users and five base stations. ..............................................................82

Figure 5-38: Training and validation losses for maximizing the minimum rate, random BS assignment, four users and four base stations (1000 samples training set). .................................83

Figure 5-39: Training and validation accuracies for maximizing the minimum rate, random BS assignment (1000 samples training set size). .................................................................83

Figure 5-40: Overall performance for maximizing the minimum rate, random BS assignment (1000 samples training set). .................................................................84
List of Abbreviations and Symbols

BS: Base Station ..................................................................................................................................16

UT: User Terminal .............................................................................................................................16

MIMO: Multiple Input Multiple Output ............................................................................................16

TDD: Time Division Duplex ..............................................................................................................17

QoS: Quality of Service ..................................................................................................................19

IC: Interference Cancellation .........................................................................................................20

IIM: Inter-cell Interference Management .......................................................................................20

SINR: Signal to Interference Plus Noise Ratio ................................................................................20

MUD: Multi User Detection ............................................................................................................21

PIC: Parallel Interference Cancellation ..........................................................................................21

SIC: Successive Interference Cancellation .....................................................................................21

RLS: recursive least square ............................................................................................................22

CRAN: Cloud Radio Access Network ............................................................................................23

NN: Neural Network .......................................................................................................................23

SGD: Stochastic Gradient Descent .................................................................................................25
L: number of layers excluding the input layer ................................................................. 26

$N_l$: number of nodes in layer l of the neural network ......................................................... 26

$u_{lj}$: output of the j-th node of layer l ............................................................................... 26

$g_{l+1,k}$: sum of all weighted outputs at layer l connected to node k of layer $l+1$ .................. 27

$W_{k,l,q}$: connection node between node q of layer l to node k of layer $l+1$ ..................... 27

ReLU: Rectified Linear Unit .................................................................................................. 27

L1 norm: sum of absolute values of all weights of the neural network............................... 31

L2 norm: sum of squared value of all weights in the network .............................................. 31

$\rho_{l,a}$: error at the a-th node at the output .......................................................................... 32

$\rho_{l+1,a}$: error gradient at the output layer $l+1$ ................................................................. 33

$\lambda$: L2 weight penalty ................................................................................................... 33

Adam: Adaptive Momentum optimization technique ......................................................... 35

$m_t$: average of the past gradients .................................................................................... 35

$1 - \beta_1$: weight decay parameter for past gradients ....................................................... 35

$1 - \beta_2$: weight decay parameter for past squared gradients ......................................... 35
\(v_t\): past squared gradients in Adam

\(|X|\): number of possible power levels

\(K_{UT}\): number of users

OFDMA: Orthogonal Frequency Division Multiple Access

\(P_k\): power of the \(k\)-th base station

\(d_{ik}\): distance between user \(i\) and base station \(k\)

\(H\): channel matrix between all base stations and a user

\(n\): additive noise vector to the received signal

\(N_R\): number of antennas at the receive side

\(y\): set of all receive messages by a user

\(\eta_{ik}\): shadow fading parameter between user \(i\) and base station \(k\)

\(\Lambda_{ik}\): path loss between user \(i\) and base station \(k\)

\(G_{ik}\): effective channel between user \(i\) and base station \(k\)

\(C_{\text{MMSE}_i}\): MMSE capacity of user \(i\)
1 Introduction

The wireless industry has continued to witness increased data traffic demand over the years along with an astonishing increase in mobile data traffic [1], [2]. Subsequently, increasing the capacity of wireless systems has been an important goal for wireless system designers and operators. Traditionally, wireless cellular communication systems divide the service area into cells, and users within the coverage area of any cell communicate with a Base Station (BS) placed at the cell center. Reducing the size of cells and placing base stations close to User Terminal (UT), i.e. increasing the density of cellular communication systems as illustrated in Figure 1-1, has been a popular approach in responding to the capacity problem in cellular communication systems. Nevertheless, substantially higher deployment costs are typically associated with high-density cellular systems, particularly the added infrastructure and required links interconnecting base stations [3].

![Figure 1-1: Densification of cellular systems](image)

Using advanced antenna systems has been another approach in response to the capacity problem. This approach focuses on increasing the capacity, rather than the density, of individual cells to increase the overall capacity of a cellular system. Such an approach is most commonly implemented in massive Multiple Input Multiple Output (MIMO) [4][5] systems where base stations with large number of antenna elements connect user terminals utilizing a smaller number of antenna elements due to size and power constraints. For example, the 3rd Generation Partnership
Project (3GPP) Long Term Evolution (LTE) supports the use of base stations with 8 antenna elements in Release 10 [6], and more recently within the context of Release 13 [7] using 64 elements at the base station and up to 8 elements at the user terminal. One limiting factor in the deployment of large scale MIMO systems is the highly demanding computational complexity in addition to the added cost, particularly at the user terminal side. Moreover, several factors, such as rapid movement of system users along with varying propagation and multipath environments, limit the performance of the communication link between the user terminal and the base station.

In practice, optimal base station placement is rarely feasible in cellular systems, particularly as the system cell density increases. Subsequently, placing base stations in dense configurations close to user terminal within the service area inevitably causes the coverage area of system cells to overlap, thus introducing intercell interference, which can, if not properly addressed, severely constrain the system performance and limit the gains from cell densification [8]. Within the context of this thesis, we primarily focus on interference caused by neighboring transmitting base stations affecting the downlink of a base station in a high-density cellular communication system. We focus minimizing this interference by determining optimal power levels for each base station to optimize the overall system performance.

We consider the case of a Time Division Duplex (TDD) system. Base stations acquire the channel state information of user terminals within their coverage area, and communicate this information such that the channel information from other base stations is known. Such a system can be realized using coordinating small cells [9] or a cloud radio access network [3], [10]. With this information an algorithm determines and communicates the optimal power level for each base station and base stations adjust transmission power levels accordingly.

Transmission power in base stations is selected from a discrete set of power levels. When increasing the number of base stations, the number of possible system power level configurations grows exponentially, making it computationally expensive to perform a greedy search over all possible power level combinations to find optimal power levels for all base stations. The feasibility
of greedy searches is further reduced as the channel variation rate increases since power level configurations need to be adjusted accordingly. Moreover, having a large number of antenna elements scales the computational complexity by increasing the channel matrix size and channel state information communication overhead.

An alternative to greedy searching is introduced in the form of machine learning in a neural network, in which an algorithm is designed to perform mappings of features for large amounts of data. Rather than performing a greedy search for every computation of optimal power levels, a set of optimal power levels for a set of previously generated or recorded channel values is determined and used as a training set for the machine learning algorithm. The machine learning algorithm then determines relevant channel features along with the corresponding optimal mapping between channel state information and power levels.

Representation learning is a set of designed methods allowing machines fed with data to learn representations which can be used for classification purposes in an automated manner. Deep learning is a layered version of representation learning, in which a slightly more abstract representation of a data set is produced using a layered non-linear data mapping to allow efficient abstraction of large and highly complex data sets with the use of a sufficient number of layers. For a classification task, higher levels of abstraction amplify features that are important for the classification task and repress irrelevant variations. Such an approach has been highly successful for a wide range of applications spanning multiple domains such image and speech recognition, drug discovery and genomics [11].

Finding the set of optimal power levels corresponding a set of values for channel state matrices in a cell is a classification problem similar in nature to image classification. Within the context of this thesis we propose a method to find the optimal power levels in a high-density cellular communication system using deep learning. We follow the work of [12] in this area, which first proposed the application of deep learning for this problem.
1.1 Contributions

We focus on the downlink of a MIMO high-density cellular network. We provide a power control scheme for this system based on neural networks. Following the work introduced by [12], this work proposes an improved learning algorithm and neural network structure which improves the training time, scalability and the computational complexity of the neural network by an order of magnitude. The main contributions of this thesis are as follows:

- Following the work of [12] a neural network framework for computing the optimal downlink power levels for a TDD MIMO high-density cellular communication system is presented. Instead of learning a separate network for each of the base stations in the system as presented in [12], the proposed system learns an abstract representation of the norm of the channel matrices, and then using that representation to predict the optimal power level for each base station. This essentially cuts the computational complexity of the predictor by factor of the total number of base stations.

- Dropout and adaptive momentum optimization technique (Adam were employed to prevent overfitting and cut the training time of the network, thus further reducing the computation time by a factor of four.

- Using shared weights (convolution) was studied and shown to significantly improve prediction accuracy.

- Binary Power levels were also studied for reduced computational complexity when utilizing larger training sets.

- Improving minimum user rate to provide guaranteed Quality of Service (QoS) was studied and performance comparisons were made relative to the case where the neural network was trained to only maximize the sum capacity.
The assumption of one user per base station in [12] was removed and the system performance was observed under round robin with increase in number of antenna elements at the transmitter side.

1.2 Organization

The thesis is organized as follows: Chapter two provides a basic overview of neural networks, along with the notation being used for the remainder of this thesis, to ensure readers can easily follow the presented discussion. Chapter three provides the system model and simulation framework for the high density cellular system presented in this work.

Chapter four presents a neural network based power control framework for the downlink of a high-density cellular TDD MIMO system, following the neural network scheme presented in [12]. The activation function, regularization, effect of training set size, and adaptive optimization to increase accuracy and reduce the training time while eliminating the need to pretrain and pre-scale the network as in [12].

Chapter five considers a single neural network instead of multiple neural networks for predicting the optimal power levels. The use of shared weights, residual connections, and varying the number of hidden units are also introduced to provide improved accuracy and significantly reduced computational complexity. Additionally, minibatch size is increased to reduce the training computational complexity and the number of overall back propagations. Having more than one user per base station (round robin) is also discussed along with another base station assignment policy, in which each user is connected to one base station under random assignment. This could happen when the variability in the system is too high for users to change their assigned base station. Furthermore, the fairness of the proposed power control scheme is discussed, and a power control scheme maximizing the minimum rate user terminal in the system is proposed. Finally, the overall system performance and scalability is discussed, and conclusions are provided in Chapter 6.
2 Literature Review

High-density cellular communication systems are typically characterized by overlapping cell coverage causing high inter-cell interference, which deteriorates the Signal to interference plus noise ratio (SINR) and degrades the overall system performance. To mitigate inter-cell interference, several Inter-cell Interference Management (IIM) techniques have been proposed. The proposed techniques generally fall into four main categories [14]: (i) inter-cell interference cancellation (IC), (ii) inter-cell interference avoidance, (iii) distributed interference management and (iv) power control for inter-cell interference management.

