Efficient Deep Learning for Massive MIMO Channel State Estimation

By

MASON DEL ROSARIO

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Zhi Ding, Chair

Lifeng Lai

Khaled Abdel-Ghaffar

Committee in Charge

2023

Acknowledgements

The research conducted in this dissertation was supported by the National Science Foundation under Grant No. ECCS-1711823, ECCS-2029027, and CNS-2002937.

Contents

List of Figures

List of Tables

Abstract

Future wireless communications networks will implement massive MIMO technologies where a base station (BS) with large multiantenna arrays serve a large number of user equipment (UE) terminals. Such multiantenna arrays enable high capacity communications via beamforming, as evidenced by work in information theory. To achieve capacity in massive MIMO networks, the base station requires accurate estimates of downlink channel state information (CSI) in order to precode (decode) transmitted (received) messages.

CSI can be estimated using known pilot signals. In the special case of time-division duplex (TDD) mode, channel reciprocity allows the BS to estimate the downlink CSI via pilots in uplink transmissions. However, in frequency division duplex (FDD) mode, channel reciprocity between uplink and downlink channels is weak, and the BS must also rely on feedback from the UE to estimate downlink CSI. Specifying an appropriate CSI feedback scheme is a key issue and involves reducing feedback bandwidth while maintaining accurate downlink CSI estimates.

Conventional methods for CSI feedback compression typically rely on compressive sensing (CS), which seeks to reconstruct high-dimensional data based on low-dimensional measurements (see for a survey of CS methods). Many CS methods are based on convex relaxations of an underdetermined least-squares problem, and such methods rely on iterative solvers (e.g., the proximal gradient method). When using an iterative solver, reconstruction can consume an undue amount of time even when measurements are available, making faster methods for reconstruction desirable.

Recent works in deep learning for compressed CSI estimation has presented viable alternatives to CS methods. These proposed methods typically employ convolutional neural networks (CNNs) to learn compressed representations of high-dimensional CSI. Deep learning architectures used in CNN-based works can be placed in one of two categories. The first category is CNN-based autoencoders, networks which utilize two subnetworks: an encoder

network which learns a low-dimensional representation with the original data as an input and a decoder network which estimates the original data as a function of the low-dimensional representation. The second category is unrolled optimization networks, which draw inspiration from the aforementioned CS methods by structuring the CNN as a finite number of repeated blocks with each block imitating an iteration of a given CS algorithm.

This dissertation explores both CNN autoencoders and unrolled optimization networks for CSI estimation while focusing on domain knowledge, including physical insight into the wireless channel or features of the communications protocol, to improve the performance of these CSI estimation networks with respect to accuracy, feedback rate, or network efficiency. Prior works have demonstrated superior performance of these architectures over conventional CS methods, and this dissertation investigates the myriad ways that domain knowledge of wireless channels and communications protocols can be leveraged to improve CSI estimation.

The first research direction discussed is spherical normalization, which involves scaling CSI data by the channel's power and can improve CSI estimation accuracy without increasing model complexity. The second research effort is deep differential encoding, which estimates the error in CSI estimates based on a one-step Markov model. We describe a deep differential encoder which offers a significant reduction in computational complexity when compared to state-of-the-art recurrent neural networks while achieving superior estimation accuracy. In the third research direction, we use sparse frequency domain pilots to estimate the truncated delay domain, enabling the usage of the delay domain CSI commonly adopted in deep learning based CSI estimation works. Additionally, we implement a heterogeneous differential encoder, which uses different network architectures at each timeslot and offers superior estimation accuracy over the prior homogeneous differential encoders. In the final chapter, we propose model reuse, where a smaller model is used multiple times on a relatively large input, and we show that this method can maintain comparable estimation accuracy while reducing computational complexity.

Chapter 1

Introduction

This dissertation details work in improving the accuracy and efficiency of deep learning methods for MIMO channel state information estimation. This chapter provides the necessary background to understand the contributions of the dissertation. Section 1.1 provides an overview of the MIMO channel and the importance of CSI estimation in MIMO-based communications networks. Section 1.2 discusses the pilot reference signals used in CSI estimation. Section 1.3 discusses MIMO channel models and introduces the primary channel model used in this work, the COST2100 model. Section 1.4 discusses prior work in compressive sensing for CSI estimation. Boldface lowercase (uppercase) letters indicate vectors (matrices). Unless otherwise specified, the norm $\|\cdot\|$ indicates the Frobenius norm. Superscripts ^{T} (^{H}) indicate the transpose (Hermitian transpose).

1.1 MIMO Channel Overview

In this work, we consider a MIMO channel with a multiple antennas ($N_B \gg 1$) at the transmitter (gNodeB or gNB) servicing a single user equipment (UE) with a single antenna. Under orthogonal frequency division multiplexing (OFDM) with N_f subcarriers, the

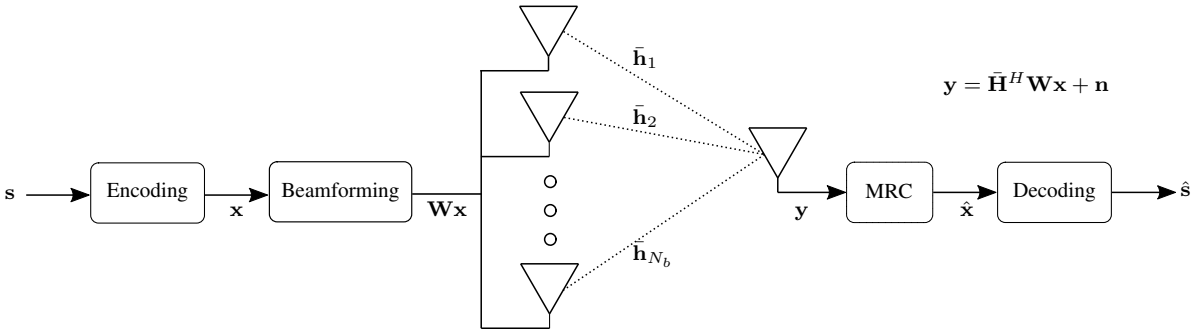


Figure 1.1: Example multi-antenna transmitter (BS, gNB) and single-antenna user equipment (UE) and relevant system values.

Table 1.1: MIMO system variables considered in this work.

Symbol	**Dimension**	**Description**
$y_{d,m}$	$\mathbb{C}^1$	Received downlink symbol on m-th subcarrier
$\mathbf{h}_{d,m}$	$\mathbb{C}^{N_b \times 1}$	Downlink channel on m-th subcarrier
$\bar{\mathbf{H}}_d$	$\mathbb{C}^{N_f \times N_b}$	Downlink CSI (spatial-frequency domain)
$\mathbf{w}_{t,m}$	$\mathbb{C}^{N_b \times 1}$	Transmitter precoding vector for m-th subcarrier
$x_{d,m}$	$\mathbb{C}^1$	Trasmitted symbol on m-th subcarrier
$n_{d,m}$	$\mathbb{C}^1$	Downlink noise on m-th subcarrier
$\tilde{\mathbf{H}}_d$	$\mathbb{C}^{N_f \times N_b}$	Downlink CSI (angular-delay domain)
$\mathbf{H}_d$	$\mathbb{C}^{R_d \times N_b}$	Truncated downlink CSI (angular-delay domain)

received symbols on the m-th subcarrier for the downlink at the receiver are given as

$$y_{d,m} = \mathbf{h}_{d,m}^H \mathbf{w}_{t,m} x_{d,m} + n_{d,m}.$$

where the individual system values are defined in Table 1.1, and a representative system model is viewable in Figure 1.1. The resulting downlink and uplink channel state information (CSI) matrices are given as

$$\bar{\mathbf{H}}_d = \begin{bmatrix} \mathbf{h}_{d,1} & \dots & \mathbf{h}_{d,N_f} \end{bmatrix}^H \in \mathbb{C}^{N_f \times N_b}.$$

To achieve near-capacity transmission rates, the transmitter needs access to an appropriate

estimate of $\bar{\mathbf{H}}_d$ [6]. Such estimates enable the use of linear precoding techniques (e.g., conjugate beamforming or zero-forcing beamforming) to realize appreciable spectral and power efficiency gains [7]. Downlink CSI estimation can be performed in time division duplex (TDD) by using uplink pilots due to channel reciprocity [8–10]. In contrast, frequency domain duplex (FDD) does not admit channel reciprocity due to frequency-selective channels, meaning CSI estimates must be acquired at the UE using pilot signals, and these estimates must be compressed then fed back to the BS.

1.2 Practical Pilot-based Channel Estimation in 4G/5G Networks

To estimate the downlink CSI in wireless networks, transmitters allocate pilot reference signals. To preserve spectral resources, pilots are restricted to a limited number of spatial-frequency positions, and the allocation of these pilots is defined in the 3GPP technical standards, TS 36.211 for 4G/LTE networks [11] and TS 38.211 for 5G/NR networks [12]. In these two standards, the pilots are called CSI reference signals (CSI-RS) or demodulation reference signals (DM-RS), respectively. Figure 1.2 shows valid placements of CSI-RS/DM-RS in the time-frequency resource grid as defined by TS 36.211 and TS 38.211.

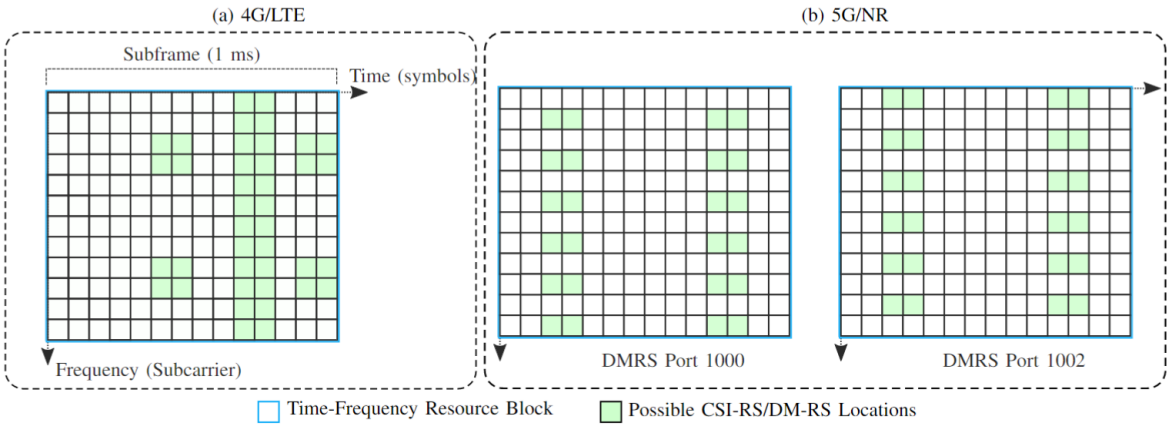


Figure 1.2: (a) LTE resource blocks with CSI-RS locations. (b) 5G NR resource blocks with DM-RS locations.

For the UE to estimate the downlink channel state at the known pilot locations, the UE can use a straightforward least-squares estimator. Denote the pilot matrix $\mathbf{X} \in \mathbb{C}^{M_b \times \tau}$ where τ is a given number of timeslots that is less than the coherence interval of the channel. The received signal at the UE ($\mathbf{y}_{\text{pilots}} \in \mathbb{C}^{1 \times \tau}$) based on the transmitted pilots is,

$$\mathbf{y}_{\text{pilots}} = \mathbf{h}_{\text{pilots}}\mathbf{X} + \mathbf{n}, \tag{1.1}$$

where $\mathbf{h}_{\text{pilots}} \in \mathbb{C}^{1 \times M_b}$ is the pilot matrix for a subset of the transmitter antennas such that $M_b < N_b$ and $\mathbf{n} \sim \mathcal{CN}(\mathbf{0}, \sigma^2\mathbf{I})$ is additive complex Gaussian noise. Pilot estimation based on $\mathbf{y}_{\text{pilots}}$ and $\mathbf{X}$ is straightforward using the least-squares solution,

$$\hat{\mathbf{h}}_{\text{pilots}} = \mathbf{y}_{\text{pilots}}\mathbf{X}^H(\mathbf{X}\mathbf{X}^H)^{-1}. \tag{1.2}$$

While many works using deep learning for CSI estimation do not address pilot estimation explicitly (i.e., they assume perfect CSI at the UE), a few such works have incorporated pilot estimation into the CSI feedback problem. These works typically apply deep learning to either A) improve upon the least-squares estimator of (1.2) or B) design the pilot matrix, $\mathbf{X}$. In [13], the authors propose a fully-connected network (FCN) which performs pilot allocation and coarse CSI estimation followed by a CNN-based attention layer for refining the estimate, and they demonstrate the efficacy of the proposed network compared to the MMSE estimator as well as other neural estimators. In [14], the authors propose to incorporate a trainable pilot design as a portion of an end-to-end model that directly maps pilots at the UE into feedback bits and then feeds those received bits at the BS into a precoding matrix, thereby avoiding explicit CSI estimation.

In this dissertation (namely, in Chapter 4), we propose an estimator of the truncated delay domain CSI (see Section 2.2.1 of Chapter 2) based on the sparse frequency domain pilots (i.e., $\hat{\mathbf{h}}_{\text{pilots}}$ of equation (1.2) above).

1.3 Geometric Stochastic Channel Models (GSCMs)

Ideally, the datasets used for the CSI estimation task would be derived from measurement campaigns (e.g., see [15–17]). However, the financial and labor costs of acquiring these datasets can be prohibitive in certain situations. In such situations, researchers use geometric stochastic channel models (GSCMs), models which are so called since they consider the **geometry** of scatterers in the wireless environment and the **stochastic** nature of the channel. Simulations based on GSCMs are advantageous to researchers for a a few reasons:

(1) Simulations permit the generation a large quantities of data in a (relatively) short period of time.

(2) Simulations enable the adjustment of parameters of the communications system (e.g., carrier frequency, UE mobility, subcarrier spacing).

GSCMs consider the problem of modeling the wireless channels at two different scales. At a small-scale, the focus is on the modeling of "scatterers," objects in the environment which reflect electromagnetic waves in several random directions. Examples of scatterers in an outdoor environment are buildings, balconies, cars, and trees, while examples of scatters in an indoor environment are walls, furniture, and people. Due to the presence of many scatterers, electromagnetic waves transmitted from a BS arrive at UEs along multiple paths at different delays, and accordingly, the scatterers are often referred to as **multipath components (MPCs)**. From the UE's perspective, the MPCs are often grouped together in specific delay regions, and each such grouping of MPCs is referred to as a **cluster**. In GSCMs, modeling is done at a cluster-level, capturing the diverse effects of many MPCs that are present in typical real-world wireless channels. Modeled clusters are typically parameterized by three values: delay, direction of departure (DoD), and direction of arrival (DoA).

While the modeling of MPCs and clusters is vital, also vital is ensuring that the results of the GSCM-based simulations are statistically consistent with the measured channel data. This consistency is ensured by selecting **large-scale parameters (LSPs)**, examples of which

are the delay spread and the angular spread. In designing GSCMs, the goal is to balance the fidelity of cluster behavior (based on delay, DoD, DoA) and of LSPs when compared to measured channel data. In practice, researchers implement GSCMs based on two different modeling approaches: system-level modeling or cluster-level modeling.

In system-level modeling, the interaction(s) between the BS and the UE(s) is first defined by choosing LSPs. This selection of LSPs is done by sampling from probability distributions. Based on the LSPs, the clusters and MPCs for each BS/UE interaction is defined.

While some well-known channel models, such as the 3GPP Spatial Channel Model (SCM) [18] and WINNER II [19], adopt the system-level approach, this modeling approach has some salient limitations. First, system-level models do not lend themselves well to high-mobility scenarios, as the LSPs are likely to change as the position of UEs change substantially. Second, system-level models make the addition of new LSPs (e.g., inter-link correlation) to a given interaction.

In contrast with system-level models, cluster-level models begin with cluster/MPC definition then move on to LSPs. First, a large number of clusters are defined by sampling from a chosen probability distribution. Then, the location(s) of UE(s) is (are) defined, and the scattering of each cluster is calculated based on the "visibility" of UEs with respect to the clusters. Finally, the LSPs are synthesized by sampling from a given probability distribution.

The cluster-level approach addresses both of the issues associated with system-level approaches. Simulating time-varying channels with high mobility is straightforward since it involves recalculating UE visibility and LSPs, and layering new LSPs into the model is simple since it is the last step of the simulation.

For all CSI tests, we mainly rely on a cluster-level GSCM, the COST2100 channel model, [20]. We use two datasets with a single base station (gNB) and a single user equipment (UE) in the following scenarios:

(1) **Indoor** channels using a 5.3GHz downlink at 0.001 m/s UE velocity, served by a gNB at center of a $20\text{m}\times20\text{m}$ coverage area.

Table 1.2: Parameters used for COST2100 simulations for both Indoor and Outdoor datasets.

Symbol	Value	Description
N_b	32	Number of antennas at gNB
N_f	1024	Number of subcarriers for OFDM link
R_d	32	Number of delay elements kept after truncation
N	10^6	Total number of samples per dataset
T	10	Number of timeslots
δ	40 ms	Feedback delay interval between consecutive CSI timeslots

(2) **Outdoor** channels using a 300MHz downlink at 0.9 m/s UE velocity served by a gNB at center of a $400\text{m}\times400\text{m}$ coverage area.

In both scenarios, we use the parameters listed in Table 1.2.

1.4 Classical CSI Estimation

Works in compressive feedback for CSI estimation in MIMO networks can be placed in three broad categories. The first category includes works which use direct quantization of continuous CSI elements to discrete levels. The quantized CSI are encoded and fed back to the transmitter [21,22]. The second category includes works which use compressive sensing, a technique which applies a random measurement matrix at the transmitter and the receiver [23, 24]. Compressive sensing assumes matrices to be encoded and fed back meet certain sparsity requirements, and compressive sensing algorithms require iterative solvers [25] for decoding, resulting in undesired latency.

The last category of work in compressive CSI feedback uses deep learning (DL), neural networks with numerous layers which are trained on large datasets using backpropagation. We will provide the background knowledge needed for deep learning in Chapter 2, Section 2.1.

1.5 Objective and Contributions

Successful efforts in DL for CSI estimation have typically utilized convolutional neural networks (CNNs) in an autoencoder structure [1]. Variations on the CNN-based autoencoder have investigated different network architectures [26], variational training frameworks [27], and denoising modules [28]. These architectural changes are largely inspired by successful application of DL in image compression [29–31].

While they can continue to push the state-of-the-art in CSI reconstruction accuracy, architectural optimizations may ultimately follow the same trends of fields such as language modeling, where state-of-the-art performance requires prohibitively massive computational resources (e.g., [32]). In comparison to the massive resources used in such NLP tasks, the computational resources available in CSI estimation are much more limited, as these CSI estimation networks are (partially) deployed on UEs.