2.1 Main Inter-Cell Interference Management Techniques

In this section we present some of the main strategies used for mitigating the intercell interference. In section 2.1.1 and 2.1.2 we discuss intercell interference coordination and management, and in section 2.1.3 we discuss distributed intercell interference management which assumes no backhaul information is available. Section 2.1.4 discusses power control as a means of intercell interference management which is the main focus of this work.

2.1.1 Interference Cancellation

Among the most commonly proposed methods for Interference Cancellation (IC) [15] is Successive Interference Cancellation (SIC), where the interfering signals are detected and subtracted from the received signal successively based on their power. On the other hand, Parallel
Interference Cancellation (PIC), where all the signals are decoded simultaneously and detection is repeated several times to counter poor signal detection reliability in initial attempts. Another type of IC is turbo based multi user detection (MUD) which is reported to have shown promising results [16]. Another type of IC is turbo based multi user detection (MUD) which is reported to have shown promising results [16] where a maximum a posteriori based turbo receiver iteratively maximizes the log likelihood of the transmitted signal $X$ given the received signal $Y$, expressed in matrix form, where the rows of $X$ denote the transmitted symbols arriving from different sources, and rows of $Y$ denote corresponding received symbols. The authors of [16] proposed several modifications, including a Recursive Least Square (RLS) algorithm instead of MMSE for channel estimation, and joint estimation of channel and the received signal, based on message passing algorithm, and show considerable performance gain in terms of error rate and channel tracking ability.

2.1.2 Inter-Cell Interference Coordination

Interference avoidance or inter-cell interference coordination (ICIC) reduces interference by modifying the transmitted signals [17]. Various methods have been discussed, such as power control, resource scheduling and frequency reuse [18]. Both IC and ICIC are discussed in [19] as separate techniques. In [20], minimizing the total transmitted power, while meeting the minimum SINR requirement in a MIMO system, is proposed to reduce inter-cell interference.

2.1.3 Distributed Interference Management

Distributed interference management attempts to minimize inter-cell interference in a cooperative manner when backhaul links are constrained and information sharing between user terminals is limited. In [21], a distributed algorithm is proposed to progressively reduce SINR targets of strongest interferers until the system SINR target is met.
2.1.4 Power Control for Inter-Cell Interference Management

Power control is instrumental for the operation of cellular systems [22]. In [23] a general wireless scheduling and interference coordination problem is formulated as an optimization problem with linear mixing utilities and cast in belief propagation framework, with the objective of calculating marginal distributions of a joint probability function. The method proposed in [23] requires transmission of messages between base stations and user terminals during the optimization process over the air. Subsequently, the traffic overhead increases under rapidly varying channel conditions, thus limiting the feasibility of such methods. In [12] the authors propose an inter-cell interference mitigation scheme that controls the base station transmission power using deep learning feed-forward neural networks (NNs) in combination with IC at user terminals to further enhance the system capacity. The problem of a high-density cellular system with one user per base station is considered, and a neural network based solution is proposed. Specifically, the norm of the channel matrices of each base station for each user are fed to a set of neural networks, such that one independent neural network is employed per base station, with each neural network predicting the optimal transmit power level for the input base station. Optimal power level computation is performed in a Cloud Radio Access Network (C-RAN) and are communicated back to base station. The neural network architecture proposed in [12] determines optimal power levels using the norm of the channel matrix only, and thus substantially reduces the amount of overhead communication in the system. However, utilizing an independent neural network for each base station is computationally inefficient and a more efficient architecture is proposed in this thesis after providing an overview of neural networks.

2.2 Neural Networks Overview

In this section we provide an overview of the neural networks. Section 2.2.1 describes an overview of supervised learning. Section 2.2.2 describes stochastic gradient descent as a means for optimizing a model. In 2.2.3, non-linear activations and deep architectures for machine learning are introduced. Section 2.2.4 summarizes neural networks, with focus on supervised learning,
including the cross-entropy as a loss function and the backpropagation algorithm. Section 2.2.5 describes the batch gradient descent, and sections 2.2.6 discusses validation and test set splits. Sections 2.2.7 and 2.2.8 present the overfitting phenomena, and ways to counter it such as early stopping and overfitting. Section 2.2.9 discusses methods to speed up neural network training, such as momentum technique and Adam, and 2.2.10 discusses residual mapping, allowing to go deeper while avoiding the unitary mappings. 2.2.11 presents shared weights and convolutions, and presents a specific convolution layer used in this work which significantly enhances the performance of the system. Section 2.3 presents a related work in application of neural network for power level control, which is considered as a baseline for this work.

### 2.2.1 Supervised Learning

Supervised learning is the most common form of machine learning and is best illustrated in the form of a classification problem. For example, consider an algorithm designed for the purpose of classifying an image into one of four categories, such as people, cars, animals, and houses. Such an algorithm can be implemented by providing an initial set of labeled images containing pictures resembling each category to initiate a training phase, in which a machine produces a vector of values characterizing each category. The algorithm design target is to match the output vector values with the relevant category for each input image. Achieving this goal requires the adaptation of the algorithm objective function based on a loss function representing the distance between the desired vector values and the machine output values. The machine then reduces classification errors by adjusting the parameters of the objective function based on the loss function output. These adjustable parameters are denoted weights, typically grouped in a vector, and determine the input-output relationship of the machine [24].

### 2.2.2 Stochastic Gradient Descent

To adjust the weight vector, a gradient vector representing the classification error increase or decrease, given any difference in the weight vector, is computed. The weights are then adjusted based on the gradient vector to minimize the loss function. The average loss function over all
training examples can be regarded as a hilly landscape of which the negative gradient determines the steepest descent towards a minimum with lower average error. In practice, when the average error is roughly the same throughout the training data set, the gradient can be estimated using a subset of the training data and verified by a test data set similar to the training set [24]. Training is thus repeated over multiple training data batches until the average training loss, referred to as the Stochastic Gradient Descent (SGD), stops decreasing. The batch gradient gives a noisy estimate of the gradient over the training and hence, the stochastic naming. When compared to more complicated optimization techniques, such as gradient descent, second order stochastic gradient descent, or second order stochastic gradient descent, SGD converges to high performing set of weights at a considerably faster rate [24], [25].

2.2.3 Non-Linear Activity Functions and Deep Architectures

Many practical machine learning applications utilize linear classifiers. A linear classifier computes a weighted sum of a featured vector. If above a specified threshold, the vector is categorized as belonging to a class. Linear classifiers divide the input space into regions and are sensitive to input vector variations such as shifts and rotations. An alternative to linear classifiers is using generic non-linear features as in the kernel methods. Kernel methods are a family of methods utilizing the Kernel trick to operate in a higher dimensional feature space without explicitly calculating the space coordinates to reduce computations. The most common kernel method is the support vector machines - discriminative classifiers that, given a labeled input data, output an optimal hyperplane that categorizes new examples. Nevertheless, kernel methods are believed to be limited in their ability to generalize beyond training data. In practice, it is conventional to use feature extractors, such as edge, corner, blob, and ridge detectors for computer vision, Mel-Frequency Cepstrum, Delta-Spectral Cepstrum coefficients or pitch based features among other methods for automatic speech recognition. Such a limitation could be avoided if the target features can be learned automatically, which is the key advantage of deep learning [24]. A deep learning architecture is a multilayer stack of linear activities and non-linear outputs, all of which are subject to learning with each non-linearity transforming the input to increase the invariance and selectivity of the
representation. With deep learning, it is possible to learn extremely complex features with high sensitivity to small details and invariance to irrelevant features. For example, in case of images, pose, background and surrounding objects [24].

### 2.2.4 Neural Networks

In this section, neural networks are introduced following the notation of [12]. A neural network consists of an input layer, an output layer and a set of concealed layers consisting of at least one layer. Additionally, as shown in Figure 2-1, each layer consists of a set of neurons:

![Layered structure of neural networks](image)

**Figure 2-1:** Layered structure of neural networks.

$u_{l,j}$ represents the output of the $j$-th node of the $l$-th layer, where $0 \leq j \leq N_l$, and $0 \leq l \leq L$, with $N_l$ being the number of nodes in the $l$-th layer excluding bias, and $L$ denoting the number of layers excluding the input layer.
The output of the bias node at layer $l$, $u_{l,0}$ is equal to 1. $W_{k,l,q}$ denotes the connection weight between node $q$ of layer $l$ to node $k$ of layer $l+1$. $g_{l+1,k}$ represents the sum of all weighed outputs at layer $l$ connected to node $k$ of layer $l+1$. Therefore, we have:

$$g_{l+1,k} = \sum_{q=1}^{N_l} W_{k,l,q} u_{l,q}$$

(2.1)

The output of the layer is obtained by passing the results of the summations described above through a non-linear activity function $f(x)$. Examples of $f(x)$ include the sigmoidal function, the Rectified Linear Unit (ReLU) function and the hyperbolic tangent. The sigmoid non-linear activity function is defined as $f(x) = \frac{1}{1+e^{-x}}$ and is shown in Figure 2-2:

![Sigmoid Activation Function](image)

**Figure 2-2**: A sigmoidal activation function

The ReLU function is defined as $f(x) = \max(x, 0)$ and shown in Figure 2-3:
Figure 2-3: A ReLU activation function $f(x) = \max(x, 0)$.

In a fully connected layer $l+1$, neuron nodes in layer $l$ are connected through weights $w_{k,l,q}$ to all neurons in $l$. Figure 2-4 depicts the structure of a generic neural network comprising an input layer, a set of hidden layers and an output layer.