Rather than focus solely on architectural optimizations or large compute, this dissertation details our attempts to use domain knowledge to enhance the performance and the efficiency of neural networks for CSI estimation. Chapter 2 provides more background information on DL for CSI estimation as well as our work in power-based normalization, which leverages CSI sparsity to improve performance. Chapter 3 describes our work in differential encoding, which exploits temporal coherence of CSI to improve estimation performance while providing a less complex network than recurrent neural network-based CSI estimation networks. Chapter 4 describes our work in pilot-based delay domain CSI estimation, where we draw an explicit link between frequency-domain CSI estimation as defined in 3GPP specifications and the delay domain CSI used in many DL-based CSI estimation works. For a visual summary of these contributions and the respective areas of domain knowledge we leverage, see the Venn diagram in Figure 1.3)

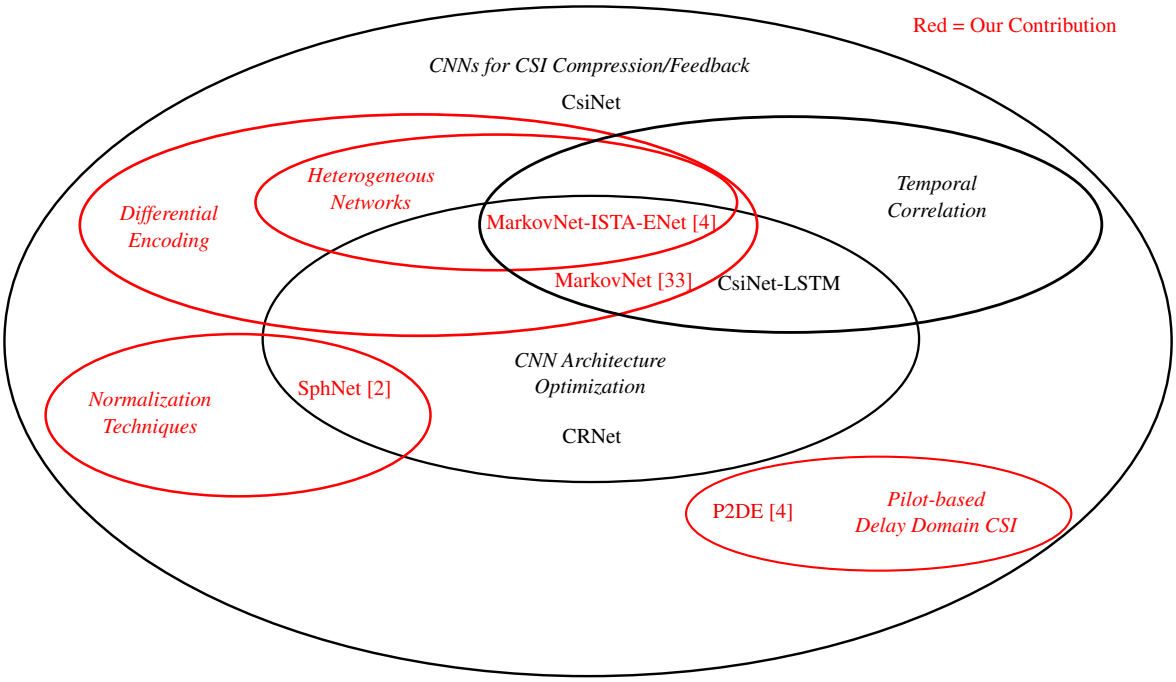


Figure 1.3: Venn diagram highlighting different aspects of domain knowledge in CNN-based CSI compressive feedback, relevant convolutional networks, and our contributions.

Chapter 2

Deep Learning for CSI Estimation

In this chapter, we will discuss important aspects of successfully applying deep learning to MIMO CSI estimation. In Section 2.1, we provide a basic overview of deep learning concepts that are pertinent to the task of CSI estimation. In Section 2.2, we discuss data pre-processing techniques and the applications of domain knowledge that have enabled deep learning-based CSI estimation. In Section 2.3, we describe our proposed CSI pre-processing technique, spherical normalization, which boosts estimation accuracy without significantly changing the computational complexity of a given estimation algorithm.

2.1 Deep Learning Background

This section provides a brief overview of relevant deep learning concepts employed in this work, including convolutional neural networks (CNNs), autoencoders, and unsupervised learning.

Deep learning (DL) is a subset of machine learning (ML), a broad class of algorithms which use data to "fit" models for prediction or classification tasks. The three predominant learning frameworks are supervised learning, unsupervised learning, and reinforcement learning. In the works proposed, we focus on *unsupervised learning*, which seeks to find a compressed representation of the data without labels (see Chapter 14 of [34] for an

overview).

Convolutional Neural Networks (CNNs): A neural network is a machine learning algorithm with multiple *layers* of parameterized linear functions followed nonlinear functions (typically referred to as 'activation' functions). The parameters for these layers can be updated via a stochastic optimizer (e.g., the Adam optimizer [35]), and given enough layers, such networks can achieve arbitrarily accurate functional approximation [36]. In recent years, neural networks with convolutional layers have established state-of-the-art performance in computer vision tasks such as image classification [37] and segmentation [38]. While the concept of a CNN has been around since at least the early 80's [39], CNNs did not see widespread adoption in computer vision until AlexNet [40]. This recent proliferation of CNNs in computer vision has been enabled by hardware advances such as graphical processing units (GPUs) and tensor processing units (TPUs) which make parallel training on large-scale datasets possible.

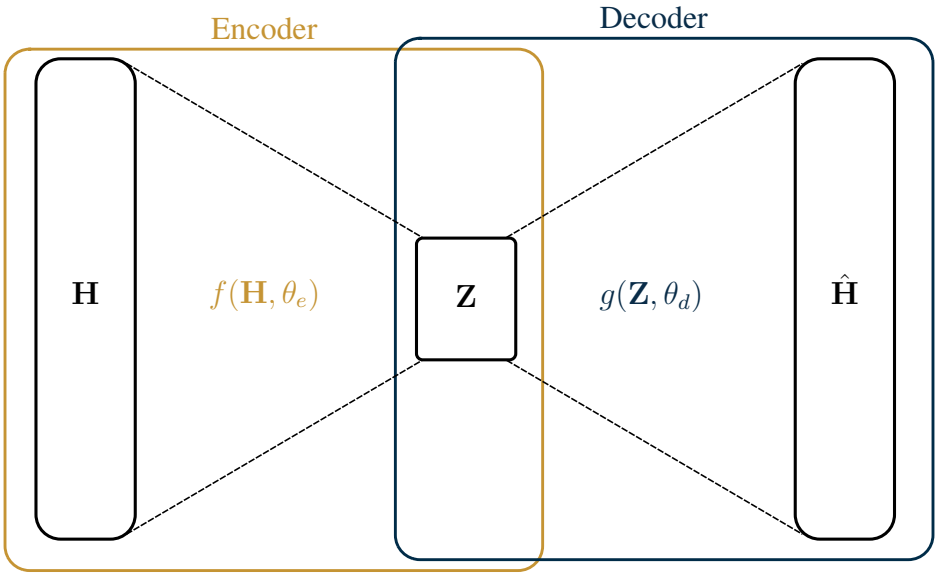


Figure 2.1: Abstract schematic for an autoencoder operating on CSI matrices $\mathbf{H}$. The encoder learns a latent representation, $\mathbf{Z}$, while the decoder learns to reconstruct estimates $\hat{\mathbf{H}}$.

A common architecture for deep unsupervised learning is the *autoencoder* (see Fig. 2.1 for a generic example). Trained end-to-end on input data, an autoencoder is comprised of an encoder and a decoder which jointly learn a compressed latent representation ($\mathbf{Z}$) and an

estimate of the input ($\hat{\mathbf{H}}$). By choosing $\mathbf{Z}$ to have lower dimension than the input, the network is forced to learn a "useful" summary of the input data. The typical objective function for such a network is the mean squared error (MSE),

$$\underset{\theta_e,\theta_d}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{H}_i - g(f(\mathbf{H}_i, \vec{\theta}_e), \vec{\theta}_d)\|^2.$$

We optimize network parameters $\vec{\theta}_e, \vec{\theta}_d$ by backpropagation and a stochastic optimization algorithm (e.g., stochastic gradient descent or Adam).

The first CNN autoencoder used for the CSI estimation task was CsiNet [1]. Figure 2.2 shows the architecture of CsiNet, where the encoder is deployed at the UE and the decoder is deployed at the BS. The encoder uses a convolutional layer and a linear layer to find a compressed representation of the input CSI, and the decoder uses a linear layer to expand the dimension of the feedback followed by two "refine blocks" and an output convolutional layer to refine the CSI estimate. After the CsiNet paper, many works attempted to bolster the performance of CNN autoencoders for CSI estimation by incorporating *domain knowledge*. This chapter details several such works, and in Section 2.2.1, we will discuss one such area of domain knowledge about CSI data: the sparse basis of the CSI data utilized in CsiNet.

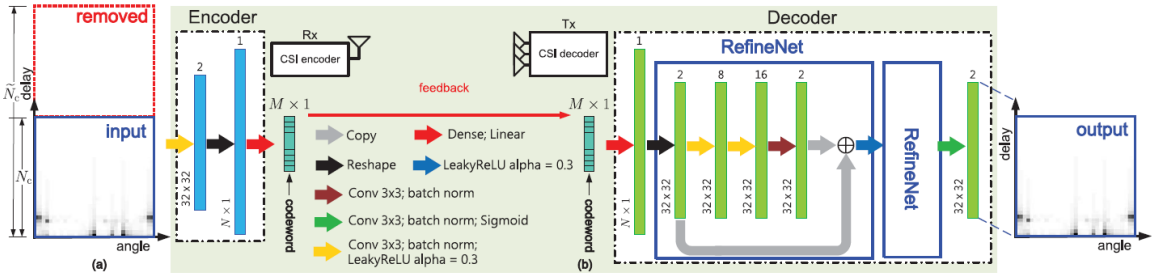


Figure 2.2: CsiNet architecture from [1]

Computational Complexity: Given the number of layers and operations involved in any deep learning algorithm, measuring the **computational complexity** of these algorithms is important. Two metrics that are commonly used in deep learning literature are:

(1) **Floating point operations (FLOPs)**: FLOPs provide a measure of the computations

used by a given layer or mathematical operation in a network. A single FLOP is defined as any arithmetic operation between floating point numbers (e.g., addition, multiplication) or any assignment of a floating point value.

(2) **Parameters**: The number of parameters in a deep learning network determines the storage cost for inference[1]. The number of parameters in a typical deep learning model can be anywhere from millions (e.g., ResNet architectures [41]) to billions (e.g., GPT-3 [32]).

See Appendix A for a more exhaustive discussion of common layers/operations used in deep networks and their corresponding computational complexity.

2.2 Data Pre-processing for CSI Data

The success of machine learning tasks relies on proper *data pre-processing*, a sequence of transformations used on the input data before fitting a model. In any machine learning task, data pre-processing is necessary to ensure that the scales of input features are similar. In deep learning for CSI estimation, three important pre-processing techniques are domain transformations, truncation, or normalization, and in this section, we will explore the important choices in pre-processing that authors have made based on domain knowledge of MIMO CSI data.

2.2.1 Sparse Basis for CSI

The first type of data pre-processing we consider is a domain transformation, the discrete Fourier transform in particular. While the **spatial-frequency** representation $\bar{\mathbf{H}}$ is used for beamforming at the transmitter, the number of non-zero elements is comparatively large. Given the dimension of $\bar{\mathbf{H}}$, feeding back entire CSI matrices is impractical. Instead, we

[1]The size of the dataset will contribute towards the storage cost but only during training/evaluation.

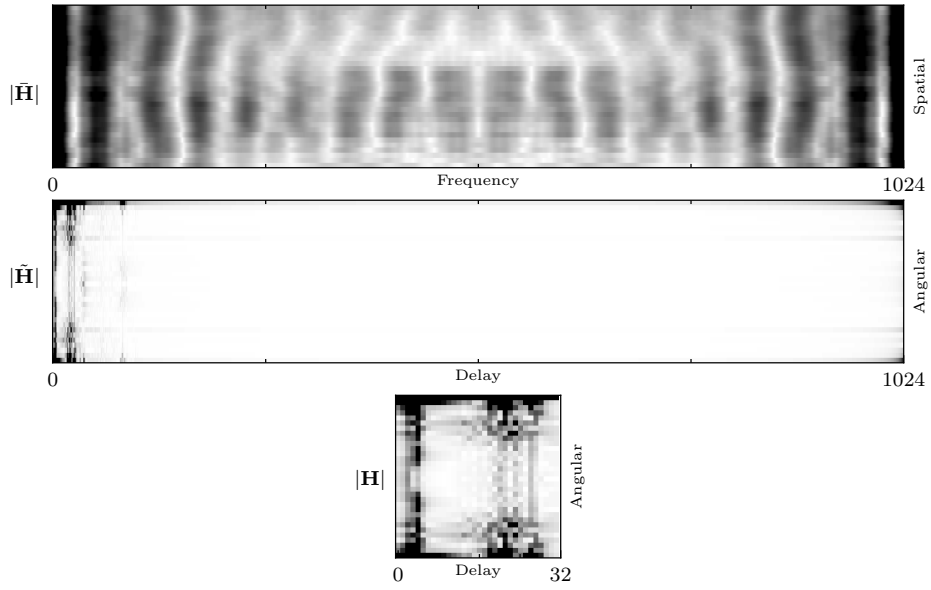


Figure 2.3: Magnitude of spatial-frequency ($\bar{\mathbf{H}}$), angular-delay ($\tilde{\mathbf{H}}$), and truncated angular-delay ($\mathbf{H}$) representations for a single random channel from the outdoor COST2100 dataset.

seek a compressed representation of a sparse transformation. The sparse representation we consider is the angular-delay representation of CSI matrices [42]. Denote the unitary inverse DFT for the spatial (frequency) axis as $\mathbf{F}_a \in \mathbb{C}^{N_b \times N_b}$ ($\mathbf{F}_d^H \in \mathbb{C}^{N_f \times N_f}$), and denote the spatial-frequency CSI matrix as $\bar{\mathbf{H}}$. The angular-delay domain representation $\tilde{\mathbf{H}}$ is given as

$$\tilde{\mathbf{H}} = \mathbf{F}_d^H \bar{\mathbf{H}} \mathbf{F}_a.$$

The delay spread of the resulting $\tilde{\mathbf{H}}$ can typically be captured with a small number of delay elements (see Figure 2.4), so we restrict our attention to the first R_d elements of $\tilde{\mathbf{H}}$, resulting in a truncated angular-delay matrix which we denote as $\mathbf{H} \in \mathbb{C}^{(R_d \times N_b)}$ for the downlink channel state. An illustrative example of this truncation can be seen at the bottom of Figure 2.3.

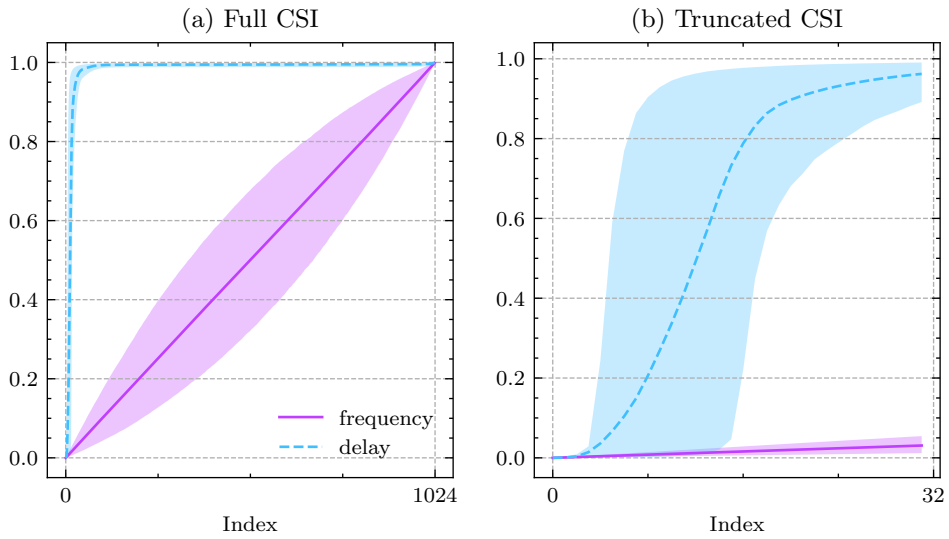


Figure 2.4: Energy CDF for 5000 CSI samples of 32 antennas and 1024 subcarriers, generated from COST2100 outdoor models described in Section 1.3. Mean percentage of energy in CSI matrix up to index is shown with 90% confidence intervals. The index denotes the amount of energy accounted for up to the corresponding frequency/delay element. The truncated angular-delay CSI contains a mean energy of 96.2% (c.i. 89.2%, 99.1%), while the truncated frequency-spatial CSI only contains a mean energy of 3.1% (c.i. 1.2%, 5.4%).

2.2.2 Bidirectional Reciprocity in FDD Networks

The next type of pre-processing under consideration is a change of coordinates. Specifically, rather than utilizing a Cartesian representation (i.e., real-imaginary channels), we can consider a polar representation (i.e., magnitude-phase). As discussed in Section 1.1, the reciprocity of downlink and uplink channels is weak in FDD wireless networks when compared to TDD. Despite this, DL CSI estimation techniques have used uplink CSI to improve the reconstruction accuracy of downlink CSI at gNB. In [43], the authors demonstrate that the correlation between the magnitude of uplink and downlink CSI elements is strong. To exploit magnitude reciprocity, they propose DualNet, a CNN autoencoder which learns a feedback encoding for the downlink CSI magnitude and decodes the feedback with the magnitude of uplink CSI as side information. The downlink phase is separately quantized and fed back to gNB via magnitude-dependent phase quantization (MDPQ). The authors demonstrate that exploiting bidirectional reciprocity can substantially improve CSI estimation accuracy.

2.2.3 Minmax Data Normalization

The last pre-processing technique we discuss is normalization. Typical deep autoencoders require normalized data to ensure that the range of the input data matches the range of the autoencoder's output function, which is typically chosen as `sigmoid` or `tanh` as pictured in Figure 2.5. To accommodate such output functions, most works in both image

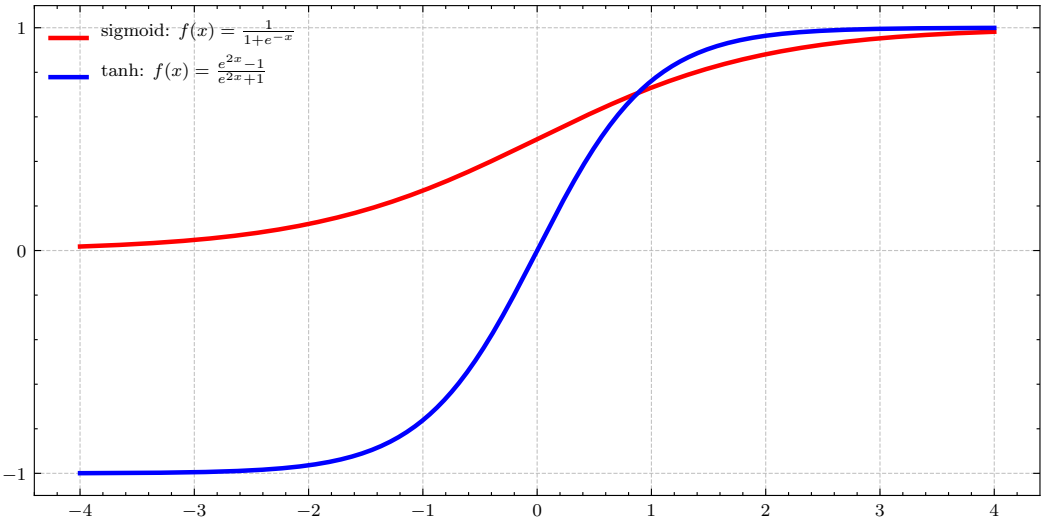


Figure 2.5: Typical activation functions used at the output of convolutional autoencoders.

compression and CSI estimation typically apply *minmax normalization*, where the extrema (i.e., the minimum and the maximum) of the real and imaginary channels are used to scale the entire dataset. For the scalar $H_n(i,j)$, the minmax-scaled version of this element is

$$H_{n,\text{minmax}}(i,j) = \frac{H_n(i,j) - H_{\text{min}}}{H_{\text{max}} - H_{\text{min}}} \in [0,1],$$

for $n \in [1,\ldots,N]$ given a dataset of N samples and i/j indexing the rows/columns of the CSI matrices. The resulting samples are cast to the range $[0,1]$.

For image data, minmax normalization results in each image's color channels scaled to the range $[0,1]$. The resulting distribution for each color channel is typically satisfactory for image tasks, as the variance is not much smaller than the range of the normalized data (see Fig. 2.6).

However, for CSI matrices, minmax normalization is applied to the real and imaginary channels of each element. For typical channel models and parameters, the distribution of channel elements tends to have much lower variance than that of image data (see Fig. 2.7). This smaller variance can be explained by the difference in the datasets' ranges – while the channels in image data (e.g., ImageNet) assume integer values between $[0, 255]$, the channels in CSI data (e.g., COST2100) assume floating point values smaller than 10^{-3}.

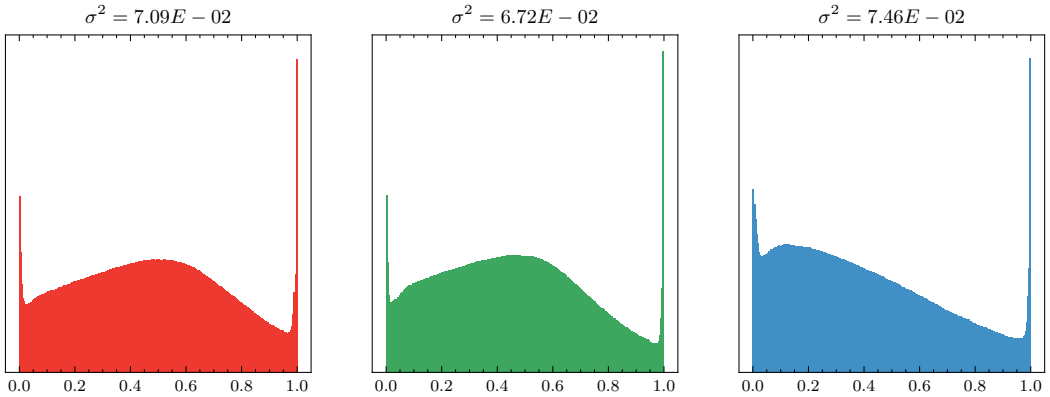


Figure 2.6: Distribution and variance of minmax-normalized ImageNet color channels ($N = 50000$) images.