Figure 2-4: A fully connected neural network with one input layer (yellow), one hidden layer (blue), and one output layer (orange).
In the classification scenario, the number of outputs would be the number of possible classes and in the case of image classification the input layer comprises a set of pixel data for an image in vector form. Nevertheless, the inputs and outputs of a neural network can vary based on the considered problem. Typically, in a supervised classification problem, where an input (such as an image) belongs to one of a possible set of classes, the desired output of the neural network would be the estimated probabilities that the input belongs to each of the considered classes. This entails a positive output of the neural network for each class, with the sum of all outputs being equal to one. In such cases, it is conventional to use a sigmoid function when two classes are considered, with probabilities greater than 0.5 being interpreted as a prediction of belonging to one class and less that entailing belonging to the other class. On the other hand, a softmax function is used at the output when more than two classes are considered.

\[ u_{L,a} = \frac{e^{g_L,a}}{\sum_{\tilde{d}} e^{g_L,\tilde{a}}} \]  

(2.2)

For all classes \(0 \leq a \leq C\) this will provide values that are between 0 and 1, and a sum of all values equal to 1, and therefore meets the viability criteria. Weights and biases at each layer are initialized randomly, and are then modified in a step-by-step process through an algorithm referred to as backpropagation. The backpropagation algorithm aims to minimize a loss function, one that represents the difference between the probability distribution at the output of the neural network and the input labels.

### 2.2.4.1 Loss Function and Backpropagation Algorithm

The backpropagation algorithm is an application of the chain rule of derivatives. Once the derivative of errors is known at the output, and given that at each layer the output of a neuron is a
differentiable function of the input, weights at each layer could be updated starting from the output layer to the input.

A classification problem is considered once again. The desired output of the neural network is a vector $v$, the size of which is the number of classes. The desired output of the $a^{th}$ node in $v$, or $v_a$ is determined as:

$$v_a = \begin{cases} 1 & \text{if } \hat{c} = c_a \\ 0 & \text{if } \hat{c} \neq c_a \end{cases}$$

(2.3)

Such a notation is referred to as a one-hot notation, meaning that in case the image being classified belongs to class $\hat{a}$ then the $a^{th}$ element of the desired output vector $v$ is 1, and all other elements of the vector $v$ are zero. Subsequently, the loss function would quantify the similarity, or lack of, between inputs and outputs. At the output of the neural network we have a softmax, the output of which is regarded as the predicted probability of belonging to each class. The softmax output would ideally match the desired output, i.e. we want to maximize the prediction probability of belonging to the correct label. The logarithm of this predicted probability can be maximized to produce better gradients for training than the actual probabilities with a refined loss function defined as:

$$J = -\sum_{\hat{a}} v_{\hat{a}} \log u_{L,\hat{a}}$$

(2.4)

Preventing the weights produced by (2.4) from having large values entails penalizing larger weights. The most common method typically employed to achieve this target is the LII norm, which sums the squared values of all weights, followed by the LI norm which is the sum of absolute
values of the weights. Compared to LII norm, LI norm produces larger weights with more zeros.
The overall loss function for the LII norm would be:

\[
J = -\sum_{a} v_{\bar{a}} \log u_{L,\bar{a}} + \frac{\lambda}{2} \sum_{l=0}^{L-1} \sum_{a=1}^{N_l} \sum_{b=1}^{N_{l+1}} w_{a,l,b}^2
\]

with \( \lambda \) referred to as the weight decay parameter. The backpropagation algorithm aims to optimize weights \( w \) in a step by step fashion such that:

\[
w_{a,l,b}(\xi + 1) = w_{a,l,b}(\xi) - \Delta w_{a,l,b}
\]

And:

\[
\Delta w_{a,l,b} = \mu \frac{\partial J}{\partial w_{a,l,b}}
\]

\( \mu \) denotes the learning rate and must be carefully set to prevent oscillations in the optimization process and ensure fast conversion. The backpropagation algorithm would then be:

**Algorithm 1: Backpropagation**

Require: \( \mu \), the learning rate
Require: Stochastic objective function $J(W)$ with parameters $W$ (the matrix of the weights of the neural network)

Require: $W_0$ the initial weight matrix

**While** $W$ not converged **do**:

Feed the neural network with values $u_0$ and compute the output $u_L$

Compute the $\rho_{L,a}$ corresponding to the error at the a-th node at the output as:

$$
\rho_{L,a} = \frac{\partial J}{\partial g_{L,a}} = \frac{\partial \left( -\sum_{a}^{\infty} v_{a} \log u_{L,a} \right)}{\partial g_{L,a}} = \sum_{\tilde{a}}^{\infty} \frac{v_{\tilde{a}}}{u_{L,\tilde{a}}} \frac{\partial u_{L,\tilde{a}}}{\partial g_{L,a}}
$$

$$
= -\frac{v_{a}}{u_{L,a}} u_{L,a} (1 - u_{L,a}) - \sum_{\tilde{a} \neq a} v_{\tilde{a}} (-u_{L,a} u_{L,\tilde{a}})
$$

$$
= -v_{a} (1 - u_{L,a}) + \sum_{\tilde{a} \neq a} v_{\tilde{a}} u_{L,a} = -(v_{a} - u_{L,a})
$$

Compute $\rho_{L,b} = \frac{\partial J}{\partial g_{l,b}}$ in descending order of $l$ for the b-th node in the l-th layer.

$$
\rho_{L,b} = \sum_{a=1}^{N_{l+1}} \rho_{L+1,a} \frac{\partial g_{l+1,a}}{\partial g_{l,b}} = \left( \sum_{a=1}^{N_{l+1}} \rho_{l+1,a} w_{a,l,b} \right) f'(g_{l,b})
$$

Compute the partial derivative with respect to the weight:

$$
\frac{\partial J}{\partial w_{a,l,b}} = \rho_{l+1,a} u_{l,b} + \lambda w_{a,l,b}
$$

Plug into 2.5 and 2.6 and update the weight values

**End While**

**Return** $W$
2.2.5 Batch Gradient Descent

Estimating the gradient based on only one sample at a time results in noisy gradient updates by SGD. On the other hand, Gradient Descent computes the average gradient for all training samples to provide more accurate gradients at the expense of increased computations. Batch gradient descent balances the trade-off between the two approaches by computing the average error among mini-batches of input samples before back-propagating the error, thus providing better gradients at a more reasonable computational cost. The model is trained by dividing the training dataset into mini-batches with a small number of training samples. SGD is then performed on each mini-batch and weights are adjusted accordingly before proceeding to the next mini-batch of the data. Once all mini-batches, iterating through all training set samples, have been processed an epoch is completed and the process is repeated starting from the first mini-batch. Specifically, an epoch is defined as a single pass through the training set, followed by a pass of the test and validation sets.

2.2.6 Training, Validation, and Test Procedure

Measuring the ability of trained machines to generalize from features learned in the training phase requires the use of an independent test set not been used for training and, subsequently, a portion of the considered data is typically set aside for testing. Upon the completion of the training phase, the performance of the trained model is verified on the test data set. Additionally, after the training is done and before testing a model on the test data set, some minor adjustments, such as the number of training steps needed to be taken, can be made. A subset of the training set is typically set aside for making these small adjustments, which is referred to as the validation set. Hence, data sets are divided into training, validation, and testing dataset.

2.2.7 Overfitting and Early Stopping

Overfitting occurs when a model under training is either too complicated, i.e. has too many hidden layers or hidden units per layer, or is trying to estimate a noisy dataset with a linear relationship between inputs and outputs using a high order polynomial. Such an approach may work well for
minimizing a performance metric such as the mean squared error during training, but fails when compared to a linear model on a validation data set that has not used for training. Overfitting also occurs in cases where a model is trained extensively, for example when over-iterating over the training data set, which causes the trained model to start to learn the noise in the data, and thus lose its ability to generalize over other data sets, resulting in a poor performance in the validation phase. Overfitting can be identified by observing the validation accuracy and loss. In such cases the training accuracy and loss continue to improve but the validation accuracy and loss deteriorate. Preventing overfitting can be achieved by terminating model training as soon as overfitting is detected and is referred to as early stopping.

2.2.8 Dropout

One way to overcome overfitting is to combine the predictions of different neural networks at the testing phase. However, computation constraints make such an approach difficult to implement in large neural networks. Dropout addresses overfitting by removing some units during the training from the network in a randomized fashion to prevent the network from co-adapting to undesired features. With dropout training is applied to thinned networks, the effect of which in the weights is removed by scaling the weights down to a smaller value, while keeping all units in the test phase [26]. As discussed in the previous section, other methods can be employed to prevent overfitting such as LI and LII regularization and stopping the training before the validation error starts to increase. However, the rationale for dropout is that the best way to regularize a fixed size model is to average the predictions of all possible sets of parameters and weighing each setting by its posterior probability given the input data [26]. While this is possible with considerable computational resources in small models, [26] proposed an implementation using an equally weighted geometric mean of an exponential number of models that share the parameters (weights of the neural network) to substantially lower computation requirements. Dropout thus allows a reduced training dataset size while increasing the training time.
2.2.9 Momentum Technique and Adam

For this research, a modified version of stochastic gradient descent proposed in [25] is employed. In this method, previous weight updates at the previous steps are also considered along with the gradient of loss w.r.t. weights used for weight update. Such an approach, essentially stochastic gradient descent with momentum, provides a smoother convergence towards the local minima of the multidimensional space that is divided by an unbiased moving average variance of the estimate of the second moment to provide a much faster convergence even without pre-training. Similar to stochastic gradient descent, it can be applied not only to optimizing the loss in neural networks, but other gradient-based optimization problems as well.

The adaptive momentum optimization technique (Adam) keeps an exponentially decaying average of past gradients, \( m_{t-1} \), and past squared gradients, \( v_{t-1} \), defined as:

\[
m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t
\]

\[
v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2
\]

The authors in [25] observed that these terms are biased towards zero, particularly in the initial time steps and when the decay rates \((1 - \beta_1)\) and \((1 - \beta_2)\) are small. Therefore, unbiased estimate of \( m_t \) and \( v_t \) are computed as follows:

\[
\hat{m}_t = \frac{m_t}{1 - \beta_{1t}}
\]

\[
\hat{v}_t = \frac{v_t}{1 - \beta_{2t}}
\]
The update rule then would be:

\[ w_t = w_t - \frac{\mu}{\sqrt{\beta_t} + \epsilon} \hat{m}_t \]

(2.12)

In [25], \( \beta_1 = 0.9 \), \( \beta_2 = 0.999 \) and \( \epsilon = 10^{-8} \) are proposed as good default values and Adam is empirically shown to compare favorably to other adaptive learning techniques. In addition, \( \mu = 0.001 \) was also proposed as a good value for machine learning problems.

### 2.2.10 Residual Connections

In some cases, increasing the number of hidden layers decreases the prediction accuracy of the network. In [32], it is observed that this phenomenon occurs in some cases, not only by overfitting, but also by higher layers learning an identity mapping of previous layers. To mitigate this issue, it was proposed to provide layers with the identity mapping of preceding layers by adding layer inputs to layer outputs, as shown Figure 2-5, to also mitigate the vanishing gradients problem.
2.2.11 Shared Weights

Using shared weights in a hidden layer instead of a fully connected layer reduces the number of parameters in the system which increases the training speed and reduces overfitting. A specific variant of this approach will be employed later in this thesis. Specifically, the input matrix of dimension $K_{UT} \times K_{BS}$, representing the norm of the channel between user terminals and base stations, is essentially the input to the network. In this context, shared weights indicate symmetry between users that maps to the columns of the channel norm matrix to be interchangeable. In other words, there exists a transformation in the first layer of the network on the columns of the matrix that is agnostic to the position of those columns. Such a transformation can be implemented with a single fully connected layer applied on each column separately as shown in Figure 2-6. In deep learning literature, this translates to a convolutional layer with filter size $K_{UT} \times 1$, with number of outputs, $n$, referred to as the depth.

**Figure 2-5:** Residual mapping between X and output of a fully connected layer.
2.3 Related Work

In [12], a power control scheme is proposed for the downlink of a TDD MIMO system, with $K$ base stations and $K$ user terminals, an equal number of antennas at the transmit and receive antennas $N$ and $\mathbf{G}_{i,k}$ representing the channel matrix between user $i$ and base station $k$ as shown in Figure 2-7.
Figure 2-7: The system considered in [12].

The considered metric for optimization is the average user rate. For this purpose, a neural network based power control scheme is proposed and compared to greedy search, maximum power transmission, the belief propagation algorithm and the distributed pricing algorithm. The number of power levels for which the rate per each user needs to be calculated is $|X|^K_{BS}$ where $|X|$ is the number of possible power levels per base station, assumed to be five in [12].

Under the greedy search, each calculation of the achievable throughput rate needs to be performed for each user, which involves a matrix inversion and a determinant, along with multiple complex matrix multiplications under the greedy search. Therefore, the computational complexity of the greedy search is $O(K_{UT}N^3|X|^K_{BS})$, which means it would be in the order of $10^6$ for $N = 4$, $K_{BS} = K_{UT} = K = 5$, and $|X| = 5$, thus rendering the greedy search infeasible for real time calculation. Moreover, the channel state information needs to be available for all the users from all base stations for calculating the optimal power levels, which entails an additional overhead to be transmitted over the backhaul. Nevertheless, the greedy search, while practically infeasible, is typically used to benchmark the performance of other algorithms as it provides an upper bond on system performance. On the other hand, the maximum power transmission requires no computations but results in the poorest performance due to high interference levels.
The belief propagation algorithm is a message passing algorithm involving the calculation of a marginal distribution of a joint probability distribution function. The probabilities of $P_1, \ldots, P_K$ are computed by iteratively exchanging messages between base stations and user terminals and the effectiveness of the algorithm depends on the number of iterations as detailed in [23]. While guaranteeing convergence to an approximate solution in a fully connected network, the drawback of the belief propagation algorithm is the high computational complexity, $O(N^3|X|^K_{BS})$ if the number of users and base stations is equal, which increases exponentially with the number of base stations. While the complexity could be tackled by Gaussian and linear approximations at the expense of reducing performance, requiring exchanging messages over the air results in a substantial overhead.

In the distributed pricing algorithm, proposed in [31], users announce their sensitivities to the current levels of interference, and power is adjusted to maximize user rate. The algorithm requires the channel state information from the interfering link and the interfered link along with a price exchange protocol; which severely limits its feasibility practical deployments due iterative computations leading to extended convergence time in spite of the lower computational complexity, $O(K_{UT}N^3|X|)$, when compared to the greedy search and belief propagation algorithms.

In [12], the authors provide a neural network based power control scheme, which, aside from the greedy search, outperforms all previous algorithms mentioned here while providing substantially lower computational complexity. In this scheme, the Frobenious norm of the estimated channel matrix from user $i$ to base station $k$, $\mathbf{G}_{i,k}$, estimated from the transmission of pilot signals based on Zadoff-Chu sequence is transmitted over the backhaul to a decision-making device in a cloud radio access network. The decision-making device computes the optimal power levels for each base station and instructs base stations to accordingly adjust power levels. The decision-making device consists of a set of neural networks, one neural network per base station, deciding the optimal power level for their respective base station based on a prediction made at the output of the network. Figure 2.8 details the scheme proposed by [12].
The neural network described in [12] is a restricted Boltzmann machine, consisting of three layers, the input layer, the hidden layer, and the output layer which is a softmax function. The input layer to the neural network is the scaled values of the Frobenious norm of the estimated channel. The scaling is put in place to ensure input values to the network are greater than zero and less than one. The activity function is a sigmoidal function, and the loss function is a softmax cross entropy. The training of the network is based on the optimal power level values obtained from the greedy search in a one hot notation, the loss function is cross entropy with LII regularization to prevent overfitting and the optimization is a stochastic gradient descent with mini-batch size of 50. The computational complexity of this method is with the order of $O(N_1(K^2 + |X|))$, where $N_1$ is the number of hidden units in the hidden layer, and is thus substantially lower than other methods. It was demonstrated that $N_1 = 200$ is a good value for optimal performance under the assumption that there are equal number of base stations and user terminal, each base station connecting a single user terminal.

**Figure 2-8:** Structure of the neural network based power control scheme presented in [12].
The method in [12] was also reported to outperform the maximum power transmission, distributed pricing and belief propagation in terms of sum capacity while eliminating both the overhead of message transmission over the air and the required iterations to converge as in belief propagation and distributed pricing. However, there are certain drawbacks to the structure proposed in [12]. The most significant of which is computational complexity and long time to train. Moreover, training needs to be performed $K_{BS}$ times, as each base station is controlled by a separate neural network. Apart from training, the employment of the average rate or the sum rate as the metric of performance does not provide any provisions for attaining a specific level of quality of service, and often leads to low QoS levels for users with unfavorable channel conditions. In the following chapters, we provide means to address these issues.

3 System Model, Simulation Assumptions and Related Work

In this section we discuss the assumptions made for simulating our small cell system. Section 3.1 discusses the model used for the small cell system and wireless channel, and the optimization metric and its relation to channel matrix values. Section 3.2 provides some detailed assumptions in modeling the environment, including the fading model, transmission type, etc.

3.1 System model

A Time Domain Duplexing Multiple Input Multiple Output high-density cellular system is considered and shown in Figures 3-1 and 3-2. In each simulation drop, users and base stations are scattered randomly in a $30m \times 30m$ environment with uniform distribution. Shadow fading, path loss, and channel matrix is randomly generated according to equations 3.3 and 3.4. The optimal power levels are then computed by a greedy search as discussed shortly, and then used as a sample for training the neural network. Then, once enough samples are generated, a neural network is trained to mimic the results of the greedy search by only having the norm of the channel between each user and base station. The CDF of system capacity is drawn in the to compare the performance
of greedy search, neural network based scheme, and the maximum power transmission. One could consider a scenario where base stations are fixed in some pre-specified random locations and the users are scattered in the environment. However, such scenario is not valid if the base stations are also mobile. Having the base stations makes the training data less diverse, potentially allowing the model to learn to predict the optimal power levels with higher accuracy. Therefore, training a model that works in the scenario considered here is potentially a more difficult problem. However, a model that performs well in this setting would likely perform as well or even on that scenario as well, and therefore our assumption provides a lower bound on performance for the case where base stations are fixed. Aside from that, drawing system capacity CDF for the case where base stations are fixed requires significantly more computations since per each drop of base station positions requires a separate training set corresponding to those combinations of locations for training the neural networks.

Figure 3-1: The considered MIMO high density cellular system.
Figure 3-2: Base stations and user terminals randomly scattered in a $30\text{m} \times 30\text{m}$ area. The blue dots represent the base stations and the red dots represent the users.

$K_{BS}$ base stations connect $K_{UT}$ user terminal, with each UT connecting to the BS providing the highest received signal power and signals received from other regarded as interference. The $k^{th}$ base station transmits an $N$-by-$1$ vector $\mathbf{s}_k$. The autocorrelation of $\mathbf{s}_k$ is assumed to be diagonal with $E(\text{tr}[\mathbf{s}_k \mathbf{s}_k^H]) = P_k$.

Relative movement between transmitters and receivers may cause certain components of the received signal to undergo frequency shift. However, in a high-density cellular system with low mobility, it can be assumed that this frequency shift is negligible. Various parameters are used to characterize wireless channels which structure received signals. Due to the broadcast nature of the wireless channel, the simple tap-delay channel model [30], which characterizes a channel by an impulse response can be utilized:

$$h(t) = \sum_l a_l \delta(t - \tau_l) \quad (3.1)$$

The Fourier transform of (3.1) would be the channel response in the frequency domain, which in general varies with frequency, and therefore such channels are referred to as frequency selective.
channels. With sufficiently small channel bandwidth, i.e. smaller than the coherence bandwidth of the channel, the channel representation in the frequency domain can be modeled by a constant factor. Contemporary wireless communication systems utilize Orthogonal Frequency Division Multiple Access (OFDMA), which divides a frequency selective channel into a set of channels with flat fading. With multiple antennas at both the transmitter and the receiver sides, a flat fading channel between each transmit and receive antenna pair would be defined under this setup. With Additive White Gaussian Noise (AWGN) at the receiver, the overall channel model can be characterized as:

\[ \mathbf{y} = \mathbf{Hx} + \mathbf{n} \]  

(3.2)

In which \( \mathbf{y} \) denotes the set of all received messages, \( \mathbf{H} \) denotes the channel matrix, \( \mathbf{x} \) denotes the set of all transmitted messages and \( \mathbf{n} \) denotes the noise, assumed under AWGN to be a Gaussian random vector with a covariance equal to \( \sigma_n^2 \mathbf{I}_{N_R} \), where \( N_R \) is the number of antennas at the receive side. To transmit the wireless signal over the air, the signal is modulated to a higher frequency. A band pass signal can be represented as a complex low pass signal. In a rich scattering environment, assuming no line of sight path between transmitter and the receiver, elements of \( \mathbf{H} \) can be assumed to be independent and identically distributed (i.i.d.) while following a complex normal distribution with zero mean and unit variance. For a single transmitter and receiver, this is the flat Rayleigh channel.