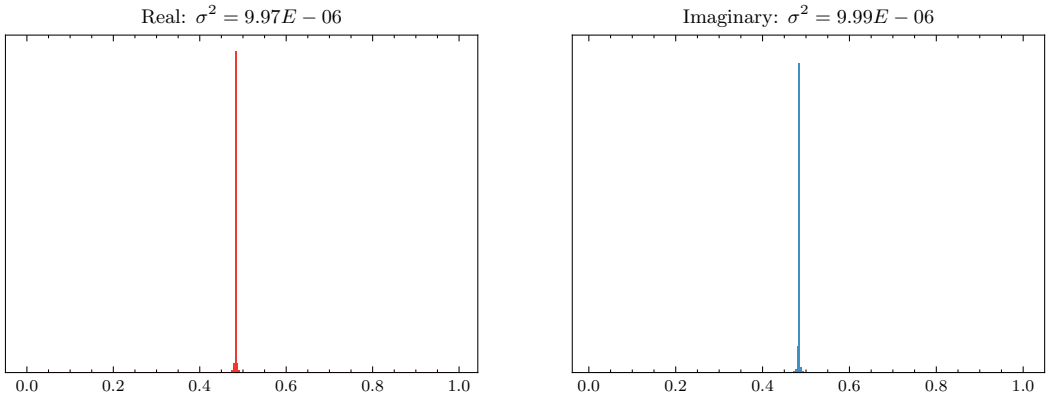


Figure 2.7: Distribution and variance of minmax-normalized COST2100 real/imaginary channels ($N = 99000$) images.

In Section 2.3, we will detail our proposed method for increasing the variance of CSI data and demonstrate that this method can improve the estimation accuracy of deep learning networks for CSI estimation.

2.2.4 Related Works

In image processing, several works have investigated normalization techniques such as batch normalization [44], instance normalization [45], layer normalization [46], and group normalization [47]. These normalization techniques scale the outputs of latent layers in neural networks, which helps to solve the problem of covariate shift [44] where the mean and variance changes between subsequent layers of the network.

Other works have studied normalization of the network's inputs. A number of works have investigated adaptive normalization techniques for time series estimation tasks [48–50]. In [51], the authors proposed a trainable input network which learns to shift, scale, and filter the unnormalized data while training the target network for a time series prediction task.

2.3 Spherical Normalization

Here, we discuss our work in spherical normalization (Section 2.3) and our optimized network architecture, CsiNet-Pro (Section 2.3.1) [2].

Rather than apply minmax normalization, which results in a low variance distribution when applied to highly sparse CSI data, we propose spherical normalization. Before describing spherical normalization in detail, consider z-score normalization. Given a random variable, x, with mean μ and standard deviation σ. The z-score normalized version of this random variable is given as

$$z = \frac{x - \mu}{\sigma}. \tag{2.1}$$

Assuming x is normally distributed, the resulting random variable, z, is a standard normal distribution such that $z \sim \mathcal{N}(0, 1)$. Inspired by z-score normalization, we seek a normalization scheme which adjusts the range of each channel sample. Under spherical normalization, each sample in the dataset is scaled by its power. Denote the n-th downlink CSI matrix of

the dataset as $\mathbf{H}_d^n$. The spherically normalized version of the downlink CSI is given as

$$\check{\mathbf{H}}_d^n = \frac{\mathbf{H}_d^n}{\|\mathbf{H}_d^n\|}. \tag{2.2}$$

Observe that (2.2) is similar to (2.1) without the mean shift in the numerator[2] and with the power term of each CSI sample rather than the variance of the entire distribution. After applying (2.2) to each sample, minmax scaling is applied to the entire dataset. The resulting dataset under spherical normalization can exhibit a larger variance than the same dataset under minmax scaling (compare Fig. 2.8 with Fig. 2.7).

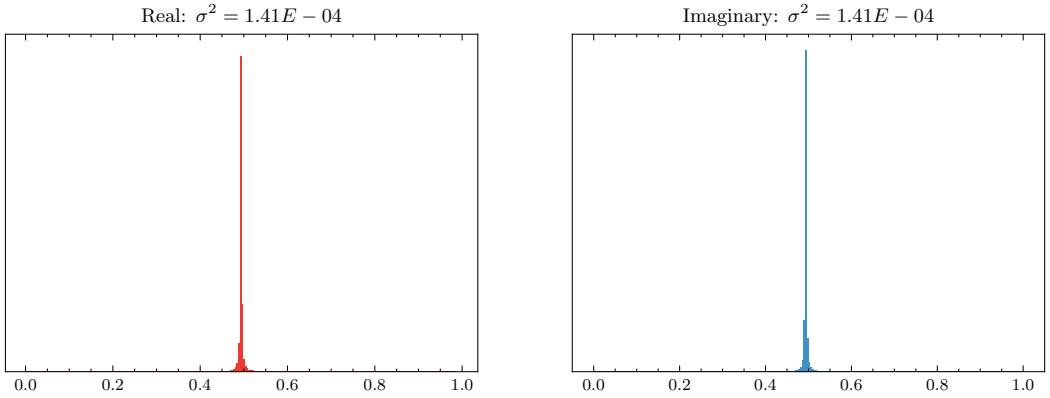


Figure 2.8: Distribution and variance of COST2100 real/imaginary channels under spherical normalization ($N = 99000$) images.

Beyond desirable properties in the input distribution, spherical normalization also results in an objective function which is better matched with the evaluation criterion. Neural networks for CSI estimation are optimized using the mean-squared error loss,

$$\text{MSE} = \frac{1}{N}\sum_{k=1}^{N} \|\mathbf{H}_k - \hat{\mathbf{H}}_k\|^2, \tag{2.3}$$

while channel state reconstruction accuracy is measured in terms of normalized mean-squared

[2]Since the mean of COST2100 data is $\approx 10^{-10}$, we can safely ignore this mean shift in spherical normalization.

error,

$$\text{NMSE} = \frac{1}{N}\sum_{k=1}^{N}\frac{\|\mathbf{H}_k - \hat{\mathbf{H}}_k\|^2}{\|\mathbf{H}_k\|^2}. \tag{2.4}$$

Observe that when the $\mathbf{H}_k$ $(\hat{\mathbf{H}}_k)$ in (2.3) is replaced with $\check{\mathbf{H}}_k$ $(\hat{\check{\mathbf{H}}}_k)$, we have

$$\begin{aligned}\frac{1}{N}\sum_{k=1}^{N}\|\check{\mathbf{H}}_k - \hat{\check{\mathbf{H}}}_k\|^2 &= \frac{1}{N}\sum_{k=1}^{N}\left\|\frac{\mathbf{H}_k}{\|\mathbf{H}_k\|^2} - \frac{\hat{\mathbf{H}}_k}{\|\mathbf{H}_k\|^2}\right\|^2 \\ &= \frac{1}{N}\sum_{k=1}^{N}\frac{\|\mathbf{H}_k - \hat{\mathbf{H}}_k\|^2}{\|\mathbf{H}_k\|^2},\end{aligned}$$

which is equivalent to (2.4). Thus, a neural network optimized with MSE as the loss function and trained using spherically normalized data is in fact being optimized with respect to NMSE of the original data.

2.3.1 CsiNet-Pro

In [2], we proposed a network with larger convolutional kernels and no residual connections called CsiNet-Pro. Large kernels (e.g., (7×7) in CsiNet-Pro) allow the network to capture features corresponding to larger delay spreads than comparatively small kernels (e.g., (3×3) in CsiNet [1]). In addition to the compressed feedback of the autoencoder, the encoder must feedback the power of the CSI matrix, $\|\mathbf{H}\|$, meaning the number of floating point elements to feed back increases from r to $r + 1$. This can be seen in Figure 2.9, which shows the CsiNet-Pro architecture using spherical normalization, which we refer to as 'SphNet.'

2.3.2 Results

Training on spherically normalized data and optimizing with respect to NMSE can yield better accuracy. Fig. 2.10 demonstrates this improvement for CsiNet and CsiNet-Pro on the

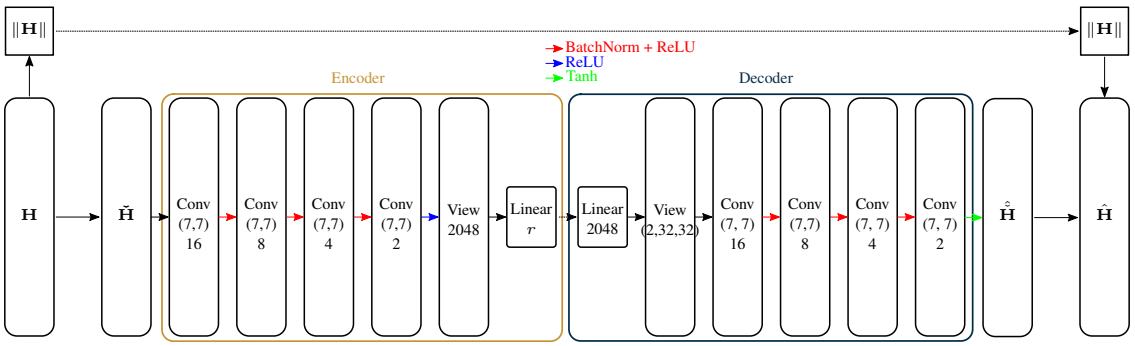
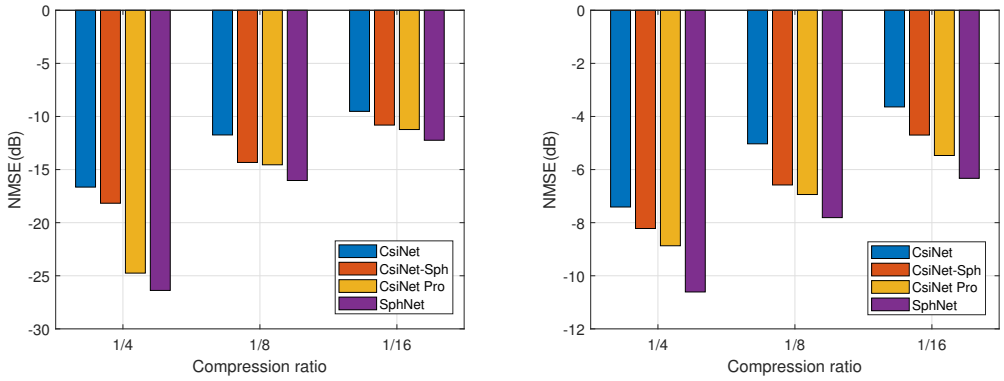


Figure 2.9: SphNet – CsiNet-Pro architecture with Spherical Normalization.

COST2100 dataset. CsiNet and CsiNet-Pro are trained with minmax normalization while CsiNet-Sph and SphNet are trained with spherical normalization.



(a) Indoor (b) Outdoor

Figure 2.10: Reconstruction error for CsiNet [1] and CsiNet-Pro with and without spherical normalization. SphNet combines CsiNet-Pro with spherical normalization [2].

This chapter has demonstrated the importance of considering unique features of CSI data when applying deep learning to compressive CSI estimation. Making such considerations allowed us to improve estimation accuracy without altering the network's architecture. In the following chapter, we will again consider a unique characteristic of CSI data, temporal correlation of consecutive CSI samples, and we will exploit this correlation to reduce the computational complexity of CSI estimation networks.

Chapter 3

Temporal Coherence and Differential Encoding

In this chapter, we consider methods for exploiting temporal correlation between CSI of subsequent timeslots. The *coherence time* of a channel is the amount of time that a channel estimate can be used before that estimate's SNR falls beneath a given threshold [52]. Within this window of time ($\Delta t = t_i - t_{i-1}$), the correlation between CSI matrices $\mathbf{H}_i$ and $\mathbf{H}_{i-1}$ is high (see Figure 3.1 for an illustrative example).

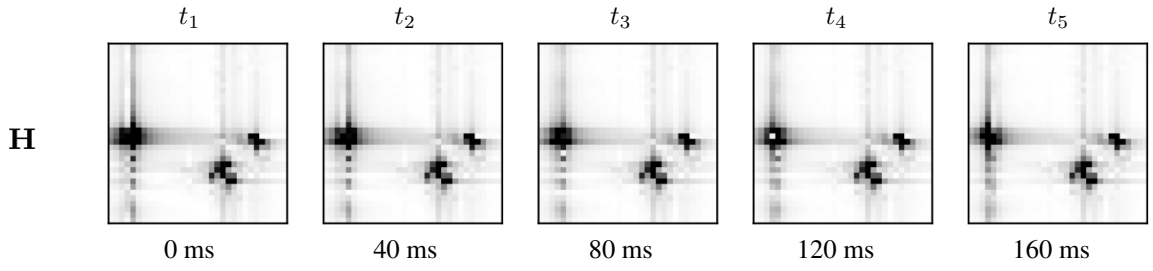


Figure 3.1: Ground truth CSI ($\mathbf{H}$) for five timeslots (t_1 through t_5) on one sample from the validation set of the outdoor dataset.

Assuming the channel exhibits temporal coherence within a certain window of time, a reasonably accurate CSI estimate at time t_{i-1} can be used to estimate the CSI at time t_i.

Generically, we can write this estimator as

$$\grave{\mathbf{H}}_i = h(\hat{\mathbf{H}}_{i-1}) \tag{3.1}$$

where $\mathbf{H}_i$ is the CSI matrix at time t_i and $\hat{\mathbf{H}}_i$ is its estimator. The estimation error under $\grave{\mathbf{H}}_i$ is

$$\mathbf{E}_i = \mathbf{H}_i - \grave{\mathbf{H}}_i. \tag{3.2}$$

Before elaborating on how to use this error term for CSI estimation, we first highlight relevant works in deep learning which exploit temporal correlation.

3.1 Recurrent Neural Networks

Prior work in temporal correlation for CSI estimation utilized state-space methods such as the Kalman filter [53–55]. Since it relies on explicit state space and noise models, the Kalman filter's predictive power in CSI estimation is limited. Furthermore, such work generally does not propose a method for feedback compression, making comparison with the following ML methods difficult.

Recent works have leveraged recurrent neural networks (RNNs) to exploit temporal correlation for CSI estimation [3, 56–59]. RNNs include recurrent layers, such as the long short-term memory (LSTM) cell or the gated recurrent unit (GRU), which are capable of learning long-term dependencies of a given process through backpropagation [60] and can be used to predict future states of the process [61].

RNNs have been used extensively in natural language processing (NLP) for machine translation [62] and sentiment extraction [63]. For such works in NLP, authors have empirically found "stacked" or "deep" RNNs to be effective (e.g., Fig. 3.2), hypothesizing that having multiple recurrent layers allows the network to extract different semantic timescales

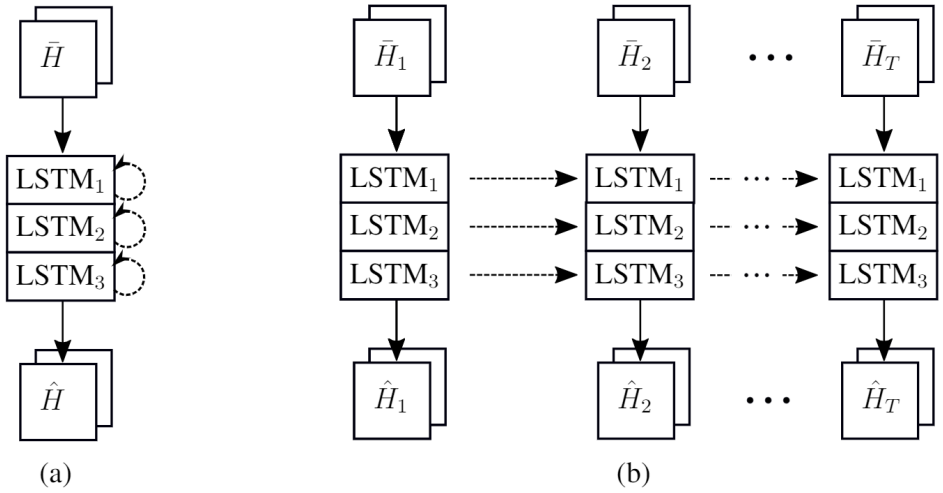


Figure 3.2: An example of LSTMs used for CSI estimation. (a) "Stacked" LSTM network of depth 3 shown with recurrent connections. (b) Same LSTM network "unrolled" into T timeslots

[63, 64]. Works in CSI estimation have taken cues from this work in NLP, proposing CSI estimation networks with stacked LSTMs after a sequence of autoencoders [3]. While such work has demonstrated the utility of RNNs, the computational cost of LSTMs can be prohibitively high. For example, the RNN portion of the network proposed in [3] accounts for 10^8 additional parameters (see Figure 3.3 for the network architecture used in [3]). Since channel estimation should not place an undue computational burden on the communications system, LSTMs can be problematic.

3.2 Differential Encoding

Rather than use RNNs to extract temporal dependencies in CSI data, we proposed a lightweight network based on the principle of differential encoding. We trained a network to estimate the error (3.2) under a linear estimator,

$$\grave{\mathbf{H}}_i = \hat{\mathbf{H}}_{i-1}\mathbf{W}$$

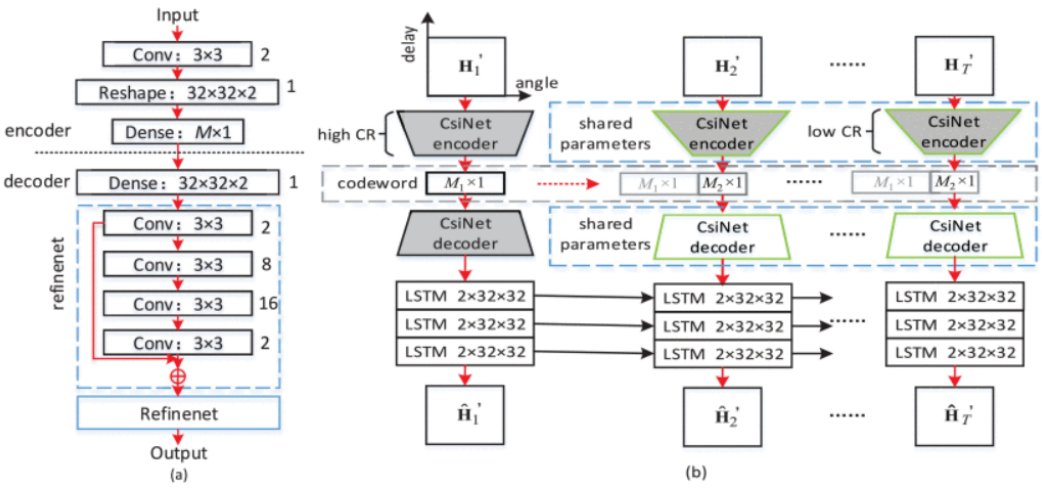


Figure 3.3: CsiNet-LSTM architecture from [3]. (a) CsiNet architecture used in each timeslot. (b) Full CsiNet-LSTM architecture using shared codewords and stacked LSTM cells after CsiNets to extract temporal correlation between timeslots.

where $\mathbf{W} \in \mathbb{C}^{R_b \times R_b}$ is the minimum mean squared error (MMSE) estimator,

$$\mathbf{H}_i = \mathbf{H}_{i-1}\mathbf{W} + \mathbf{E}_i$$

$$\mathbf{H}_{i-1}^H\mathbf{H}_i = \mathbf{H}_{i-1}^H\mathbf{H}_{i-1}\mathbf{W} + \cancelto{\mathbf{0}}{\mathbf{H}_{i-1}^H\mathbf{E}_i},$$

where the cancellation of the product $\mathbf{H}_{i-1}^H\mathbf{E}_i$ is due to the principle of orthogonality (i.e., the error terms are orthogonal to the observed data). Denoting the cross correlation matrix as $\mathbf{R}_i = \mathbb{E}\left[\mathbf{H}_{t-i}^H\mathbf{H}_t\right]$, we solve for the MMSE estimator,

$$\mathbf{W} = \mathbf{R}_0^{-1}\mathbf{R}_1.$$

In practice, the population correlation matrices are estimated via finite samples of size N,

$$\hat{\mathbf{R}}_k = \frac{1}{N_{\text{train}}}\sum_{j}^{N_{\text{train}}} \mathbf{H}_{i-k}^H(j)\mathbf{H}_i(j),$$

where $\mathbf{H}_i(j)$ is the j-th sample in the training set. The MMSE estimator based on the sample correlation matrices is written as

$$\hat{\mathbf{W}} = \hat{\mathbf{R}}_0^{-1}\hat{\mathbf{R}}_1.$$

In Appendix B, we describe the general multivariate autoregressive model which depends on $\hat{\mathbf{W}}$[1]. However, we can simplify this model to use a scalar coefficient, $\hat{\gamma} \in \mathbb{R}$, as

$$\hat{\gamma} = \frac{\text{Trace}(\hat{\mathbf{R}}_1)}{\sum_k^{R_d}\sum_l^{N_b}\hat{\mathbf{R}}_0(k,l)}, \tag{3.3}$$

where k (l) are the row (column) indices of the correlation matrices. The estimator in this case is

$$\grave{\mathbf{H}}_i = \hat{\gamma}\hat{\mathbf{H}}_{i-1}. \tag{3.4}$$

Under the estimator γ, we proposed to encode the error, $\mathbf{E}_t$, using a convolutional autoencoder, $f(\mathbf{E}_t)$,

$$\hat{\mathbf{E}}_i = g(f(\mathbf{E}_i, \vec{\theta}_e), \vec{\theta}_d),$$

where $\mathbf{E}_i = \mathbf{H}_i - \hat{\gamma}\hat{\mathbf{H}}_{i-1}$. The base station has access to the estimators $\hat{\gamma}$ and $\hat{\mathbf{H}}_{i-1}$, and the resulting CSI estimate at t_i is

$$\hat{\mathbf{H}}_i = \hat{\gamma}\hat{\mathbf{H}}_{i-1} + \hat{\mathbf{E}}_i \tag{3.5}$$

[1]Furthermore, Appendix B discusses this multivariate model for the p-step case, i.e. using p previous timeslots rather than a single timeslot as described in this section. However, based on experimental results, models with more than one previous timeslot provided only marginal improvements over the one-step model.

3.2.1 MarkovNet

In [33], we proposed MarkovNet, a deep differential autoencoder. Each timeslot of MarkovNet uses an instance of CsiNet-Pro with unique parameters. The network at the first timeslot (t_1) is trained directly on the CSI (i.e., $\mathbf{H}_1$). For all subsequent timeslots, t_i for $i \geq 2$, we use the MMSE estimator (3.4) to produce an error term $\mathbf{E}_t$, and the autoencoder in each timeslot is trained to produce an error estimate, $\hat{\mathbf{E}}_t$. The estimated error is added back per (3.5) to produce a refined estimate.

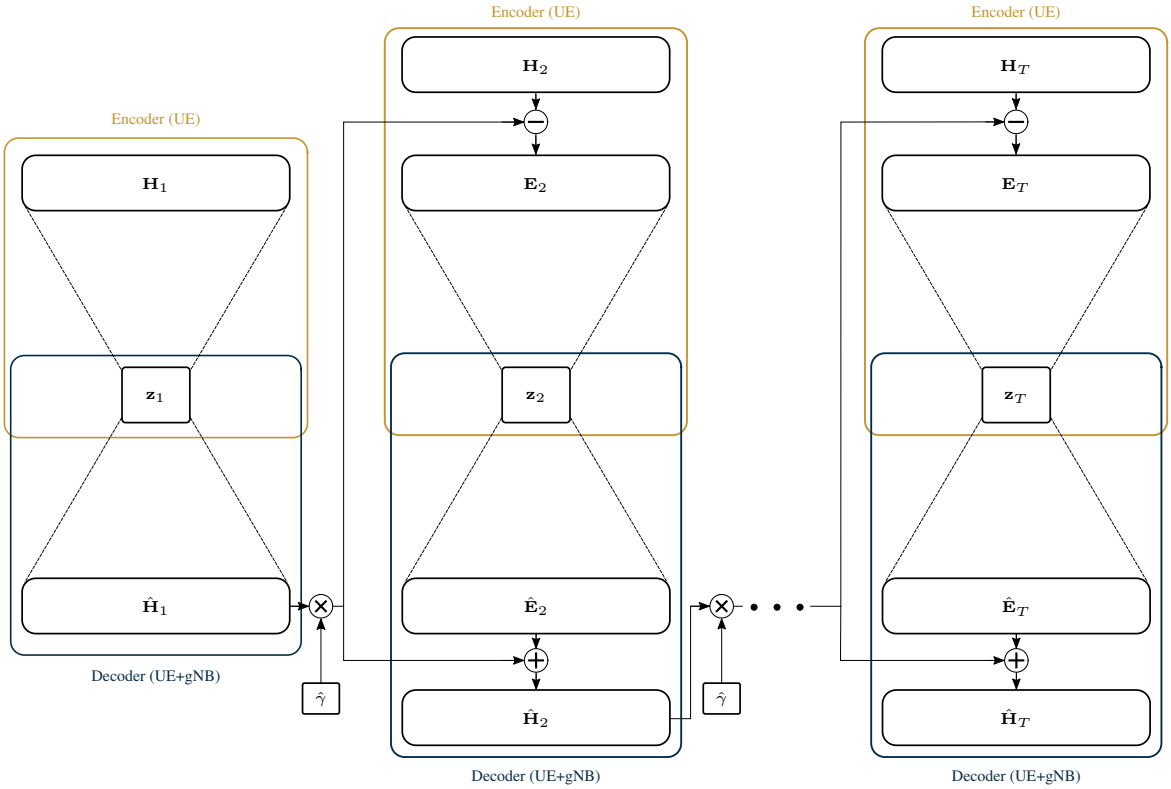


Figure 3.4: Abstract architecture for MarkovNet. Networks at t_i for $i \geq 2$ are trained to predict the estimation error, $\mathbf{E}_i$.

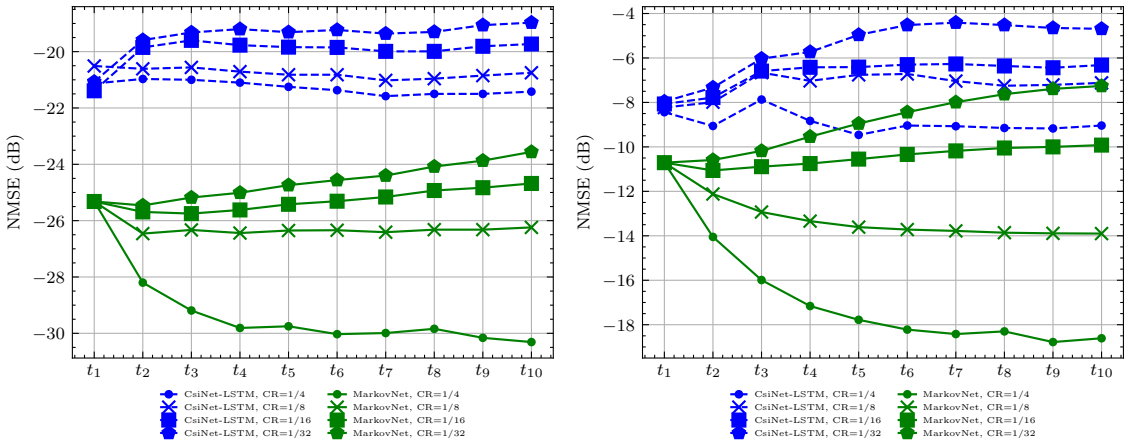
3.3 Results

We compare MarkovNet with CsiNet-LSTM [3] on the indoor and outdoor COST2100 datasets (for details, see Section 1.3). For MarkovNet, we train the network at the first

timeslot for 1000 epochs. In each subsequent timeslot, we initialize the network using the weights from the previous timeslot and train for 200 epochs. We use a batch size of 200. We perform a training/testing split of 75k/25k samples, and we estimate $\hat{\gamma}$ using the training set. To compare the estimation accuracy of each network, we report the NMSE.

3.3.1 Network Comparison

Figure 3.5 shows the NMSE of MarkovNet and CsiNet-LSTM for four different compression ratios. For the indoor network, all instances of MarkovNet achieve lower NMSE than all instances of CsiNet-LSTM. In the outdoor scenario, each CR for MarkovNet demonstrates lower NMSE than the corresponding CR for CsiNet-LSTM. Between both channel scenarios, MarkovNet shows gradual improvement for subsequent timeslots if the CR is high enough while CsiNet-LSTM only improves gradually in the outdoor environment for CR$= \frac{1}{4}$. Figure 3.6 shows a random sample from the test set, $\mathbf{H}$, and the estimates produced



(a) Indoor (b) Outdoor

Figure 3.5: NMSE comparison of MarkovNet and CsiNet-LSTM at various compression ratios (CR).

by CsiNet-LSTM and MarkovNet for a CR of $\frac{1}{4}$. This sample contains three "peak" magnitude regions. While both networks manage to capture the two larger samples, MarkovNet is able to recover the small peak magnitude region (green arrow) which CsiNet-LSTM fails to produce (red arrow).

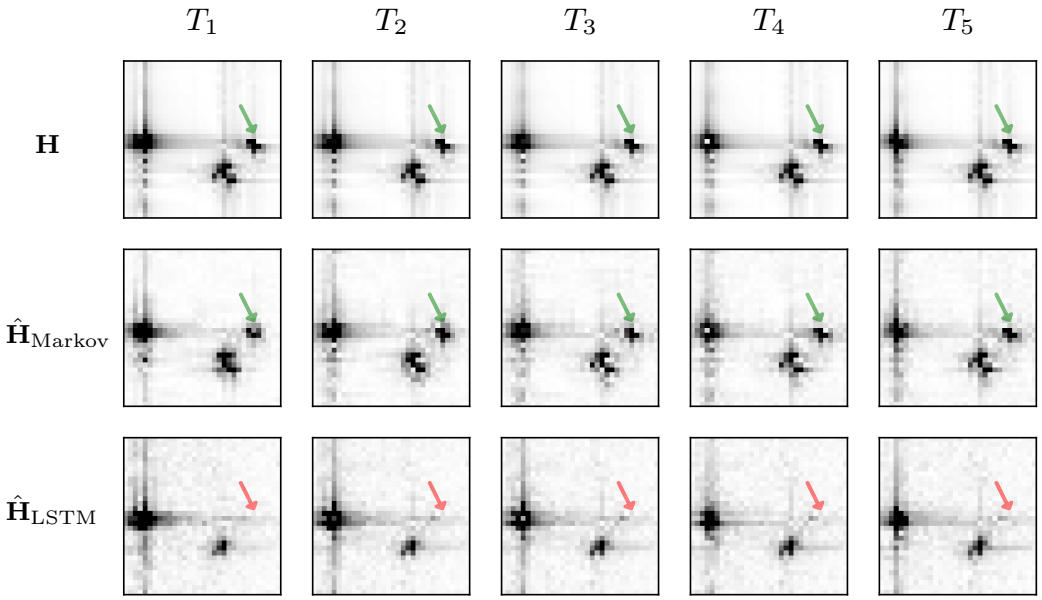


Figure 3.6: CSI ($\mathbf{H}$), MarkovNet estimates ($\hat{\mathbf{H}}_{\text{Markov}}$), and CsiNet-LSTM estimates ($\hat{\mathbf{H}}_{\text{LSTM}}$) across five timeslots (T_1 through T_5) on one outdoor channel sample from the test set, using CR $= \frac{1}{4}$.

3.3.2 Performance under Quantization

To understand the effect of quantization on the network performance, we choose to use μ-law companding and uniform quantization on the latent feedback elements. We begin by performing a logarithmic scaling on the feedback elements, x,

$$f(x) = \frac{\text{sign}(x)\ln(1+\mu|x|)}{\ln(1+\mu)},\ 0 \leq |x| \leq 1. \tag{3.6}$$

After applying (3.6) to the signal, uniform quantization is applied to yield

$$\hat{x} = \Delta \left\lfloor \frac{f(x)}{\Delta} \right\rceil \tag{3.7}$$

where $\Delta = 2^{-(b-1)}$ for b-bit quantization. Finally, an inverse logarithmic scaling is applied to quantized signal,

$$F(\hat{x}) = \frac{\text{sign}(\hat{x})\,(1+\mu)^{|\hat{x}|} - 1}{\mu},\ -1 \leq \hat{x} \leq 1. \tag{3.8}$$

In short, the described μ-law quantization scheme involves applying (3.6), then (3.7), then (3.8) to each feedback element. Figures 3.7 and 3.8 show the performance of MarkovNet and CsiNet-LSTM under μ-law quantization where $\mu = 255$.

In the Outdoor scenario, the performance of each network at each compression ratio does not change substantially for different numbers of quantization bits. In contrast, the performance for each network/compression ratio in the Indoor scenario drops appreciably for smaller quantization bits. This is possibly because the non-quantized performance of the Indoor networks is much better (i.e., -20dB to -17dB) than the performance of the Outdoor networks (i.e., -12dB to -5dB), and so a small change in accuracy for the Indoor network is more noticeable than a small change for the Outdoor network. We note that the performance of either network could potentially benefit from quantization during training, as all the results in Figures 3.5a and 3.5b are trained with continuous feedback.

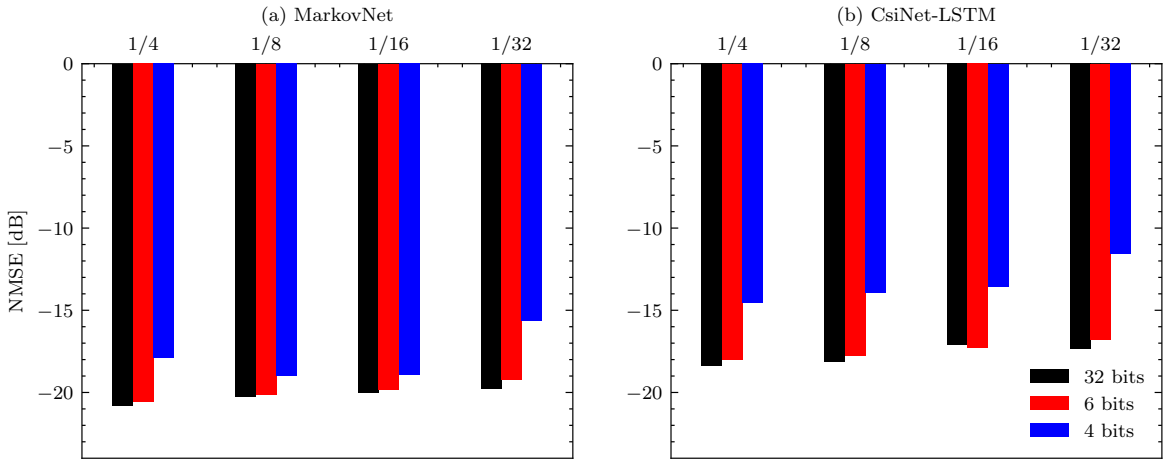


Figure 3.7: NMSE comparison of MarkovNet and CsiNet-LSTM for the Indoor scenario with feedback subject to μ-law quantization using fixed step size, $\Delta = 2^{-(b-1)}$, for b bits.

3.3.3 Computational Complexity

The resulting network requires no recurrent layers, resulting in a substantial reduction in computational complexity. Table 3.1 shows the number of parameters and FLOPs per timeslot for CsiNet-LSTM, MarkovNet, and CsiNet. The parameter count of MarkovNet

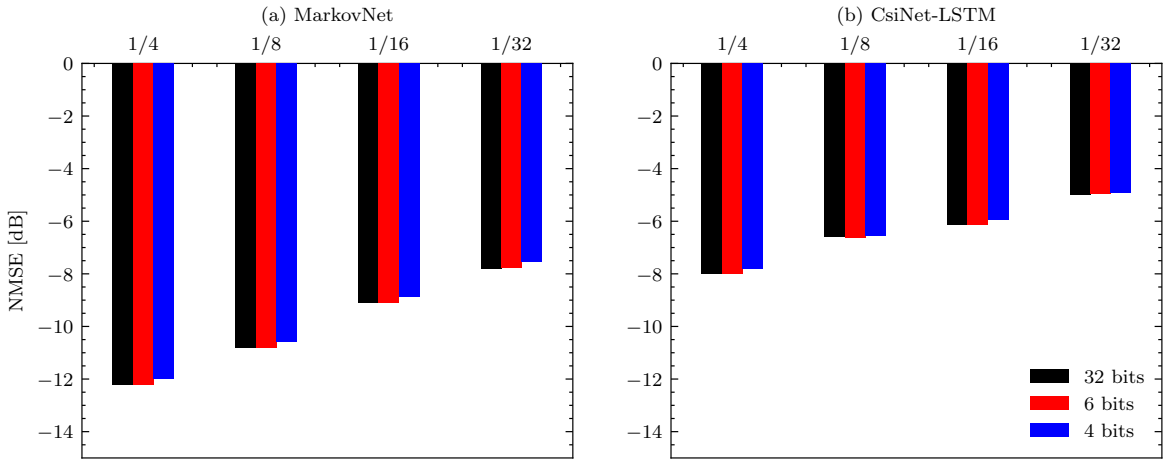


Figure 3.8: NMSE comparison of MarkovNet and CsiNet-LSTM for the Outdoor scenario with feedback subject to μ-law quantization using fixed step size, $\Delta = 2^{-(b-1)}$, for b bits.

is on par with CsiNet, and CsiNet-LSTM requires orders of magnitude more parameters. While the number of FLOPs for MarkovNet is nearly 10 times smaller than CsiNet-LSTM, MarkovNet requires 5 to 10 times more FLOPs than CsiNet due to the increased kernel size of CsiNet-Pro.

Table 3.1: Model size/computational complexity of tested temporal networks (CsiNet-LSTM, MarkovNet) and comparable non-temporal network (CsiNet). M: million.

	Parameters			**FLOPs**		
	CsiNet-LSTM	**MarkovNet**	**CsiNet**	**CsiNet-LSTM**	**MarkovNet**	**CsiNet**
CR=1/4	132.7 M	2.1 M	2.1 M	412.9 M	44.5 M	7.8 M
CR=1/8	123.2 M	1.1 M	1.1 M	410.8 M	42. 4 M	5.7 M
CR=1/16	118.5 M	0.5 M	0.5 M	409.8 M	41.3 M	4.7 M
CR=1/32	116.1 M	0.3 M	0.3 M	409.2 M	40.8 M	4.1 M
CR=1/64	115.0 M	0.1 M	0.1 M	409.0 M	40.5 M	3.9 M

Chapter 4

Bandwidth Efficient Pilot-based CSI Feedback

This chapter details an estimator for the UE-side angular-delay domain CSI based on a limited number of spatial-frequency domain pilots. This scheme adheres to the 3GPP standards for pilot allocation across time-frequency resources as described in Section 1.2.

Section 4.1 details our proposed pilots-to-delay estimator (P2DE), and Figure 4.1 demonstrates the operating principle behind P2DE. Appendix C describes off-diagonal regularization as a countermeasure for ill-conditioned CSI matrices. Section 4.1.1 explicitly links the proposed P2DE to the 3GPP placement of CSI-RS/DMRS resource elements and describes our proposed diagonal pilot pattern which adheres to LTE/NR specifications. Section 4.2 describes an extension of our previously proposed differential encoding network which uses heterogeneous CNNs for different timeslots. Finally, Section 4.3 presents results for the P2DE and our propose heterogeneous differential encoding networks.