The received signal at the UE side suffers from path loss. The path loss \( \Lambda_{ik} \) between user terminal \( i \) and base station \( k \) is the product of distance path loss and log normal shadowing. Therefore:

\[ \Lambda_{ik} \propto d_{ik}^{\alpha} \eta_{ik} \]  

(3.3)

Where the operator \( \propto \) represents the proportional relationship between the left and right sides of above expression, \( d_{ik} \) is the distance from user terminal \( i \) to base station \( k \), \( \eta_{ik} \) is a random variable following a log-normal distribution, and \( \alpha \) is the path loss exponent. The elements of the
normalized channel matrix between user $i$ and base station $k$, $\mathbf{H}_{ik}$, under Rayleigh flat fading are complex normal random variables with zero mean and unit variance. Considering the effect of path loss and shadow fading, the effective channel $\mathbf{G}_{ik}$ between the user $i$ and base station $k$ would be:

$$\mathbf{G}_{ik} = \frac{1}{\sqrt{A_{ik}}} \mathbf{H}_{ik}$$

(3.4)

The received signal vector $\mathbf{y}_i$ can be expressed as:

$$\mathbf{y}_i = \mathbf{G}_{ii} \mathbf{s}_i + \sum_{i,i\neq k} \mathbf{G}_{ik} \mathbf{s}_k + \mathbf{n}_i$$

(3.5)

With $\mathbf{n}_i$ being the $N$-by-1 noise vector of the $i^{th}$ user terminal. Therefore, the MMSE capacity of user $i$, $C_{MMSE/i}$, would be:

$$C_{MMSE/i} = \log_2 \left( \det \left( \mathbf{I}_N + \frac{P_i}{NT} \bar{\mathbf{R}}_{NI,i} \mathbf{G}_{ii} \mathbf{G}_{ii}^H \right) \right)$$

(3.6)

Where:

$$\bar{\mathbf{R}}_{NI,i} = \sum_{k \neq i} \mathbf{G}_{ik} \mathbf{G}_{ik}^H + \sigma_n^2 \mathbf{I}_N$$

(3.7)

$\sigma_n^2$ is the average noise power and $\det(.)$ represents the determinant of the matrix. The achievable system capacity is then defined as:
\[ C = \frac{1}{K_{UT}} \sum_{k=1}^{K_{UT}} C_k \]  

(3.8)

And the objective is to find a power level set that maximizes the system capacity:

\[ \hat{P} = \arg \max_P C \]  

(3.9)

### 3.2 Simulation Assumptions

To account for high-density cellular deployments in non-ideal environments, it is assumed that users and the base stations are deployed randomly within an area of 30m × 30m as was shown in Figure 3-2. An LTE system with OFDM modulation and a channel model described in section 3.1 with low user mobility is assumed. The channel state information is assumed to be available at the receive side and the shadow fading parameter was assumed to be equal to 8 dB, and the antenna gain for BSs and UEs is equal to one. The path loss exponent is assumed to be equal to 4, with the noise power being equal to -100 dBm. The set of possible transmit power levels \( P \) was chosen as: \{-10, -5, 0, 5, 10\} dBm, with 5dBm steps being considered a sufficient resolution for transmit power level [12]. However, only two power level, \{-10, 10\} dBm can be considered sufficient and increasing the number of power levels does not necessarily increase the performance under this setup as shown in Figure 3.3 a and b. per each transmission, the positions of base stations and user terminals are randomly changing, simulating over all transmissions, the capacity is plotted in a cumulative distribution function (CDF) form. Simulations are performed for setups of three, four, and five base stations.
Figure 3-3: Average capacity for greedy search vs the number of power levels for a simulation environment stated in table 3.1

Table 1: Simulation assumptions for Figure 3.3. User assignment to base stations can be either random or to the base station with maximal channel norm

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8dB</td>
<td>30m × 30m</td>
<td>-100</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>10 dBm</td>
<td>-10 dBm</td>
</tr>
</tbody>
</table>
4 Enhanced Neural Network Design for Power Control

This chapter examines the best design choices for neural network based power control in high-density cellular systems. Predicting optimal power levels based on channel norms is straightforward when the system setup adopted by [12], in which a single user connects to each base station, is assumed. Moreover, as prediction is easier to perform in cases with smaller number of users/base stations a smaller number of user-base station pairs initially considered and expanded later on. While unrealistic, the setup assumed by [12] is sufficient to determine how the hyper-parameters and the neural network structure can be adjusted to perform good predictions in more realistic scenarios. It must be emphasized that in practical scenarios, where channel measurements are less volatile than the generated stochastic channel values, it is reasonable to anticipate improved system performance due to the adaptability of neural networks to more consistent training data distributions.

4.1 Activation Function

Figure 4-1 shows the histogram of the logarithm of average channel matrix norm between users and the base stations for the case of four users and four base stations, drawn from 15000 samples. The strong variations in the values of this parameter, from Order of 0.0001 to order of 10000, suggest that using a rectified linear unit or other non-linearities that do not bound the input data might be problematic in terms of convergence. Scaling the values suppresses smaller values of the channel norm and is thus undesirable. Using a bounded rectified linear unit is also problematic because a significant portion of the channel norm falls in a regime where the channel is large with an ambiguous threshold. Moreover, there should be a difference even between larger values, as a value in the order of 1000 is still more significant than one of order of 10000, and therefore thresholding on the input values does not help. Subsequently, sigmoid activation performs better than the rectified linear unit (with or without scaling/thresholding).
4.2 Greedy Search for Transmit Power Level Optimization

As training is performed offline, the computational complexity of the algorithm used for training is far less important than the run-time computational complexity and the neural network approach provides the most benefit when trained to provide a performance close to greedy search. Therefore, greedy search is used for both generating the training data set and evaluating the overall performance of considered schemes. As has been shown in [12], neural networks trained with greedy searches outperform networks trained distributed pricing and belief propagation methods. Therefore, within the scope of this work focus is centered on reducing the training time and computational complexity of neural networks trained with a greedy search.

For each transmission frame, i.e. simulation cycle, base stations and users are randomly deployed and the channel gain parameters are generated. The average capacity for every possible power level combinations is then calculated and once the average capacity is calculated for all possible base station power level vectors, the combination providing the highest average link capacity is declared optimal. In short, the procedure for simulating the greedy search algorithm at every transmission is as follows:
Algorithm 2: Greedy search for power level optimization

1. Deploy base stations and users in random positions

2. Generate The values for the channel

   For $0 \leq i \leq K_{UT} - 1$ do:

   for $0 \leq k \leq K_{BS} - 1$ do:

   Compute physical distances $d_{ik}$ between user $i$ and base stations $k$

   Compute the path loss between user $i$ and base station $k$ for as:

   $\Lambda_{ik} \propto d_{ik}^\alpha \eta_{ik}$

   Generate the values the normalized channel between user $i$ and base station $k$, $H_{ik}$ with zero mean and unit variance complex normal distribution

   Compute the channel gain matrix as $G_{ik} = \frac{1}{\sqrt{\Lambda_{ik}}} H_{ik}$

3. for all possible values of transmission power level vectors $\{P = (P_0, P_1, \ldots, P_{K_{BS}}) | P_k \in \{-10, -5, 0, 5, 10\} \text{dBm}\}$:

   for $0 \leq i \leq K_{UT} - 1$ do:

   Compute average MMSE capacity of users:
\[ C_{\text{MMSE},i} = \log_2 \left( \det \left( \mathbf{I}_{NR} + \frac{P_i}{N_T} \tilde{\mathbf{R}}_{NI,i} \mathbf{G}_i \mathbf{G}_i^H \right) \right) \]

Where:

\[ \tilde{\mathbf{R}}_{NI,i} = \sum_{k \neq \text{BS}(i)} \mathbf{G}_{ik} \mathbf{G}_{ik}^H + \sigma_n^2 \mathbf{I}_{NR} \]

(BS(i) is the base station assigned to user i)

Compute the average capacity corresponding to the vector \( \mathbf{P} \):

\[ C(\mathbf{P}) = \frac{1}{K_{UT}} \sum_{i=1}^{K_{UT}} C_{\text{MMSE},i} \]

4. Choose the value for \( \mathbf{P} \) corresponding to the maximum \( C \):

\[ \mathbf{P}_{opt} = \arg \max_{\mathbf{P}} C \]

Return \( \mathbf{P}_{opt} \)

### 4.3 Neural Network Training Procedure

As discussed in Chapter 2, training, validation and test datasets are required. Each of these datasets consist of a set of vectors and a desired label for each vector. The vectors are fed into the neural network and the network is trained such that by adjusting internal parameters, i.e. the weights in each layer, it can produce labels that are similar to the desired labels. In our problem, the vector being input to the neural network is the Forbenious norm of the channel gain matrices between
users and the base stations and the desired output, or the labels, is optimal transmit power levels. The neural network should thus be able to read the norm of channel gain, and predict near-optimal power levels as identified by the greedy search. Therefore, values for the channel generated and the corresponding optimal power levels are determined by a greedy search. The generated data and labels are then divided into training, validation, and test sets. The network is trained with the training set with small adjustments made based on validation and the performance of the predictor, i.e. the accuracy of the predictions, on the test set. Once the network is trained, the system capacity using neural network power control is compared with the system capacity under greedy search.