4.1 Pilots-to-delay Estimator (P2DE)

Denote $\boldsymbol{\eta}_i \in \mathbb{C}^{N_f}$ as the i-th row of the spatial-frequency matrix $\mathbf{H}$, and denote the downsampled version of $\boldsymbol{\eta}_i$ as $\boldsymbol{\eta}_{d,i} \in \mathbb{C}^{M_f}$ where $M_f << N_f$. Thus, the spatial-frequency

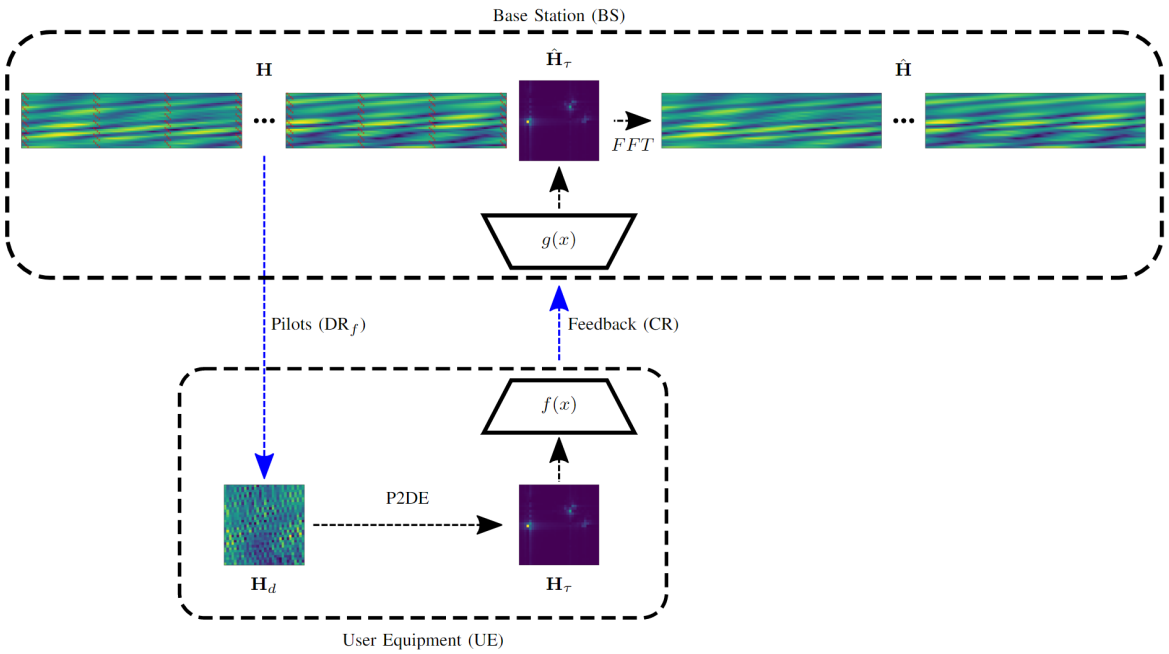


Figure 4.1: Compressive CSI estimation based on linear P2D estimator. First, we use downlink pilots to generate a sparse, frequency domain CSI estimate of size $M_f << N_f$. We then apply the P2D estimator, $\mathbf{Q}_{N_t}^{\dagger}$ of (4.5), to establish the truncated delay domain CSI estimate. We train a learnable encoder, $f(x)$, and decoder, $g(x)$, to compress and decode the feedback, respectively. The gNB recovers the frequency domain CSI from the decoded delay domain CSI estimate.

CSI, $\mathbf{H}$, and its downsampled counterpart, $\mathbf{H}_d$, can be written as,

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \\ \vdots \\ \boldsymbol{\eta}_{N_b} \end{bmatrix} \in \mathbb{C}^{N_b \times N_f}, \ \mathbf{H}_d = \begin{bmatrix} \boldsymbol{\eta}_{d,1} \\ \boldsymbol{\eta}_{d,2} \\ \vdots \\ \boldsymbol{\eta}_{d,N_b} \end{bmatrix} \in \mathbb{C}^{N_b \times M_f}. \tag{4.1}$$

$\boldsymbol{\eta}_{d,i}$ is related to $\boldsymbol{\eta}_i$ by the downsampling matrix for the i-th antenna port, $\mathbf{P}_i$, as

$$\boldsymbol{\eta}_{d,i} = \boldsymbol{\eta}_i \mathbf{P}_i \ \forall \ i \in [1, \ldots, N_b]. \tag{4.2}$$

Denote the delay-domain CSI vector, $\tilde{\boldsymbol{\eta}}_i$, which is defined as

$$\tilde{\boldsymbol{\eta}}_i \mathbf{F} = \boldsymbol{\eta}_i, \tag{4.3}$$

where $\mathbf{F}$ is the $\mathbf{C}^{N_f \times N_f}$ discrete Fourier transform (DFT) matrix. To relate the frequency domain pilots to the delay domain, we apply the pilot downsampling matrix $\mathbf{P}_i$ to both sides of (4.3),

$$\begin{aligned} \tilde{\boldsymbol{\eta}}_i \mathbf{F} \mathbf{P}_i &= \boldsymbol{\eta}_i \mathbf{P}_i \\ \tilde{\boldsymbol{\eta}}_i \mathbf{Q}_i &= \boldsymbol{\eta}_{d,i} \end{aligned} \tag{4.4}$$

where $\mathbf{Q}_i = \mathbf{F}\mathbf{P}_i \in \mathbb{C}^{N_f \times M_f}$ is the downsampled DFT matrix. Leveraging the sparsity of CSI data in the delay domain (see Section 2.2.1, Figure 2.3), many works choose to feedback and compress the truncated delay domain vectors, $\tilde{\boldsymbol{\eta}}_{c,i} \in \mathbb{C}^{N_t}$. The zero-padded vector $\tilde{\boldsymbol{\eta}}_i$ defined as

$$\tilde{\boldsymbol{\eta}}_i = \left[\tilde{\boldsymbol{\eta}}_{c,i}, \mathbf{0}_{N_f - N_t}\right]. \tag{4.5}$$

Based on 4.4, the delay domain can be related directly to the pilots by taking the pseudoinverse,

$$\begin{aligned} \tilde{\boldsymbol{\eta}}_i \mathbf{Q}_i \mathbf{Q}_i^T &= \boldsymbol{\eta}_{d,i} \mathbf{Q}_i^T \\ \tilde{\boldsymbol{\eta}}_i &= \boldsymbol{\eta}_{d,i} \mathbf{Q}_i^T \left(\mathbf{Q}_i \mathbf{Q}_i^T\right)^{-1} \\ &= \boldsymbol{\eta}_{d,i} \mathbf{Q}_i^{\#} \end{aligned} \tag{4.6}$$

When the pilot patterns $\mathbf{P}_i$ are equidistant and regularly spaced, the P2DE matrices $\mathbf{Q}_i \mathbf{Q}_i^T$ are typically well-conditioned. However, more irregular patterns can result in ill-conditioned matrices $\mathbf{Q}_i \mathbf{Q}_i^T$, making these matrix inversions unstable. To compensate for this ill-conditioning, we propose to use off-diagonal regularization (ODIR) to condition the P2DE matrices. This form of regularization is described in more detail in Appendix C.

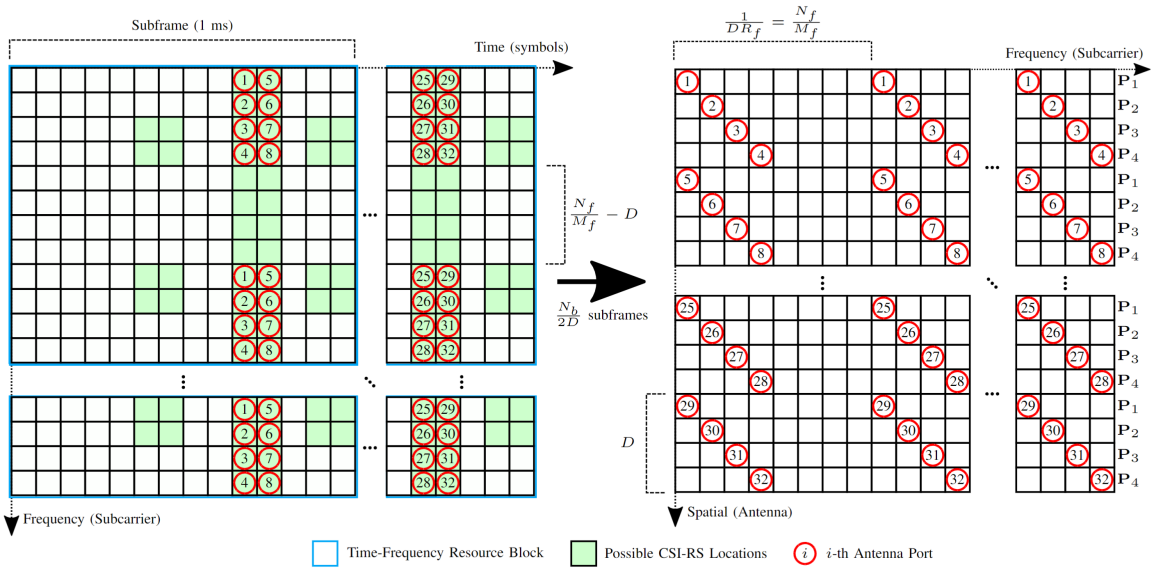


Figure 4.2: (a) LTE Resource Blocks and CSI-RS locations where antenna port pilots are allocated. (b) Schematic for diagonal pilots with relevant parameters, size of diagonal D and frequency downsampling ratio DR_f. In this diagram, $N_b = 32, D = 4, \text{DR}_f = \frac{1}{8}$. The pilot matrix $\mathbf{P}_j$ indicates the downsampling pattern for the j-th element of the diagonal pattern. The number of subframes necessary to populate (b) is inversely proportional to D.

4.1.1 Diagonal Pilot Allocations for 3GPP Standards

As discussed in Section 1.2, 3GPP specifications describe the allocation of pilots to time-frequency resources in 4G/LTE [11, 65] and 5G/NR [12] radio networks, where the reserved resource elements are called CSI reference signals (CSI-RS) for the former and demodulation reference signals (DMRS) for the latter.

In order to connect the P2DE as described in Section 4.1 to the 3GPP specifications, we must specify the corresponding pilot patterns in the time-frequency grid. In Figure 4.2, we show an example of our proposed 'diagonal' pilot pattern in an LTE network using CSI-RS locations. We refer to this pattern as diagonal since it is diagonal in the spatial-frequency domain. The benefit of the diagonal pattern can be understood by considering the time needed to acquire downsampled CSI matrix, $\mathbf{H}_d$.

Algorithm 4.1 shows the process for acquiring the delay domain P2DE from sparse frequency domain pilots.

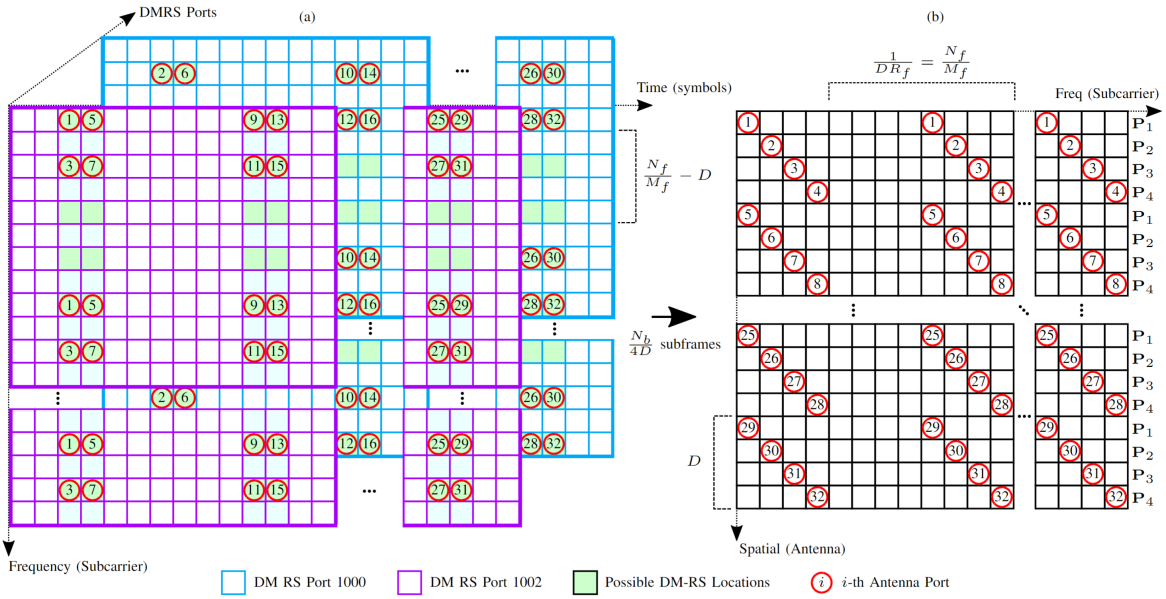


Figure 4.3: (a) 5G NR Resource Blocks and DMRS locations where antenna port pilots are allocated. (b) Schematic for diagonal pilots with relevant parameters, size of diagonal D and frequency downsampling ratio DR_f. In this diagram, $N_b = 32, D = 4, \text{DR}_f = \frac{1}{8}$. The pilot matrix $\mathbf{P}_j$ indicates the downsampling pattern for the j-th element of the diagonal pattern. The number of subframes necessary to populate (b) is inversely proportional to D.

Algorithm 4.1 Pilots-to-delay Estimator (P2D) for Diagonal Pilot Pattern

Input: P2DE Matrices, $\mathbf{Q}^{\#}_{c,j},\ j \in \{1, \ldots, D\}$
Input: Pilot spatial-frequency CSI, $\mathbf{H}_d \in \mathbb{C}^{N_b \times M_f}$
Initialize: Spatial-delay CSI, $\tilde{\mathbf{H}}_\tau \in \mathbb{C}^{N_b \times N_t}$
Initialize: Angular-delay CSI estimate, $\mathbf{H}_\tau \in \mathbb{C}^{N_b \times N_t}$
for $i = 1, 2, \ldots, N_b$ **do**
    # *Index for j-th pilot matrix*
    $j = ((i-1) \bmod D) + 1$
    # *Apply P2D to i-th antenna port*
    $\boldsymbol{\eta}_{d,i} = \mathbf{H}_d(i, :)$
    $\tilde{\mathbf{H}}_\tau(i, :) = \boldsymbol{\eta}_{d,i} \mathbf{Q}^{\#}_{c,j}$
end for
 # *Convert from spatial to angular*
$\mathbf{H}_\tau = \mathbf{F}_{N_b} \tilde{\mathbf{H}}_\tau$
Return $\mathbf{H}_\tau$

4.2 Heterogeneous Differential Encoding with P2DE

Chapter 3 introduced the concept of differential encoding for CSI feedback compression. The delay domain P2DE CSI introduced in this chapter can also be used in the differential encoding framework. Figure 4.5 showcases the data path when utilizing the P2D estimates with a differential encoding network.

The prior work proposed a differential encoding network that used an identical autoencoder at each timeslot. We refer to such a network as 'homogeneous.' In contrast, we can consider a 'heterogeneous' network, where we use different network architectures at different timeslots.

In this section, we describe a heterogeneous differential encoder which combines deep compressive sensing with deep autoencoders.

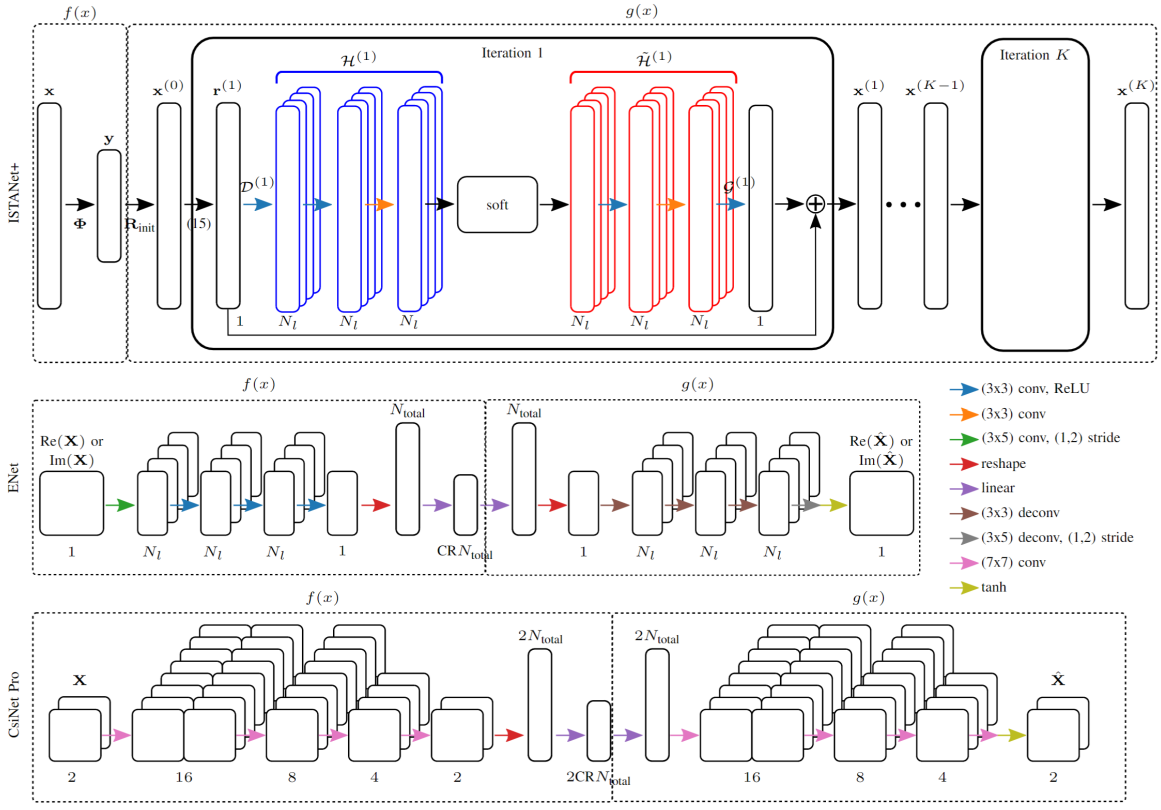


Figure 4.4: Compressive CSI estimation architectures used in this work. $f(x)$ denotes the encoder, and $g(x)$ denotes the decoder. $N_{\text{total}} = N_b N_t$ is the size of the real or imaginary channel. N_l is the number of latent channels in a convolutional layer.

4.2.1 Iterative Optimization Networks for CS-based CSI Feedback

While CNN autoencoders have been dominant in CSI estimation, recent work from image processing has shown promise in using trainable CS algorithms based on CNNs[1]. These works treat iterative CS algorithms as sequential networks by "unrolling" them into discrete blocks [66, 67]. Investigating unrolled CS algorithms for CSI estimation warrants consideration, as CS algorithms can have guaranteed convergence under mild sparsity conditions (in contrast with CNNs autoencoder approaches, which do not have such guarantees). Since CSI data exhibits sparsity in the delay domain, specifying an appropriate compressive sensing approach could provide appreciable performance gains in our differential CSI encoding architecture.

To exploit the temporal coherence of the MIMO channel, we propose to construct a differential encoding network using an unrolled optimization network based on a trainable version of the iterative shrinkage-thresholding algorithm (ISTA), called ISTANet+ [67]. See the top of Figure 4.4 for a diagram of ISTANet+. Denote measurement matrix for the ISTANet+ as

$$\mathbf{\Phi} \in \mathbf{R}^{N_{\text{total}}\text{CR} \times N_{\text{total}}}. \tag{4.7}$$

For compressive sensing approaches, the measurement matrix is analogous to the 'encoder' of autoencoder approaches, i.e., $f(x) = \mathbf{\Phi}x$. The 'decoder' consists of K iterations of the following update steps,

$$\mathbf{r}^{(k)} = \mathbf{x}^{(k-1)} - \rho^{(k)}\mathbf{\Phi}^{\top}(\mathbf{\Phi}\mathbf{x}^{(k-1)} - \mathbf{y}) \tag{4.8}$$

$$\mathbf{x}^{(k)} = \mathbf{r}^{(k)} + \mathcal{G}^{(k)}\left(\tilde{\mathcal{H}}^{(k)}\left(\text{soft}\left(\mathcal{H}^{(k)}(\mathcal{D}^{(k)}(\mathbf{r}^{(k)}), \theta^{(k)})\right)\right)\right) \tag{4.9}$$

where $\mathbf{y} = \mathbf{\Phi}\mathbf{x}$, $\mathbf{x}^{(0)} = \mathbf{R}_{\text{init}}\mathbf{y}$, and $\mathbf{R}_{\text{init}} = \mathbf{X}\mathbf{Y}(\mathbf{Y}\mathbf{Y}^{\top})^{-1}$ is the initialization matrix for

[1]For an overview of conventional compressive sensing solutions as well as the original ISTA algorithm, see Appendix D

the training data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_{\text{train}}}]$ and the training measurement matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{N_{\text{train}}}]$. 'soft(·)' denotes the soft threshold function,

$$\text{soft}(x, \theta) = \text{sign}(x)\text{ReLU}(|x| - \theta). \tag{4.10}$$

$\mathcal{G}^{(k)}, \mathcal{D}^{(k)}, \mathcal{H}^{(k)}, \tilde{\mathcal{H}}^{(k)}$ indicate trainable nonlinear mappings (in this case, CNNs), and $\mathcal{H}^{(k)}, \tilde{\mathcal{H}}^{(k)}$ are subject to the symmetry constraint $\mathcal{H}^{(k)} \circ \tilde{\mathcal{H}}^{(k)} = \mathbf{I}$[2]. The rectified linear unit (ReLU) is given as

$$\text{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

In the proposed differential encoding scheme, we use an instance of ISTANet+ in the first timeslot, t_1, with a large compression ratio such that $\text{CR}_{t_1} \geq \text{CR}_{t_i}$ for all $i > 1$. This choice in compression ratio allows us to initialize the network with a high-quality estimate at the first timeslot. Notably, the training data matrix, $\mathbf{X}$, differs between timeslots. For the first timeslot, the data vectors $\mathbf{x}_i$ are vectorized versions of the CSI matrices,

$$\mathbf{x}_j = \text{vec}\left(\mathbf{H}_{\tau,1}^{(j)}\right) \text{ for } j \in [N_{\text{train}}]. \tag{4.11}$$

However, the data vectors for all other timeslots are vectorized versions of the error matrices,

$$\mathbf{x}_j = \text{vec}\left(\bar{\mathbf{E}}_i^{(j)}\right) \text{ for } j \in [N_{\text{train}}]. \tag{4.12}$$

Denote the parameters for ISTANet+ in the t_i-th timeslot as $\mathbf{\Theta}_{t_i} = \{\mathcal{G}^{(k)}, \mathcal{D}^{(k)}, \mathcal{H}^{(k)}, \tilde{\mathcal{H}}^{(k)}, \theta^{(k)}, \rho^{(k)}\}_{k=1}^{K}$.