**Algorithm 3: Training and assessment of the neural network as a power control device**

1. Generate the training, validation, and test set

   a. **For** the size of the training set **iterate**:

      i. Deploy base stations and users in random positions

      ii. Generate The values for the channel

      1. Calculate physical distances $d_{i,k}$ between user $i$ and other base stations $k$ for $0 \leq i \leq K_{UT} - 1$ and $0 \leq k \leq K_{BS} - 1$

      2. Compute the path loss between user $i$ and base station $k$ for $0 \leq i \leq K_{UT} - 1$ and $0 \leq k \leq K_{BS} - 1$ as:

\[
\Lambda_{ik} \propto d_{ik}^\alpha \eta_{ik}
\]
3. Generate the values the normalized channel between user i and base station k, $H_{ik}$ with zero mean and unit variance complex normal distribution, for $0 \leq i \leq K_{UT} - 1$ and $0 \leq k \leq K_{BS} - 1$

4. Compute the channel gain matrix as $G_{ik} = \frac{1}{\sqrt{A_{ik}}} H_{ik}$ for $0 \leq i \leq K_{UT} - 1$ and $0 \leq k \leq K_{BS} - 1$

iii. for all possible values of transmission power level vectors $\{P = (P_0, P_1, \ldots, P_{K_{BS}}) | P_k \in \{-10, -5, 0, 5, 10\} dBm\}$ do:

$$C_{MMSE,i} = \log_2 \left( \det \left( I_{NR} + \frac{P_i}{N_T} \tilde{R}_{Ni,i} G_{ii} G_{ii}^H \right) \right)$$

Where:

$$\tilde{R}_{Ni,i} = \sum_{k \neq i} G_{ik} G_{ik}^H + \sigma_n^2 I_{NR}$$

And

$$C(P) = \frac{1}{K_{UT}} \sum_{i=1}^{K_{UT}} C_i(P)$$

iv. Choose the value for $P$ corresponding to the maximum $C$

$$P_{opt} = \ arg \max_P C$$

v. Compute Forbenious norm of the channel $G_{ik}$ from user i to BS k in a matrix $G_{Ni,k}$ for $0 \leq i \leq K_{UT} - 1$ and $0 \leq k \leq K_{BS} - 1$
vi. Store the obtained \((G^{iteration}_{N,k}, P^{iteration}_{opt})\) for the optimal \(P\) corresponding to \(G^{iteration}_{N,k}\), with one hot notation in \(P^{iteration}_{training}\).

b. Repeat a for the validation and test dataset

2. Train the neural network based on the procedure described in section (2.2.5), with \(G^{iteration}_{N,training}\) as the input and the vector \(P^{iteration}_{training}\).

3. Observe the performance of the trained neural network on the validation dataset \(G^{iteration}_{N,validation}, P^{iteration}_{validation}\). Adjust minor modifications, e.g., number of training steps.

4. Once the modifications are done, test the performance of the network on the test data set \(G^{test}_{N,validation}, P^{test}_{validation}\).

5. Observe the performance of the model as a power control device

   a. For enough iterations, do:

      i. Deploy the base stations and user terminals

      ii. Generate the random values for the channel

      iii. Compute the Forbenious norm of the channel values, from user \(i\) to BS \(k\), for \(0 \leq i \leq K_{UT} - 1\) and \(0 \leq k \leq K_{BS} - 1\), and obtain the and by placing its rows along the first row, reshape the obtained matrix into a vector \(G^{iteration}_{N}\)

      iv. Feed the values of \(G^{iteration}_{N}\) to the neural network obtained after step 3 to obtain the predicted optimal transmit power levels
v. Compute the average MMSE capacity of the with the predicted optimal power levels

b. Compute the CDF of the system capacity, and CDF of capacity using greedy search and maximum power transmission

4.3.1 Effect of Training Data Set Size

Increasing the training data set size increases the prediction accuracy of the model as shown in Figure 4-2 comparing the prediction accuracy based on 1000 and 3000 samples. While smaller training sets result in lower computations in the offline phase, proper training results in robust predictor performance to channel variations and minimizes required updates to channel fluctuations as shown in Figure 4-3. Unlike [12], where LII regularizer with weight decay parameter $\lambda = 10^{-4}$ is used, dropout is the only utilized regularization method in both cases.
Figure 4-2: Validation accuracy vs Training set size (3000 vs 1000 training samples).

Figure 4-3: Validation loss vs Training set size (3000 vs 1000 training samples).

4.4 Choice of Regularization Method

Dropout is compared with LII norm in this section as the regularization method of choice. LII norm provides a slight increase in the prediction and validation accuracies as observed in Figure 4-6. A three-layer neural network (two hidden layers) trained with a set for a 16-by-16 MIMO system is considered. Without any regularization, the performance of an SGD optimizer with a learning rate $2 \times 10^{-4}$ and 500 hidden units per layer is shown in Figure 4-4:
The use of LII regularization with $\lambda = 10^{-4}$ is examined and shown in figures below. LII regularization provides a slight increase in performance at a computational disadvantage imposed by the computation and backpropagation of the gradient of LII norm of weights.
Supplementing the LII regularizer with dropout improves performance and reduces tendency to overfit as shown in the figures below.

**Figure 4-6:** Training and validation accuracy with and without LII regularization.

**Figure 4-7:** Training and validation losses with LII regularization.
Figure 4-8: Training and validation accuracies with dropout, LII, LII and dropout, and with no regularizer

Figure 4-9: Train and validation loss for dropout and LII as a regularizer
Nevertheless, using dropout only provides the highest validation accuracy as observed in figure Figure 4-10.

**Figure 4-10**: Validation accuracy, dropout only and dropout and LII regularized

**Figure 4-11**: Validation losses-dropout regularized
4.4.1 Increasing Training Speed Using Adaptive Optimization

Simple Gradient Descent, as discussed in Chapter 2 and shown in Figure 4-4 performs slowly in the optimization process during the training of neural networks. Therefore, to improve training convergence, [33] proposed to pre-train the neural network to suitable. However, while pre-training increases the convergence speed, the convergence remains slow with 2000 epochs needed for network convergence. An alternative approach for reducing training time is by adaptively changing the step size using Adam described in [25] and explained in chapter 2. As Figure 4-12 and Figure 4-13 show the training speed substantially improved, with convergence reached in 200 epochs, while avoiding fluctuations in the loss function optimization.

![Figure 4-12: Validation accuracies for Adam vs SGD.](image)

**Figure 4-12:** Validation accuracies for Adam vs SGD.
Figure 4-13: Training and validation loss for Adam optimizer (with dropout).

While Adam improves accuracy and convergence speed, training must be terminated in a timely manner to prevent overfitting. Nevertheless, the network’s sensitivity to the termination point is reduced with the use of fewer hidden units. Reducing the number of hidden units from 500 to 200 also provides better validation accuracy as shown in Figure 4-14.
Figure 4-14: Validation accuracy with different number of hidden units.

Figure 4-15: Training and validation loss with different number of hidden units.

The convergence speed for Adam optimizer can be further improved by adopting a higher learning rate. For example, a learning rate of 0.01 provides faster convergence when compared to a learning rate of 0.0002. However, higher learning rates might cause oscillations which is undesirable. In [25], $\alpha = 0.001$ is suggested as a good parameter for the learning rate in machine learning applications. Upon experimenting with different values, $\alpha = 0.002$ is chosen to avoid oscillations in the learning curve as verified by Figure 4-16.
Figure 4-16: Training and validation accuracies for different Adam learning rates.

Figure 4-17: Validation losses for different Adam learning rates.
5 Unified Neural Network with Multiple Outputs as a Predictor

To reduce the computational complexity during training and runtime, we present a structure which aims to compute the power levels for all base stations in one forward pass. The rationale behind this idea is that when using a dedicated neural network to obtain the predicted optimal power levels for each base station, all neural networks are being fed with the same input data, which is the vector $\mathbf{G}_N$ representing the norm of channels between all users and base stations. Therefore, neural networks are being trained to learn multiple representations for the same input data. Such a redundancy can be eliminated by learning a single representation of the input vector $\mathbf{G}_N$, which can be used for predicting all power levels for all base stations, and defining one neural network with multiple output vectors, as opposed to multiple neural networks with single output vector, with each vector corresponding to one of the base stations. To train this neural network, a loss function is defined as the sum of cross entropies of the predicted power level for each base station and their respective optimal power level from the greedy search.

The proposed structure, shown in Figure 5-1, feeds the norm of the channel gains into a single neural network with multiple outputs to reduce the training and testing computational complexity by a factor of $K_{BS}$. During the forward pass, the neural network computes the representation $u_L$ based on which a logistic regression is performed in the last layer, and the output vectors $u_{O,k}$ are produced ($0 \leq k \leq K_{BS}$) for all base stations, thus reducing the computational complexity of the forward pass by a factor of $K_{BS}$. 
5.1 Training Procedure and Performance Optimization

To accommodate the case of a neural network with multiple outputs, the network can be trained multiple times with different loss functions corresponding to each output. At each step, the loss function would be the cross entropy between one of the output vectors of the neural network and the corresponding labels. The computational complexity during the forward pass is thus reduced by a factor of $K_{BS}$ while the training computational complexity remains the same as the training needs to be performed $K_{BS}$ times. An alternative approach is to define the loss function as the average cross entropy between each output of the neural network and the corresponding labels. Training would then be performed only once and the computational complexity of both forward pass and the training would be reduced by a factor of $K_{BS}$. However, under this approach the network tends to learn features that provide good performance for some base stations while
performing poorly for others, i.e. the average cross entropy is not minimized by minimizing the cross entropy individual base stations. Moreover, at each step, the corresponding error for all output vectors is computed and needs to be back-propagated, i.e. the training complexity cannot be contained by computing a single loss and back-propagating only the corresponding error. Figure 5-2 shows the performance of the system for a neural network predicting the power levels for two users based on a single representation with two hidden layers consisting of 200 hidden units each, a dropout rate equal to 0.5 and a learning rate of 0.002.

![Figure 5-2: Training and validation accuracies, two predictions with shared representation.](image)

![Figure 5-3: Training and validation losses, two predictions with shared representation.](image)
Utilizing a large training set increases the performance of the power level prediction at the expense of increased training time. In dynamically varying environments, e.g. when the number of users, propagation parameters, etc. change rapidly, the neural network needs to be continuously updated. This highlights the importance of the compactness of the training set and computations required for training and running the power control model to make it appealing for use in wireless communications.