[2]Where ∘ denotes the function composition, $(f \circ g)(x) = f(g(x))$

The loss function is a weighted sum of the MSE and the symmetry constraint, i.e.,

$$L(\mathbf{\Theta}_{t_i}) = L_{\text{MSE}} + \alpha L_{\text{sym}} \tag{4.13}$$

$$L_{\text{MSE}} = \frac{1}{N_{\text{batch}} N_{\text{total}}} \sum_{i=1}^{N_{\text{batch}}} \|\mathbf{x}_i^{(K)} - \mathbf{x}_i\|_2^2 \tag{4.14}$$

$$L_{\text{sym}} = \frac{1}{N_{\text{batch}} N_{\text{total}}} \sum_{i=1}^{N_{\text{batch}}} \sum_{k=1}^{K} \|\tilde{\mathcal{H}}^{(k)}(\mathcal{H}^{(k)}(\mathbf{x}_i)) - \mathbf{x}_i)\|_2^2 \tag{4.15}$$

where $N_{\text{total}} = N_b N_t$ is the size of the truncated CSI matrix, K is the number of iterations in ISTANet+, and N_{batch} is the batch size used during training. As denoted in equations (4.11) and (4.12), the vectors $\mathbf{x}_i$ depend on the timeslot.

4.3 Results

4.3.1 Accuracy of P2DE

We assess the accuracy of the P2DE under values of M_f and D. Fig. 4.6 demonstrates the accuracy of the P2DE at the UE (i.e., before compression and feedback) for different frequency downsampling ratios. The P2DE achieves impressive accuracy even under aggressive values of DR_f (e.g., better than -30dB at $\text{DR}_f = \frac{1}{8}$). Additionally, the effect of increasing the diagonal size, D, is apparent at larger compression ratios (i.e., CR $\geq \frac{1}{8}$), where the accuracy of the P2DE to a value as low as -25dB. For smaller compression ratios (i.e., CR $\leq \frac{1}{16}$), increasing D has a marginal effect on the accuracy of the P2DE.

We assess the accuracy of the P2DE assuming noise from pilot estimation. To simulate pilot estimation error, we use additive Gaussian noise,

$$\hat{\mathbf{H}}_d = \mathbf{H}_d + \mathbf{N}_d$$

where the elements of $\mathbf{N}_d$, $\mathbf{N}_d(i,j) \sim \mathcal{N}(0, \sigma^2)$ for $i \in [1, 2, \ldots, N_b]$, $j \in [1, 2, \ldots, M_f]$.

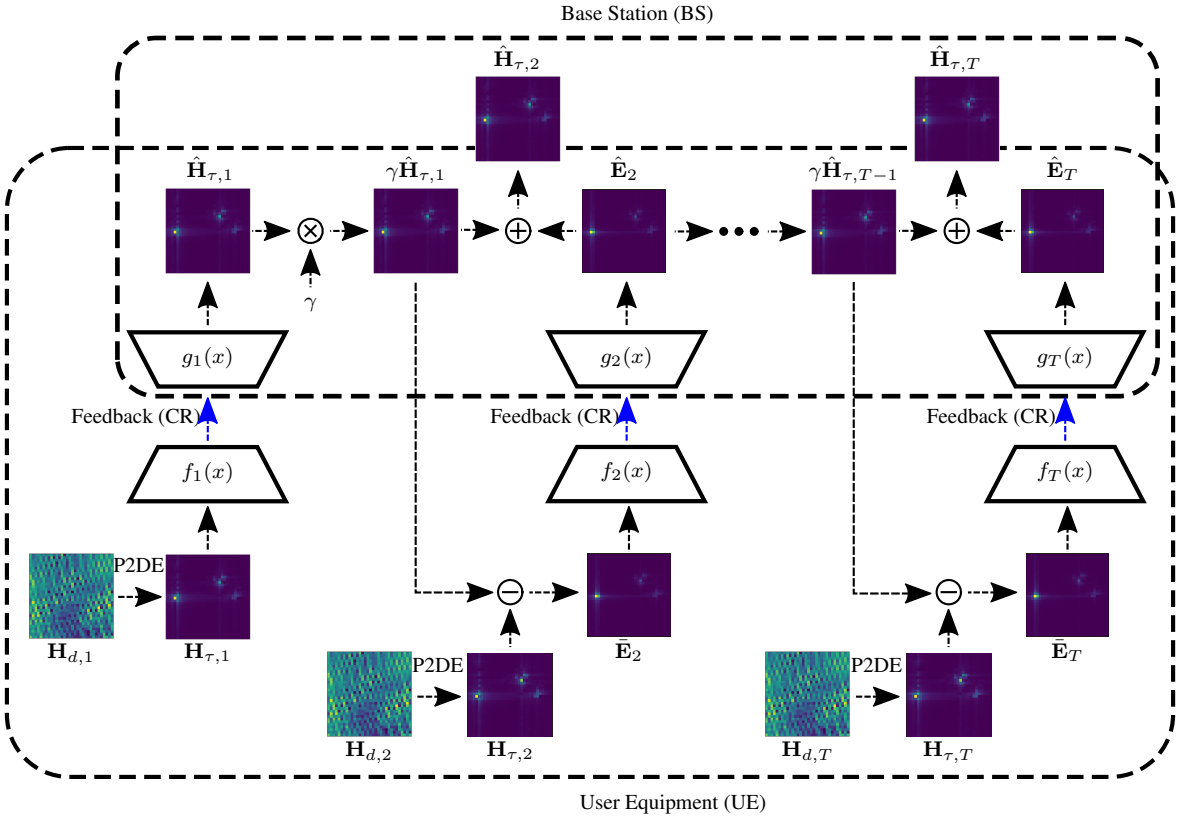


Figure 4.5: Diagram of a CSI estimation network using compressed differential feedback based on the linear P2DE. First, downlink pilots are used to estimate a downsampled frequency domain CSI estimate, $\bar{\mathbf{H}}_t \in \mathbb{C}^{N_b \times M_f}$ where $M_f << N_f$ at the t-th timeslot. Then, the P2DE $\mathbf{Q}_{N_t}^{\#}$ of Algorithm 4.1 is applied to estimate $\tilde{\mathbf{H}}_t$. After P2DE, the learnable transforms $f_t(x)$ and $g_t(x)$ are used to compress and decode the feedback, respectively. For $t = 1$, the encoder/decoder are applied directly to $\tilde{\mathbf{H}}_1$. In all subsequent timeslots ($t > 1$), the differential term $\mathbf{E}_t$ is compressed and fed back.

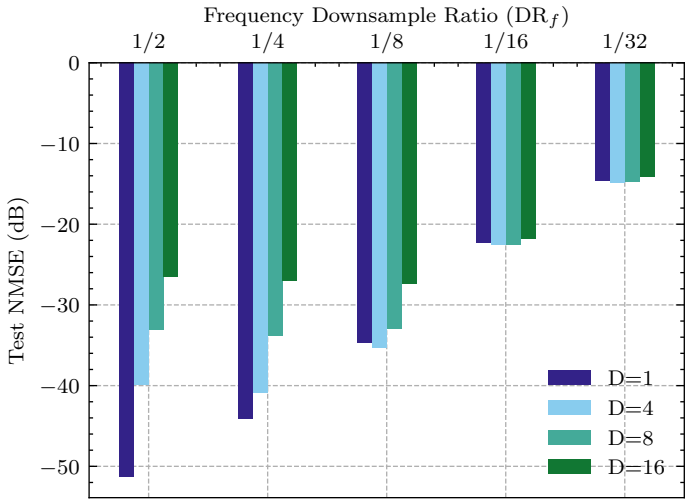


Figure 4.6: P2D estimation performance under different frequency downsampling ratios ($\text{DR}_f = \frac{M_f}{N_f}$) and diagonal dimensions (D) for the Outdoor COST2100 dataset. Downsampling is done along the frequency axis.

To achieve different SNR values for $\hat{\mathbf{H}}_d$, we simply vary the noise variance σ^2, and we use the P2DE at different pilot estimation noise levels. Figure 4.7 shows the accuracy of the P2DE for different values of σ^2.

In addition to varying the pilots estimation SNR, we also showcase the effect of varying δ (i.e., the ODIR parameter as described in Appendix C). We observe that δ helps the P2DE achieve better performance under both low-noise and noisy conditions, i.e.,

- **Low-noise condition (SNR $= -20$ dB)**: The P2DE goes from -8 dB to -22 dB for $\delta = 0$ and $\delta = 0.5$, respectively.
- **Noisy condition (SNR ≥ -10 dB)**: The P2DE goes from -9 dB to -30 dB for $\delta = 0$ and $\delta = 0.5$, respectively.

4.3.2 P2DE Compression Network Comparison

Having assessed the initial accuracy of the P2DE at the UE, we now apply a deep learning network to compress the output of the P2DE. Figure 4.8 demonstrates the accuracy of IS-TANet+ [67] for multiple compression ratios (CR) using the P2DE as its input. For progres-

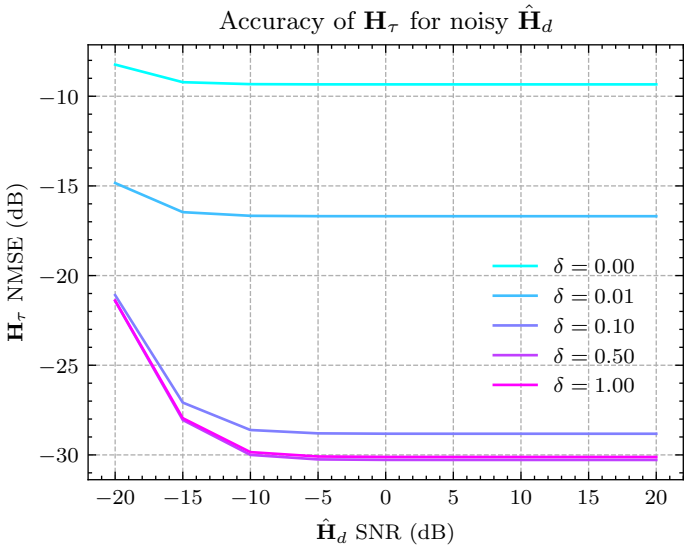


Figure 4.7: Accuracy of P2DE output, $\mathbf{H}_\tau$, assuming noisy pilots, $\hat{\mathbf{H}}_d$. Additive Gaussian noise is used to model the error inherent in pilot estimation. Here, $D = 4, \text{DR}_f = \frac{1}{32}$.

sively smaller compression ratios, the accuracy of ISTANet+ remains stable until $\text{DR}_f = \frac{1}{32}$, at which point the network's performance degrades.

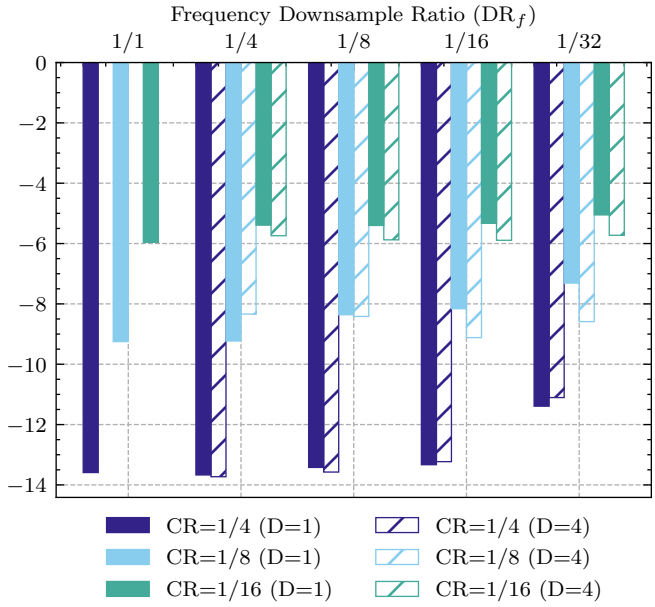


Figure 4.8: Performance of ISTANet+ for multiple compression ratios using P2D estimates with different downsampling ratios ($\text{DR}_f = \frac{M_f}{N_f}$) for the Outdoor COST2100 dataset. Non-diagonal pattern ($D = 1$) is compared with a diagonal pattern of size $D = 4$. Performance for $\text{DR}_f = 1/1, D = 4$ is omitted since it is equivalent to the $\text{DR}_f = 1, D = 1$ case.

In addition to ISTANet+, we assess the accuracy deep CNN autoencoders using P2DE as an input. Figure 4.9 shows the accuracy of ISTANet+ compared to ENet [68] and SphNet [2]. The performance of ISTANet+ is better than both ENet and SphNet at larger compression ratios (CR $\in [\frac{1}{4}, \frac{1}{8}]$), and the performance of ISTANet+ and ENet is comparable at CR $= \frac{1}{16}$.

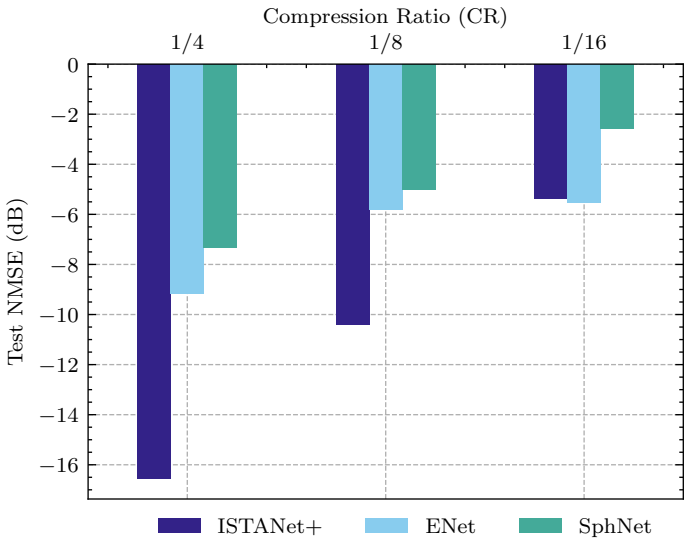


Figure 4.9: Performance comparison for different feedback compression networks using P2D estimates (DF$_f = 1/16, D = 4$) for Outdoor COST2100 dataset. For all tested networks, we use $N_{\text{phase}} = 4$, resulting in an augmented training set with 80k samples.

To assess the accuracy of these networks under quantization, we also conduct an experiment where the latent feedback elements are subjected to μ-law companding and uniform quantization (as described in Section 3.3). We present these results in Figure 4.10.

4.3.3 Heterogeneous Differential Encoding Networks

Figure 4.11 shows the performance of the proposed heterogeneous differential encoding networks compared to homogeneous networks. We observe that ENet provides worse initial performance than ISTANet+ (i.e., at t_1) but provides a more improvement in accuracy than ISTANet+ in subsequent timeslots (i.e., at $t_2, t_3, \dots$). Based on this observation, we expect the best network configuration to be the heterogeneous network, MN-IE. Figure 4.11 supports this reasoning, as MN-IE achieves better asymptotic performance (i.e., as i for t_i

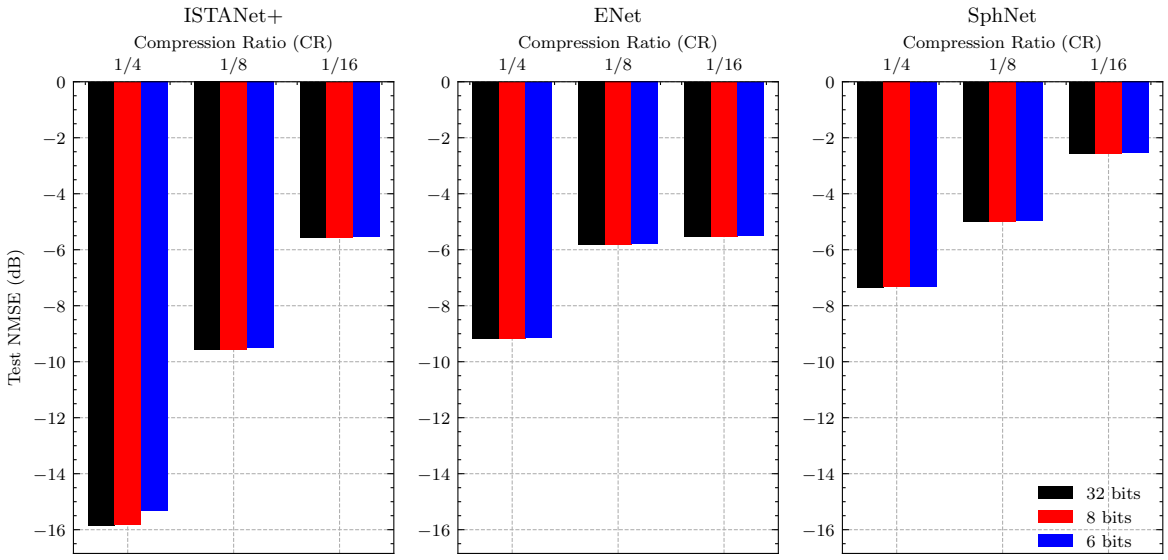


Figure 4.10: Tested networks where feedback is subject to μ-law companding ($\mu = 255$) and uniform quantization for different numbers of quantization bits. P2DE parameters are $D = 4, \text{DR}_f = \frac{1}{16}$.

increases) than MN-I.

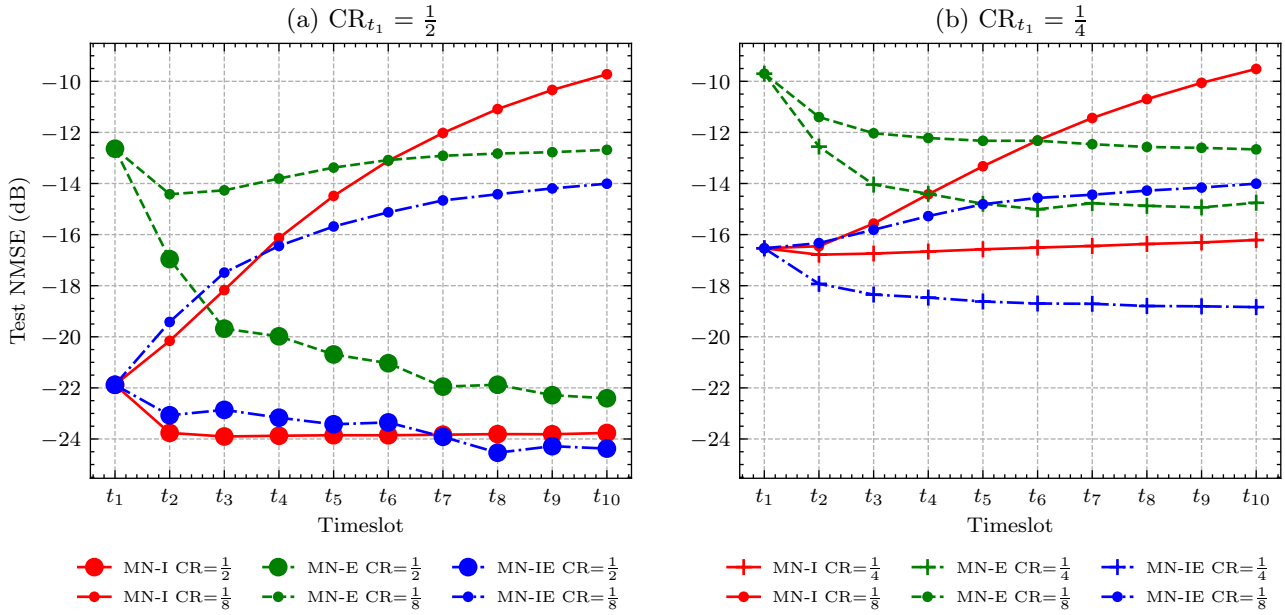


Figure 4.11: Compressive CSI estimation using differential encoding and linear P2D estimator ($M_f = 128, \text{DR}_f = \frac{1}{8}, D = 4$). MarkovNet-ISTA (MN-I), MarkovNet-ENet (MN-E), and MarkovNet-ISTA-ENet (MN-IE) are tested using two different compression ratios in the first timeslot, $\text{CR}_{t_1} \in \left[\frac{1}{2}, \frac{1}{4}\right]$.

4.3.4 Computational Complexity

We assess the computational complexity (as defined in Chapter 2, Section 2.1) of the network architectures tested in this work. While ISTANet+ has superior accuracy compared to the autoencoder networks, we observe that its computational complexity is much higher. This discrepancy in complexity further motivates the heterogeneous network architecture of MN-IE. While MN-I uses T copies of ISTANet+, MN-IE uses one copy of ISTANet+ to provide an initial estimate and $T-1$ copies of ENet to compress the differential term.

Table 4.1: Computational complexity of networks used in this work. **Bold face** in a column indicates lowest value for given compression ratio. "CR" = compression ratio, "Enc" = encoder, "Dec" = decoder. FLOPs indicate computation during inference (i.e., not training/back-propagation).

		Parameters (M)				FLOPs (M)	
		Trainable		All			
	CR	Enc	Dec	Enc	Dec	Enc	Dec
ISTANet+	$1/2$	**0.00**	**0.34**	2.10	4.54	**2.10**	393.78
	$1/4$	**0.00**	0.34	1.05	2.44	**1.05**	373.85
	$1/8$	**0.00**	0.34	0.52	1.39	**0.52**	363.89
	$1/16$	**0.00**	0.34	0.26	0.87	**0.26**	358.91
ENet	$1/2$	0.55	0.55	**0.55**	**0.55**	29.98	29.70
	$1/4$	0.29	**0.29**	**0.29**	**0.29**	29.46	29.18
	$1/8$	0.16	**0.16**	**0.16**	**0.16**	29.20	28.92
	$1/16$	0.09	**0.09**	**0.09**	**0.09**	29.07	28.79
CsiNet Pro	$1/2$	1.06	1.06	1.06	1.06	12.16	**12.16**
	$1/4$	0.53	0.53	0.53	0.53	11.11	**11.11**
	$1/8$	0.27	0.27	0.27	0.27	10.59	**10.59**
	$1/16$	0.14	0.14	0.14	0.14	10.33	**10.33**

Chapter 5

Improving Computational Efficiency

The prior works discussed in this dissertation have leveraged domain knowledge to improve the performance of deep neural networks for CSI compression. In this chapter, we consider some methods for improving the efficiency of such networks.

In Chapter 4, we considered how to use frequency domain pilots at the UE to estimate the truncated delay domain. While it is advantageous to acquire the delay domain CSI at the UE from a sparsity perspective, utilizing the P2DE at the UE creates additional computational and memory burden. Consider the P2DE estimator as outlined in Algorithm 4.1. The algorithm iterates over the N_b antennas and performs the matrix multiplication $\eta_{d,i}\mathbf{Q}_{c,j}^{\#}$ in each iteration. To run the algorithm, $N_b M_f(N_f - 1)$ FLOPs are required[1], and $2D(M_f N_f)$ parameters must be stored[2]. Under the parameters used in Chapter 4 Section 4.3, utilizing the P2DE at the UE could result in an additional 1.04×10^6 FLOPs and 2.62×10^5 parameters. The additional computational burden of the P2DE is summarized in Table 5.1.

Since UEs are typically compute/memory constrained devices (e.g., cell phones, tablets, IoT devices), it is preferable to reduce any such burden where possible. In this chapter, we propose two methods to reduce computational complexity at the UE. In Section 5.1, we

[1]Considering the $M_f(N_f - 1)$ FLOPs involved in matrix multiplication (as discussed in Appendix A.3), and considering that this matrix multiplication occurs N_b times.

[2]Considering the matrices $\mathbf{Q}_i^{\#} \in \mathbb{C}^{M_f \times N_f}$ for $i \in [1, \ldots, D]$.

Table 5.1: Computational complexity of P2DE for $D = 1$ (diagonal pattern size), $N_f = 1024$ (number of subcarriers), and $N_b = 32$ (number of antennas in uniform linear array).

$\mathbf{M_f}$	**32**	**64**	**128**
FLOPs	$1.05 \cdot 10^6$	$2.10 \cdot 10^6$	$4.19 \cdot 10^6$
Parameters	$6.55 \cdot 10^4$	$1.31 \cdot 10^5$	$2.62 \cdot 10^5$

propose a method to estimate the delay domain CSI at the BS based on compressive sensing feedback of the frequency domain pilots. In Section 5.2, we propose a method for using a simple model to estimate subsets of subcarriers in CSI matrices. In Section 5.3, we present results demonstrating that these methods maintain CSI estimation accuracy while reducing encoder-side complexity by as much as a factor of 10.

5.1 Direct Pilot-based Feedback

Many works in deep learning based compressive CSI estimation have leveraged delay domain CSI given its sparsity [1]. Recent work has confirmed the viability of acquiring the truncated delay domain at the UE under realistic conditions (i.e., using sparse frequency-domain pilots) [4].

However, the process of acquiring delay domain CSI places additional computational burden on the UE, and given the computational constraints of typical UEs (e.g., cell phones, tablets, IoT devices), minimizing the computational burden on the UE should be prioritized.

To reduce the computation performed at the UE, we propose to move the computation of the delay domain to the BS by compressing and feeding back the frequency domain pilots. The data flow of this scheme is illustrated in Fig 5.1, where the P2DE (see [4] for full details) is utilized at the BS rather than the UE.

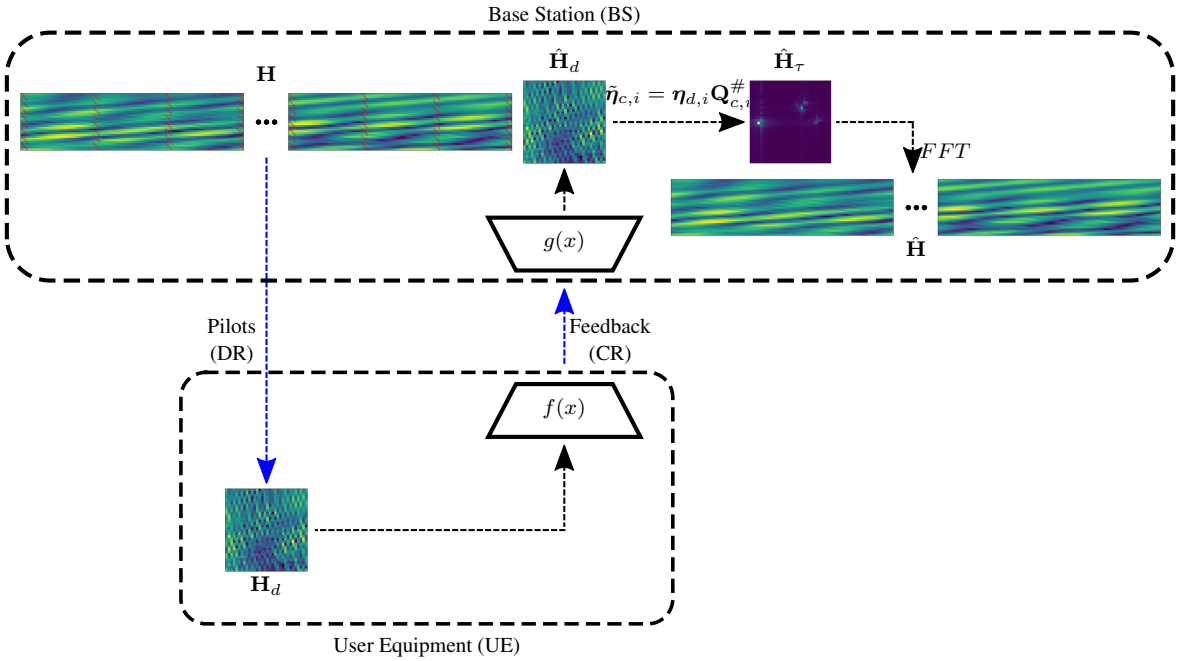


Figure 5.1: Compressive CSI estimation based on linear P2D estimator on BS side. First, we use downlink pilots to generate a sparse, frequency domain CSI estimate of size $M_f << N_f$. This downsampled frequency domain CSI estimate is compressive sensing at the UE and fed back to the BS. At the BS, we apply the P2D estimator, $\mathbf{Q}^{\#}_{N_t}$ [4] to acquire the truncated delay domain CSI estimate. We train a learnable encoder, $f(x)$, and decoder, $g(x)$, to compress and decode the feedback, respectively. The BS recovers the frequency domain CSI from the decoded delay domain CSI estimate.

5.2 Model Re-use

In order to enjoy the benefits of highly accurate CS networks while reducing network complexity at the UE, we propose to use a relatively simple CS model on contiguous blocks of K subcarriers where $K \leq M_f$. The operating principle behind this approach is inspired by block compressive sensing [69], and the concept is shown in Fig. 5.2.

In this work, we study a deep CS network called ISTANet+ [67]. For ISTANet+, the compression occurs at the UE by a simple matrix multiplication,

$$f(x) = \mathbf{\Phi x} \tag{5.1}$$

where $\mathbf{\Phi}$ is referred to as the measurement matrix and $\mathbf{x}$ is the flattened CSI matrix. In the case of block compressive sensing, the measurement matrix operates of K contiguous subcarriers across all angular indices, i.e., $\mathbf{x} \in \mathbb{R}^{N_K}$, $\mathbf{\Phi} \in \mathbb{R}^{\mathrm{CR}K \times K}$ where $N_K = 2KN_a$ is the dimension of the K subcarriers from the CSI matrix. The decoder of ISTANet+ is identical to the original paper, the only major differnce being the dimension of the latent 2D convolutions to accommodate K subcarriers rather than all M_f pilot subcarriers.

To produce a full CSI estimate at the BS, the encoder is run $\frac{M_f}{K}$ times on adjacent blocks of subcarriers, producing multiple feedback payloads. At the BS, each of these payloads is decoded to estimate a block of subcarriers, and the final CSI estimate is simply the concatenation of all these decoded payloads.

5.3 Results

To evaluate the effectiveness of direct frequency-domain CSI feedback and of model re-use, we use CSI data generated by the COST2100 model in an Indoor and an Outdoor scenario [20]. The parameters used to generate both datasets are given at the end of Chapter 1 Section 1.3. For all tests, we utilize ISTANet+ with spherical normalization [2].

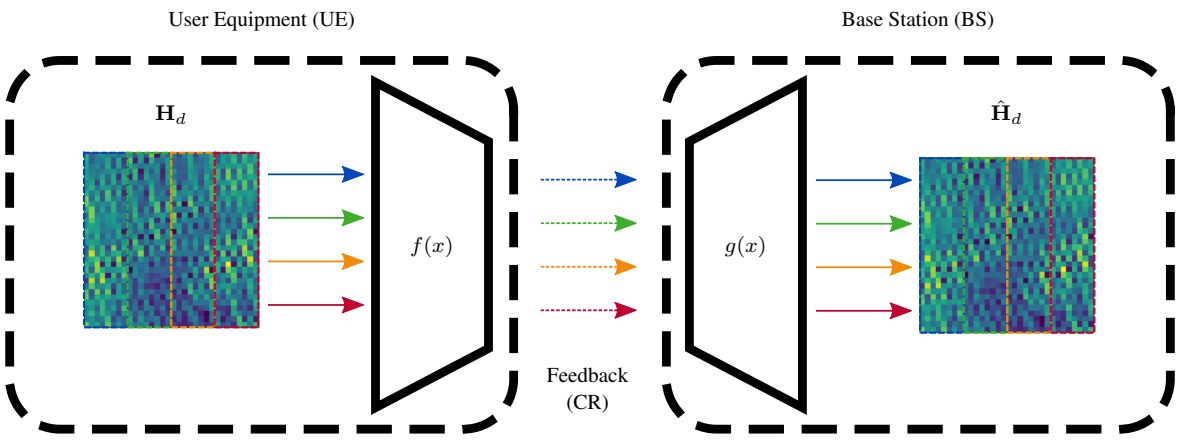


Figure 5.2: Compressive CSI estimation with model re- use. Rather than compressing the entire input, $\mathbf{H}_d$, the encoder compresses K contiguous subcarriers from the input, resulting in $\frac{M_f}{K}$ payloads of feedback (shown in different colors above). Meanwhile at the BS, the decoder recovers each set of K contiguous subcarriers based on the $\frac{M_f}{K}$ payloads, and the decoded payloads are combined to generate the full frequency domain estimate.

5.3.1 Accuracy vs. K Subcarriers

Figures 5.3 and 5.4 compare the reused models with frequency domain compression against the original delay domain model (shown in red in both Figures). We test the networks using $K \in [2, 4, 8, 16, 32, 64, 128]$ with the requirement that $K < M_f$. We observe that in the Indoor network, there is a gradual increase in NMSE as the number of subcarriers reduces. In the case of the Outdoor network, the frequency domain-based feedback results in a drop in NMSE by about 2.5 dB.

5.3.2 Accuracy vs. Network Complexity

In Figures 5.5 and 5.6, we compare the reused models with frequency domain compression against the original model with delay domain compression (shown in red in both Figures). Both Figures show the performance of the given model vs. a complexity metric, either log(FLOPs) or the number of model parameters. The complexity of the model varies with K, which we vary in the same way as described above in Section 5.3.1. Importantly, each complexity metric is given for the *encoder* (i.e., the UE-side complexity).

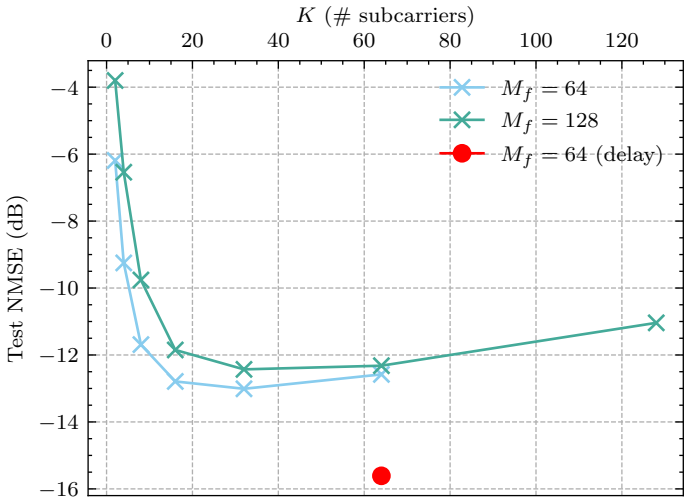


Figure 5.3: NMSE vs. K subcarriers of shared model architectures (i.e., complexity is w.r.t. FLOPs and parameters at UE). Outdoor COST2100 model.

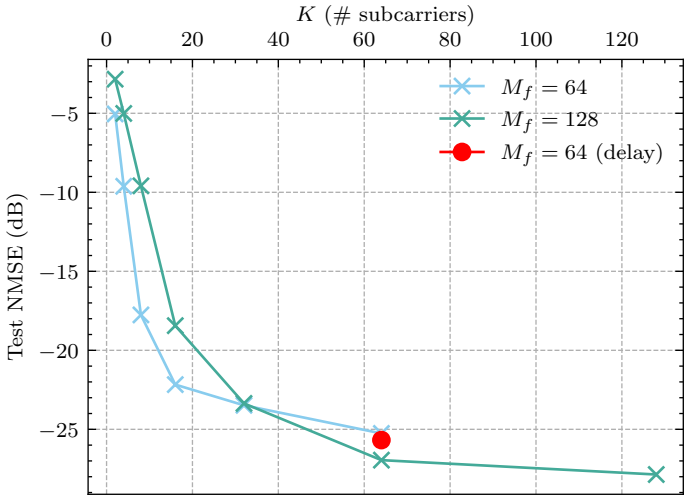


Figure 5.4: NMSE vs. K subcarriers for shared model architectures (i.e., complexity is w.r.t. FLOPs and parameters at UE). Indoor COST2100 model.

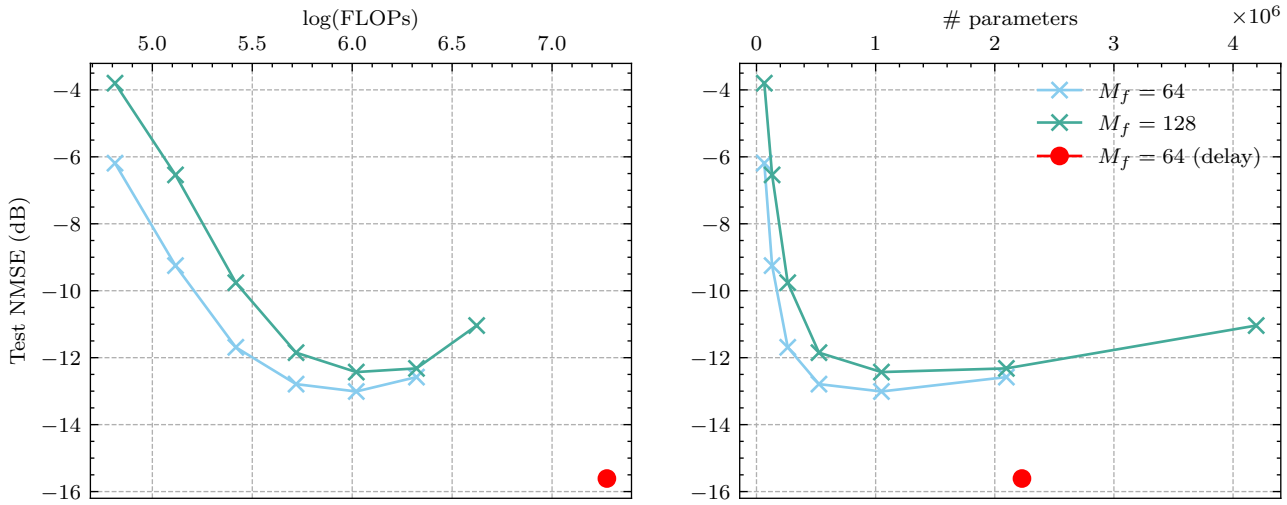


Figure 5.5: NMSE vs. encoder complexity of shared model architectures (i.e., complexity is w.r.t. FLOPs and parameters at UE). Outdoor COST2100 model.