There are ways to reduce the cost of the training procedure, for instance using convex optimization based methods to generate the training set instead of a greedy search. As an example, the authors of [34] impose a set of constraints on the power optimization problem in the uplink of a two-tier femtocell network, and then show that the power level optimization is convex given those constraints and optimize power in a few iterations of power level updates. A similar approach can be taken in the downlink, and it has the benefit of providing more training data at a lower cost. This implies higher scalability to accommodate more users and base stations in the system. Similarly, the approach taken in this work can be easily applied in the uplink as well. Nevertheless, the greedy search provides more flexibility in imposing constraints and policies on which samples to choose. Another method is to rely on prior information about the data, for instance the symmetries within the training dataset to increase the training efficiency in terms of convergence speed and number of required training samples. One way to consider the symmetries within the training dataset is by convolutional neural networks, imposing shared weights applied on the columns/rows of the channel norm matrix used as a feature for performing predictions. The shared weight implies that all user terminals/base stations are similar to each other. Adding convolutional layers in this fashion also helps because it makes the network deeper while maintaining the same overall number of parameters and therefore same complexity. On the other hand, making the network deeper with a fully connected layer does not add a meaningful change in the performance of the prediction model. This approach of using a convolutional layer is studied in the following section.
5.2 Shared Weights in the First Layer

The simulation setup assumes base stations have the same capabilities and a similar assumption is adopted for user terminals. To reflect this information in the model structure, a similar transformation is applied across users and base stations by exploiting shared weights, meaning that the weights of the channel matrix in the first layer are shared among the transformations applied on different base stations/users. This essentially is a convolutional structure, except that the application of a pooling layer is not examined here. Figure 5-4 shows the validation accuracy for the two cases:

![Figure 5-4: Prediction accuracy for the convolution.](image)

Application of a convolutional layer also reduces the number of parameters in the fully connected layers which significantly reduces the complexity in the forward and backward pass. Using the shared weights, the learning rate can be increased without losing performance. As observed in bellow:
5.3 Hidden Layers and Residual Connections

While an additional hidden layer can be added to provide representations, the increased computations (an additional 200-by-200) matrix multiplication, potential overfitting and identity mapping (higher layer learning to copy the representations from the previous layer) are undesirable associated outcomes. Identity mapping can be addressed using residual connections as shown in Figure 5-7 and overfitting can be controlled by controlling the model size and regularization. As for computational complexity, using shared weights (explained in 3.11.3) significantly reduces the number of parameters in the fully connected layers and reduces computational complexity. Therefore, eliminating the additional hidden layer increases the performance of the predictions as shown in Figure 5-8.
**Figure 5-6:** One hidden layer performs better with shared representations.

**Figure 5-7:** Validation accuracy for residual mappings.
5.4 Reducing the Number of Hidden Units

Figure 5-8 shows the prediction accuracy for two cases where the number of hidden units is 50 and 200 per hidden layer. It is observed that by reducing the number of hidden units, the computational complexity of the forward pass and training can be considerably reduced without degrading performance. When the number of users and base stations is small, (e.g. four users and four base stations) the largest part of the complexity in the forward pass is in obtaining the second layer representations based on the first layer, an $N_{\text{hidden}}^2$ multiplication. The computational complexity of the forward pass is $K_{\text{UT}} \times K_{\text{BS}} \times N_{\text{hidden}} + N_{\text{hidden}}^2 + N_{\text{hidden}} \times K_{\text{BS}} \times N_{\text{class}}$ where $N_{\text{class}}$ is the number of power levels. Now, for $N_{\text{hidden}} = 200$, and $K_{\text{UT}} = K_{\text{BS}} = 4$ and $N_{\text{class}} = 5$, this would be $3200 + 40000 + 4000 = 47200$ multiplications. $N_{\text{hidden}} = 50$ results in $800 + 2500 + 1000 = 4300$ multiplications, i.e. more than an order of magnitude less multiplications. The number of hidden units can be further reduced using shared weights, which further reduces the computational complexity of the hidden layer, limits the number of parameters in the system, prevents overfitting and provides faster convergence. Since the backpropagation can be seen as a forward pass for the error gradients in a reverse neural network, same argument can be applied in that case and therefore limiting the number of hidden units significantly reduces the complexity of both training and test/online time.

![Figure 5-8: Number of hidden units needed for shared representation.](image-url)
5.5 Activity Function in the Second Layer

While the activity function of choice in the first layer is a sigmoid function, utilizing a different activity function in the second layer can improve performance. Specifically, as shown in Figure 5-9, choosing a ReLU in the second layer improves the system performance.

![Validation accuracy, ReLU vs Sigmoid in the second layer.](image)

**Figure 5-9:** Validation accuracy, ReLU vs Sigmoid in the second layer.

5.6 Mini-Batch Size

The loss function at each step can be designed as the maximum of the mini-batch loss function corresponding to all $K_{BS}$ output vectors of the neural network, such that only the loss for the output with the most severe performance is backpropagated. The network would then arrive to a more well rounded hidden representation of the input data and perform equally well for all outputs. Computational complexity would also be reduced for the training procedure as well as the forward pass by a factor of $K_{BS}$. Additionally, this method acts as a regularization as it prevents the loss to converge easily and causes some oscillations at the beginning of training. In general, reducing the mini-batch size increases computations in each epoch in the training time while along with back propagations in the backward pass, resulting in higher accuracy and faster convergence in terms of number of epochs. However, the overall training time increases due to increased backward
passes. The authors of [35] propose scaling the mini-batch size learning rate at the same time to reduce the training time of the neural networks. Figure 5-10 shows validation accuracy for mini-batch sizes 16 and 100. The smaller mini-batch size achieves higher accuracy and converges in a lower number of epochs, but the overall training time is increased. On the other hand, the larger mini-batch size converges faster (in terms of overall training time) with a slightly lower accuracy.

![Figure 5-10: Effect of mini-batch size on performance.](image)

5.7 Reducing the Complexity of the Greedy Search by Limiting the Search Space

As has been shown in the previous section, under the simulation assumptions considered within this work, having more than two power levels does not imply a meaningful increase in the system performance. While this is not necessarily generalizable to other cases without proper investigation, it is worth mentioning that in this case the noise component in the SINR is negligible compared to the interference component as expected in high-density cellular communication systems. Subsequently, the ratio of power levels is more important than their absolute value in
cases where noise power is very low compared to interference power. Therefore, it is somewhat reasonable to argue that in binary power levels provide sufficient granularity for comparable performance with higher granularities at a lower computational cost. It was also noted that under the assumptions made for the environment, which are similar to the ones made in [12], even when considering five power levels, the outcome of the greedy search often tends to be the lowest possible power level, -10 dBm, or the highest, 10 dB, with less cases in between. Figure 5-12 shows the histogram on the results for power levels of the base stations, while performing a greedy search aiming at maximizing the average rate. As it is seen, in most of the cases, the power levels are either the highest level or the lowest. We could therefore argue that we could limit the number of power levels considered to only two with little to no loss of optimality in the training. Additionally, the lower the number of power levels, the more training samples can be generated with the same computational resources, and therefore the better accuracy we can get from the neural network predictor. Therefore, to have a limited number of power levels, having two power levels, -10 dBm and 10 dBm, or three, -10 dBm, 0 dBm, and 10 dBm is sufficient. We consider the former, for most of the experiments in this work unless otherwise stated.
Figure 5-11: CDF of system capacity with varying number of power levels.

Figure 5-12: Histogram of number optimal power levels for a base station with greedy search. (index 0 corresponds to power level -10 dBm and index 4 to +10 dBm).
5.8 Improving Minimal User Rate

As discussed in Chapter 3, the employment of the average rate or the sum rate as the metric of performance does not provide any provisions for attaining a specific level of UT (QoS), and often leads to low QoS levels for users with unfavorable channel conditions. Figure 5-13 clearly illustrates the large gap between the average user rate and the minimum user rate when utilizing the average user rate as the optimization metric. Instead of maximizing the average user rate, the minimum user rate can be utilized as the optimization performance metric and a greedy search is accordingly performed as follows:

Algorithm 4: Modified Greedy search for power level optimization

1. Deploy base stations and users in random positions

2. Generate The values for the channel

\[
\text{For } 0 \leq i \leq K_{UT} - 1 \text{ do:}
\]

\[
\text{for } 0 \leq k \leq K_{BS} - 1 \text{ do:}
\]

Compute physical distances \(d_{i,k}\) between user \(i\) and base stations \(k\)

Compute the path loss between user \(i\) and base station \(k\) for as:

\[
\Lambda_{ik} \propto d_{ik}^{\alpha} \eta_{ik}
\]

Generate the values the normalized channel between user \(i\) and base station \(k\), \(H_{ik}\) with zero mean and unit variance complex normal distribution
Compute the channel gain matrix as $G_{lk} = \frac{1}{\sqrt{\lambda_{lk}}} H_{lk}$

3. for all possible values of transmission power level vectors \( \{ P = (P_0, P_1, \ldots, P_{BS}) | P_k \in \{-10, -5, 0, 5, 10\} \text{dBm} \}$:

   for $0 \leq i \leq K_{UT} - 1$ do:

   Compute average MMSE capacity of users:

   $$C_{MMSE,i} = \log_2(\det \left( I_{NR} + \frac{P_i}{N_T} \bar{R}_{NI,i} G_{ii} G_{ii}^H \right))$$

   Where:

   $$\bar{R}_{NI,i} = \sum_{k \neq BS(i)} G_{lk} G_{lk}^H + \sigma_n^2 I_{NR}$$

   (BS(i) is the base station assigned to user i)

   Compute the average capacity corresponding to the vector $P$:

   $$C(P) = \frac{1}{K_{UT}} \sum_{i=1}^{K_{UT}} C_{MMSE,i}$$

   Choose the value for $P$ corresponding to the maximum $C$:

   $$P_{\text{opt}} = \arg \max_P C$$

   Return $P_{\text{opt}}$
And the steps to train and observe the performance of the neural network are also modified accordingly. To increase accuracy, a larger training set size of 90000 samples is examined and dropout is not utilized to avoid under-fitting due to the lower disparateness of power level combinations maximizing the minimum rate. Essentially, once a channel between a user and assigned base station is poor, it is harder to maximize the user rate by altering the power levels of interfering base stations. Nevertheless, the updated optimization framework resulted in a 16% increase in the minimal user rate at the expense of reducing the average user rate by around 10%.

**Figure 5-13:** CDF of average and minimum rate for greedy search (maximizing average rate) and maximum power. Greedy search with average rate as objective deteriorates the minimum rate when compared to maximum power transmission.
Figure 5-14: Training and validation losses, maximizing the minimum rate (90000 training samples).

Figure 5-15: Training and validation accuracies, maximizing minimum rate (90000 samples).
5.9 Having More than One User per Base Station

While the discussion in [12] was limited to the case where one base station is assigned to exactly one user at a time, in a more realistic scenario, the number of users exceeds the number of base stations. Under such an assumption, there would be at least one base station connecting more than one user, which requires the introduction of user scheduling means such as round robin and proportional fair scheduling, or more advanced techniques such as dirty paper coding for simultaneous transmission. However, within the scope of this work we are designing a scheme for power control, which can be applied to both cases and therefore, for the sake of simplicity, we opt for round robin scheduling, detailed in following section.
5.10 Overall System Performance

Figure 5-17 shows the performance of the overall system comparing a fully connected neural network with residual mapping versus a network consisting of a convolutional layer followed by two fully connected layers and residual mappings. The first network has 50 hidden units per layer, learning rate Adam optimizer with learning rate 0.002, and 700 training epochs with 3000 training samples for $16 \times 16$ MIMO with four users and four base stations and five possible power levels for the base stations ranging between -10 dBm and 10 dBm. As observed in section 5.7, only two power levels are enough when considering the simulation assumptions adopted by [12]. Nevertheless, to benchmark our model and show its capacity for making the predictions in more complex scenarios, we chose this value for this simulation. $16 \times 16$ MIMO makes the generation of the training set more expensive. However, it also shows the scalability of this approach with respect to the number of antennas since only the Frobenious norm of the channel gain matrix is used for processing. The second network, shown to provide closest performance to greedy search, has a convolutional layer with a filter size of one-by-four, depth four and sixteen hidden units at the hidden layers with residual mapping. The learning rate for this case is 0.01 and the dropout rate is 0.5 for both cases in the fully connected layer.
5.10.1 Five Users and Five Base Stations

Increasing the number of base stations and user terminals in the system adversely affects the performance by degrading the prediction accuracy. Figure 5-18 and Figure 5-20 show the prediction accuracy and the performance of the overall system for the case of five users and five base stations, respectively. The prediction accuracy is lower than the previous case (four users, four base stations, in Figure 5-17) and adding the convolution significantly enhances the predictor performance. A dropout rate equal to 0.1 is chosen to allow the network to converge while still benefitting from the regularization capability of dropout. However, as we are limiting the size of the model and therefore its ability to over-fit the data, dropout and regularization can limit the model’s ability to converge and fit the training data.

**Figure 5-17:** Performance of the overall system, proposed structure vs the baseline (single hidden layer with sigmoidal activation).
**Figure 5-18:** Validation accuracy for five users, using shared weights vs only using fully connected layers.

**Figure 5-19:** Training and validation losses for the convolutional structure.
5.10.2 Multiuser Systems: Round-Robin Scheduling

Training a model based on round-robin between users connecting to the same base station is comparably a harder problem to solve since it causes a smoothing on the sensitivity of the performance w.r.t. the power levels due to averaging on user rates. Subsequently, the training set size for the neural network must be increased to maintain predictor accuracy at the expense of increased computations. The increased computations can be contained by reducing the number of power levels being considered from five to two. Therefore, for the case of round-robin, we consider 30000 training samples instead of 3000 discussed in the previous cases. However, by reducing the number of power levels from 5 to 2, the computational complexity of generating the training samples can decreases by a factor of \((\frac{5}{2})^{K_{BS}}\). With 5 base stations, this figure would be around 100 corresponding to a 10-fold decrease in the overall complexity of training generation. To enhance the performance of the classifier, the depth of the convolution was increased to 16, and the number of hidden units was increased to 64, resulting in a significant increase in the complexity.
of neural network both during training and testing. The system performance is shown in Figure 5-21 and Figure 5-22.

**Figure 5-21:** training and validation accuracy for round-robin, five users and five base stations.

**Figure 5-22:** Overall performance for round-robin, five users and five base stations.
An alternative approach for reducing computational complexity is reverting to using one neural network per base station, each of which having a small number of neurons. Compared to [12] and the initial approach, the overall computational complexity is significantly lower both during the training and test time while the accuracy is higher as observed in Figure 5-26. The performance of this updated model is first examined with depth four for the convolution, sixteen hidden units in terms of prediction accuracy and a training set size is of 9000. As it is seen the figure below performance is improved over the previous case:

![Training and validation accuracy](image)

**Figure 5-23:** Training and validation accuracy, one neural network per base station, reducing the number of fully connected neurons.

The average prediction accuracy and overall performance is then examined with the training set size set to 18000:
Figure 5-24: Training and validation accuracies, multiple ConvNets (18000 training samples).

Figure 5-25: Training and validation losses, multiple ConvNets (18000 training samples).

Having close training and validation accuracies indicates the model is not overfitting, i.e. the training set size can be further reduced without significant loss in performance. Figure 5-27 shows the prediction accuracy in case of a smaller training set size.
Figure 5-26: Performance of overall system, Multiple ConvNets vs Single ConvNet, multiple smaller ConvNets require less computations for training and forward pass and less training samples.

Next, increasing the number of hidden units without increasing the depth in the convolution, to increase the stability in the convergence, is examined. The training set size is reduced from 18000 samples to 9000 samples.
**Figure 5-27**: Training and validation accuracies, one ConvNet per prediction, (9000 training samples).

**Figure 5-28**: Training and validation losses, one ConvNet per prediction, 9000 training samples.
Figure 5-29: Performance of overall system for round-robin, one ConvNet per prediction, (9000 training samples vs 18000 training samples).

As it is seen this reduction does not make a significant difference in the overall performance of the power control algorithm. For improving the stability in the convergence, the depth in the convolution can be increased to 6 and the number of hidden units per layer can also be increased to 30. Figure shows the learning curves for such cases.
Figure 5-30: Training and validation loss for different convolutional layer depths.

Figure 5-31: Training and validation accuracies for different convolutional layer depths.
5.10.3 Under-Fitting in Larger Training Sets

An example illustrating the negative effect of regularization when employing large training sets is shown in Figure 5-32 and Figure 5-34, in which a dropout rate of only 0.1 significantly degrades the system performance when the training set size is 18000. In general, dropout (and regularization) is most useful when the available training data is limited compared to the model capacity. On the other hand, by limiting the model size through convolutions and reducing the number of hidden units, thus limiting its ability to over-fit, and increasing the training data, dropout becomes unnecessary at best if not harmful as in this case (Figure 5-32 and Figure 5-34).

**Figure 5-32:** Prediction accuracy for dropout rate of 0.1 vs no dropout.
5.10.4 Random Assignment of Exactly One User per Base Station

The case of having one user randomly assigned per base station is considered to study scenarios in which optimal assignment cannot be performed due to high user mobility. A 1-by-1 convolution
without non-linearity and bias with depth one (for automatic pre-scaling) is introduced with a training set size 9000. The results, a validation accuracy around 86.1% and training accuracy of 87% as shown in Figure 5-36, indicate that the model is not overfitting and therefore a potential for using a smaller training set.

**Figure 5-35**: Training and validation losses, random base station assignment, five users and five base stations, 9000 training set size.
**Figure 5-36**: Training and validation accuracies, random base station assignment, five users and five base stations.

**Figure 5-37**: Overall system performance, random base station assignment, five users and five base stations.

Random user assignment, intuitively reduces the average rate minimum rates are worse. For this reason, the relative improvement in both minimum and sum rates are higher when compared to optimal user assignment, particularly for the minimum rate as shown in Figures (3.58-3.40), where the system performance with depth 6 and 24 hidden units is shown.
Figure 5-38: Training and validation losses for maximizing the minimum rate, random BS assignment, four users and four base stations (1000 samples training set).

Figure 5-39: Training and validation accuracies for maximizing the minimum rate, random BS assignment (1000 samples training set size).
Summary and Conclusions

Wireless communication systems are complex systems with different controllable parameters, when looking at large scale systems with many different users, efficient use of machine learning techniques is among the most promising approaches for efficient operation of these systems. For using a neural network in a wireless communication setting, there are general considerations associated with machine learning, such as the ones discussed in this work (parameter sharing, adaptive optimization for faster convergence, etc.), along with aspects critical to wireless applications such as computational efficiency due to the dynamic nature of wireless systems and the processing constrains on mobile devices. As with other machine learning problems, it is always beneficial to have more data, less parameters, and faster training. Incorporating prior knowledge of the data structures, such as the parameter sharing proposed in this work, can significantly improve the network performance. Network size, depth, etc. should be chosen large enough that they allow the network to fit the data but not overfit.
We have demonstrated the feasibility of solving optimization problems in real time, in this case for power levels, using neural networks. Neural networks have been shown, in our proposed application, to capable of improving convergence speed without requiring multiple iterations and the transmission of messages over the air. The ability of neural networks to efficiently approximate complex algorithms, in this case the greedy search for power control in a small cell, has also been demonstrated under different settings for base station assignments, ignoring the intracell interference and users connecting to base station with the strongest signal, round robin base station assignment and random one user per base station assignment. We have shown that neural networks provide a flexible foundation for optimizing different objective functions, the average MMSE rate as well as the minimum rate. Moreover, we have shown that using appropriate neural network structures reflecting the temporal and spatial structure of the input data leads to significant performance increases as well as significant reductions in computational complexity of the training. Finally, as some optimization problems are harder to perform for the neural network, and in those cases, we have shown that more data can be useful in those cases to enhance the results.
References


1412.6980 [cs.LG], December 2014.