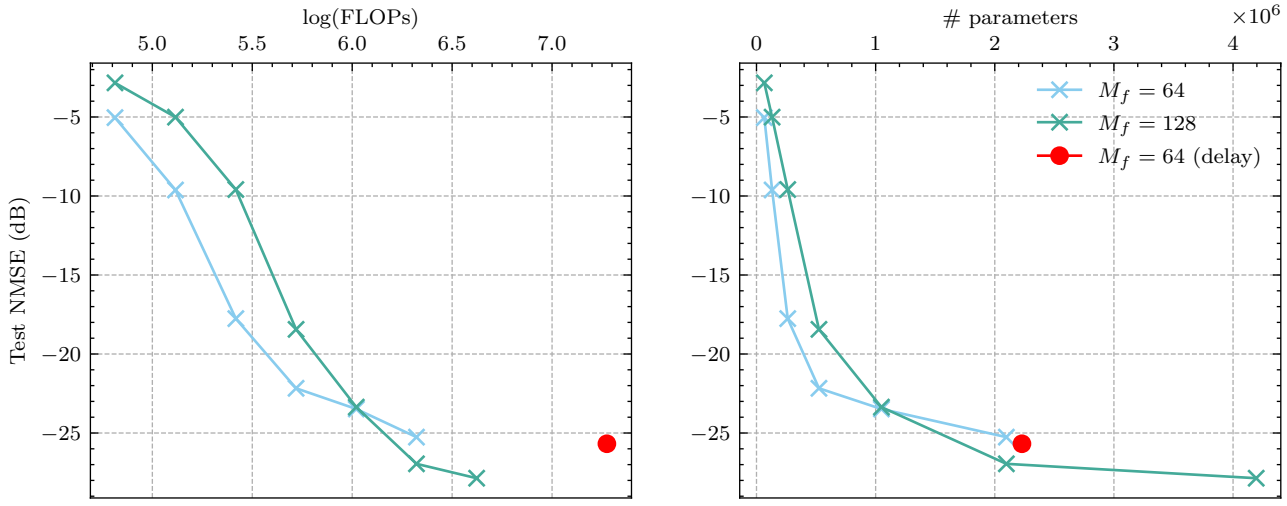


Figure 5.6: NMSE vs. encoder complexity for shared model architectures (i.e., complexity is w.r.t. FLOPs and parameters at UE). Indoor COST2100 model.

Figures 5.7 and 5.8 show the same performance and complexity metrics as Figures 5.5 and 5.6 but with respect to the full model complexity (i.e., encoder and decoder, UE-side and BS-side). When it comes to parameters, we observe trends that are similar to the encoder-side parameters in both environments. However, the number of FLOPs for the $M_f = 128$ case is twice that of the the $M_f = 64$ case since the number of FLOPs of the decoder scales linearly with M_f.

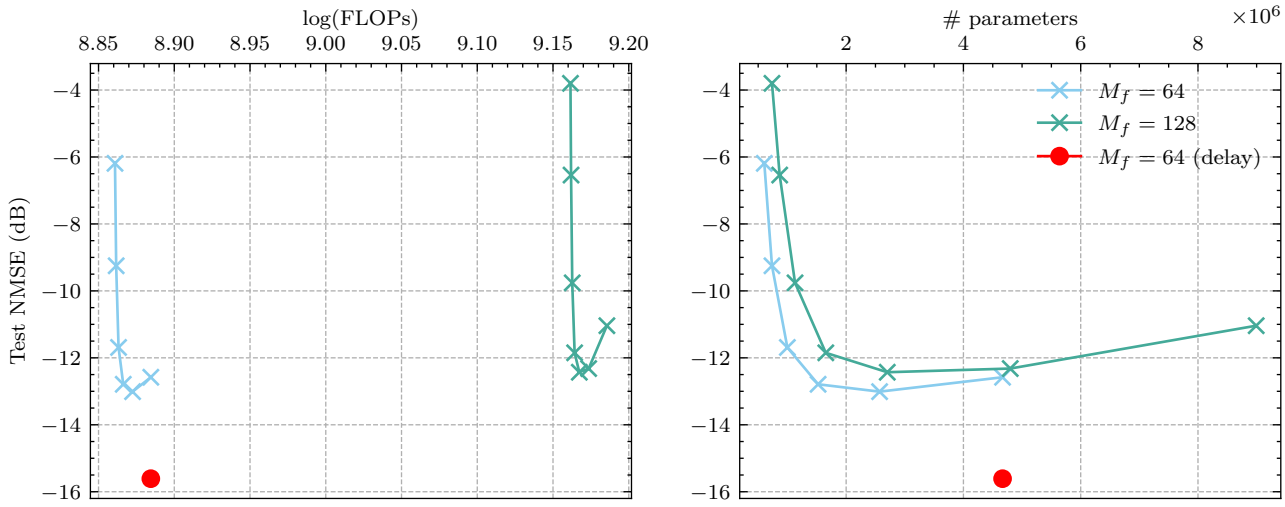


Figure 5.7: NMSE vs. complexity of shared model architectures. **Left**: Complexity w.r.t. FLOPs of encoder and decoder. **Right**: Complexity w.r.t. parameters of encoder and decoder.). Outdoor COST2100 model.

5.4 Discussion

In this chapter, we presented two methods for reducing the complexity of deep learning based CSI compression networks. We first proposed to move the delay domain estimation to the BS by compressing and feeding back frequency domain information to directly estimate the pilots at the UE. We then proposed to use a deep learning model to estimate blocks of K contiguous subcarriers. We tested a combination of both of these approaches in both the Indoor and Outdoor COST2100 scenarios, and we demonstrated that these approaches can reduce the UE-side computational complexity while maintaining or slightly increasing the

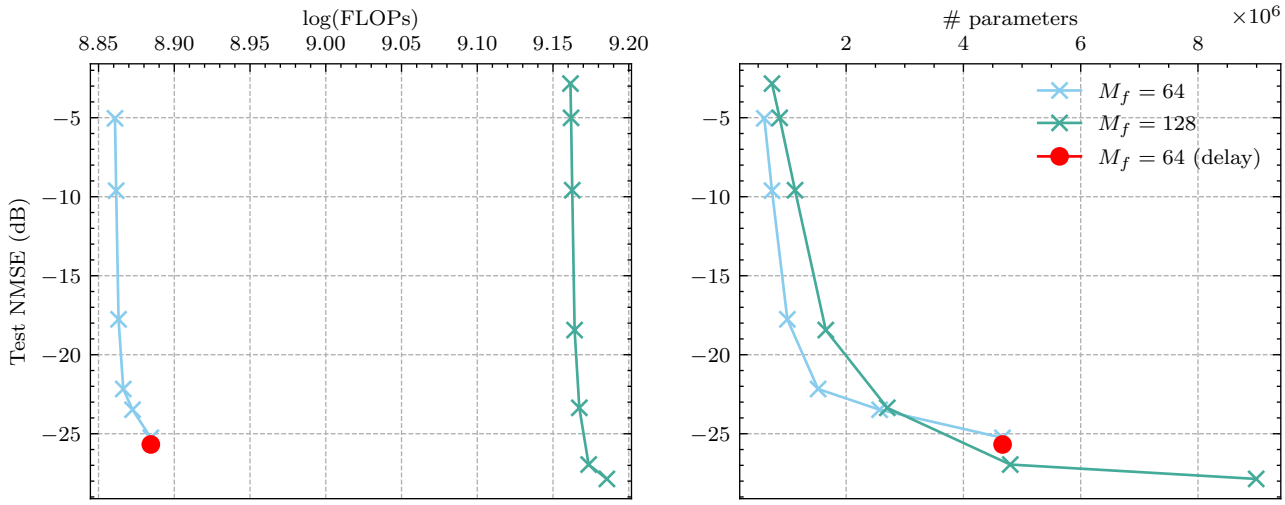


Figure 5.8: NMSE vs. complexity for shared model architectures. **Left**: Complexity w.r.t. FLOPs of encoder and decoder. **Right**: Complexity w.r.t. parameters of encoder and decoder.). Indoor COST2100 model.

network's final accuracy.

Chapter 6

Conclusion

This dissertation investigates techniques to improve the performance and efficiency of deep neural networks for the task of MIMO CSI estimation. In Chapter 2 we discussed the importance of data pre-processing techniques, and we showed the efficacy of our proposed spherical normalization technique. In Chapter 3, we exploited the temporal correlation of the wireless channel, and we demonstrated the superior performance and efficiency of a deep differential encoder compared to recurrent neural networks. In Chapter 4, we presented two main contributions: an accurate estimator of the delay domain CSI based on sparse frequency domain pilots and a hetergeneous differential encoding network combining deep compressive sensing networks with autoencoders. We showed the accuracy of our pilot-based delay domain estimator, even under aggressive sparsity and noisy pilot estimates. Furthermore, we verified the improved performance of heterogeneous networks over homogeneous networks. Finally in Chapter 5, we proposed a scheme which re-uses a simple model on contiguous blocks of multiple subcarriers, and we showed that this method can maintain accuracy while reducing the computational complexity of the network by a factor of 10.

Across all these works, we investigated techniques which exploited domain knowledge of the wireless channel to improve estimation accuracy and computational efficiency while better conforming to 3GPP protocols. Further work in CSI compression should take a sim-

ilar approach by taking advantage of features of the wireless channel, the communications protocol, or CSI data itself.

6.1 Future Works

In addition to the work discussed in this dissertation, there are important additional directions which future works in deep learning for CSI compression should address. Here, we discuss a few such directions, including compression bounds for CSI estimation, networks with trainable codewords, and ...

6.1.1 Rate-distortion Bounds for CSI Feedback

Many works provide results for their proposed CSI feedback networks using the NMSE at a small number of compression ratios. This approach allows for fair comparison between comparable networks/algorithms, but it does not answer a more important question: *At a given compression ratio, what is the theoretical distortion limit?*

Information theory provides us with a framework to answer this question: the rate-distortion curve. The rate-distortion of a random variable describes the optimal error that can be achieved whenever that variable is encoded using a given number of bits.

The challenge with applying rate-distortion theory to MIMO CSI data is the lack of well-defined distributions for practical channel data. While rate-distortion bounds are known for well-defined distributions (e.g., univariate or multivariate Gaussian distributions), the same can not be said for empirical data.

One possible approach to constructing a rate-distortion curve is to estimate the differential entropy of quantized CSI data. For a CSI matrix, $\mathbf{H}$, assume i.i.d. $\mathbf{H}_{(i,j)}$ where i (j) denotes the row (column) of $\mathbf{H}$. The differential entropy of the (i,j)-th element is

$$h(\mathbf{H}_{(i,j)}) = -\int p(\mathbf{H}(i,j)=k)\log p(\mathbf{H}_{(i,j)}=k)dk,$$

In practice, the distribution $p(\mathbf{H}_{(i,j)})$ is difficult to obtain. We can instead resort to the Kozachenko–Leonenko (KL) estimator [70] for each element in $\mathbf{H}$ and average over the elements,

$$h(\mathbf{H}) \leq \sum_{i}^{R_d} \sum_{j}^{n_T} \hat{h}(\mathbf{H}_{(i,j)}) = h_{\text{UB}}(\mathbf{H}), \tag{6.1}$$

for KL estimator $\hat{h}$. Based on Theorem 8.3.1 from Cover [71], for sufficiently small quantization interval $\Delta = \frac{1}{2^n}$, the entropy of a quantized random variable is related to its differential entropy as,

$$H(\mathbf{H}^{\Delta}) = h(\mathbf{H}) + n, \tag{6.2}$$

for n-bit quantization. Thus, the differential entropy estimator admits an estimate for the entropy of the quantized CSI, $\hat{H}(\mathbf{H}^{\Delta}) = \hat{h}(\mathbf{H}) + n$.

To establish a rate-distortion curve, we can use the estimator outlined above on CSI data with Gaussian noise, i.e.

$$\mathbf{H}_{\sigma,(i,j)} = \mathbf{H}_{(i,j)} + v \text{ for i.i.d } v \sim \mathcal{N}(0, \sigma^2).$$

Using the corrupted CSI matrices $\mathbf{H}_{\sigma} = \left[\mathbf{H}_{\sigma,(i,j)}\right]_{i\in[R_d],j\in[N_b]}$, we calculate the bounds $\hat{h}(\mathbf{H}_{\sigma}^{\Delta})$ from or (6.1) for different noise levels σ to establish a rate-distortion curve.

6.1.2 Trainable Codewords

In addition to estimating the rate-distortion bounds of CSI data, new works should investigate techniques for improving the compression efficiency of CSI feedback networks. Some prior work has investigated feedback quantization in deep learning-based CSI compression. In [72], the authors propose DeepCMC, an autoencoder structure where the continuous compressed elements are discretized via uniform quantization then encoded using

context adaptive binary arithmetic coding (CABAC) [73]. Since uniform quantization is non-differentiable, the authors do not perform true quantization during training and instead apply uniformly distributed noise to approximate quantization noise [72]. In [74], the authors propose AnalogDeepCMC, which encodes latent elements as power-normalized complex elements and decodes using maximal ratio combining. The authors also report the achieved rate of AnalogDeepCMC for different CSI overhead ratios.

Further work into trainable codewords might borrow ideas from deep learning based image compression. For example, the soft-to-hard vector quantization (SHVQ) framework [75] could be used to imbue a CSI compression network with quantized codewords. To describe the framework, we choose a vector dimension, m, by which to partition the latent space $\mathbf{Z} = f_e(\mathbf{H}, \theta_e)$, and we denote the vectorized version of $\mathbf{Z} \in \mathbb{R}^r$ as $\tilde{\mathbf{Z}} \in \mathbb{R}^{r/m \times m}$. We define the m-dimensional codebook of size L as $\mathbf{C} \in \mathbb{R}^{m \times L}$. The soft assignments of the j-th latent vector $\tilde{\mathbf{z}}_j$ can be written as

$$\phi(\tilde{\mathbf{z}}_j) = \left[\frac{\exp(-\sigma \|\tilde{\mathbf{z}}_j - \mathbf{c}_\ell\|^2)}{\sum_{i=1}^{L} \exp(-\sigma \|\tilde{\mathbf{z}}_j - \mathbf{c}_i\|^2)} \right]_{\ell \in [L]} \in \mathbb{R}^L \tag{6.3}$$

where (6.3) is typically referred to as the *softmax* function, a differentiable alternative to the max function. The hyperparameter σ controls the temperature of the softmax scores, with a lower σ yielding a more uniform distribution and a higher σ yielding a "peakier" distribution (i.e., $\sigma \to \infty \Rightarrow \phi(\tilde{z}_j) \to \max(\tilde{z}_j)$). Using the soft assignments, the latent vectors are quantized based on the codebook $\mathbf{C} \in \mathbb{R}^{m \times L}$,

$$Q(\tilde{\mathbf{z}}_j, \mathbf{C}) = \phi(\tilde{\mathbf{z}}_j)\mathbf{C}^T. \tag{6.4}$$

The quantized version of the latent vector is taken by reshaping $\mathbf{Q}(\tilde{\mathbf{Z}}, \mathbf{C}) \in \mathbb{R}^{r/m \times m}$ into $\hat{\mathbf{Z}} \in \mathbb{R}^d$, and the decoder produces the CSI estimates as $\hat{\mathbf{H}} = h(\hat{\mathbf{Z}}, \mathbf{C})$. An abstract illustration of an autoencoder using soft quantization can be seen in Figure 6.1.

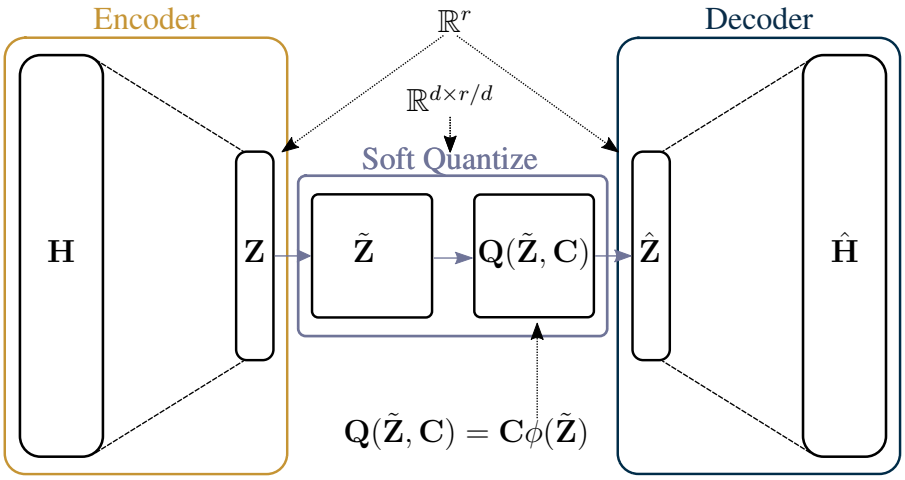


Figure 6.1: Abstract architecture for a CSI compression network with the 'SoftQuantize' layer ($Q(\tilde{\mathbf{Z}})$), a continuous, softmax-based relaxation of a d-dimensional quantization of the latent layer $\mathbf{Z}$.

References

[1] C. Wen, W. Shih, and S. Jin, "Deep Learning for Massive MIMO CSI Feedback," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, Oct 2018.

[2] Z. Liu, **M. del Rosario**, X. Liang, L. Zhang, and Z. Ding, "Spherical Normalization for Learned Compressive Feedback in Massive MIMO CSI Acquisition," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.

[3] T. Wang, C. Wen, S. Jin, and G. Y. Li, "Deep Learning-Based CSI Feedback Approach for Time-Varying Massive MIMO Channels," *IEEE Wireless Comm. Letters*, vol. 8, no. 2, pp. 416–419, April 2019.

[4] **M. del Rosario** and Z. Ding, "Learning-Based MIMO Channel Estimation under Spectrum Efficient Pilot Allocation and Feedback," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2022.

[5] E. Crespo Marques, N. Maciel, L. Naviner, H. Cai, and J. Yang, "A Review of Sparse Recovery Algorithms," *IEEE Access*, vol. 7, pp. 1300–1322, 2019.

[6] A. Goldsmith, S. A. Jafar, N. Jindal, and S. Vishwanath, "Capacity limits of mimo channels," *IEEE Journal on selected areas in Communications*, vol. 21, no. 5, pp. 684–702, 2003.

[7] H. Yang and T. L. Marzetta, "Performance of conjugate and zero-forcing beamforming in large-scale antenna systems," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 2, pp. 172–179, 2013.

[8] F. Kaltenberger, H. Jiang, M. Guillaud, and R. Knopp, "Relative Channel Reciprocity Calibration in MIMO/TDD Systems," in *2010 Future Network Mobile Summit*, June 2010, pp. 1–10.

[9] D. Mi, M. Dianati, L. Zhang, S. Muhaidat, and R. Tafazolli, "Massive mimo performance with imperfect channel reciprocity and channel estimation error," *IEEE Transactions on Communications*, vol. 65, no. 9, pp. 3734–3749, 2017.

[10] Q. Gao, F. Qin, and S. Sun, "Utilization of channel reciprocity in advanced mimo system," in *2010 5th International ICST Conference on Communications and Networking in China*, Aug 2010, pp. 1–5.

[11] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation," 3rd Generation Partnership Project (3GPP), TS 36.211, Jan. 2022. [Online]. Available: http://www.3gpp.org/dynareport/36211.htm

[12] ——, "NR; Physical Channels and Modulation," 3rd Generation Partnership Project (3GPP), TS 38.211, Jan. 2022. [Online]. Available: http://www.3gpp.org/dynareport/38211.htm

[13] M. B. Mashhadi and D. Gündüz, "Pruning the pilots: Deep learning-based pilot design and channel estimation for MIMO-OFDM systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6315–6328, 2021.

[14] F. Sohrabi, K. M. Attiah, and W. Yu, "Deep learning for distributed channel feedback and multiuser precoding in fdd massive mimo," *IEEE Transactions on Wireless Communications*, vol. 20, no. 7, pp. 4044–4057, 2021.

[15] C. Shepard, J. Ding, R. E. Guerra, and L. Zhong, "Understanding real many-antenna mu-mimo channels," pp. 461–467, 2016.

[16] X. Du and A. Sabharwal, "Massive mimo channels with inter-user angle correlation: Open-access dataset, analysis and measurement-based validation," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2021.

[17] C. Li, S. De Bast, E. Tanghe, S. Pollin, and W. Joseph, "Toward fine-grained indoor localization based on massive mimo-ofdm system: Experiment and analysis," *IEEE Sensors Journal*, vol. 22, no. 6, pp. 5318–5328, 2022.

[18] 3GPP, "3GPP TR 25.996 V6.1.0 (2003-09) Technical Report," 3rd Generation Partnership Project (3GPP), TS 25.996, 2003. [Online]. Available: http://www.3gpp.org/

[19] P. Kyösti *et al.*, "Winner ii channel models, dr1. 1.2," *available in: https://www. ist-winner. org/WINNER2-Deliverables/D1. 1.2. zip*, 2007.

[20] L. Liu, C. Oestges, J. Poutanen, K. Haneda, P. Vainikainen, F. Quitin, F. Tufvesson, and P. D. Doncker, "The COST 2100 MIMO Channel Model," *IEEE Wireless Communications*, vol. 19, no. 6, pp. 92–99, December 2012.

[21] B. Makki and T. Eriksson, "On hybrid arq and quantized csi feedback schemes in quasi-static fading channels," *IEEE transactions on communications*, vol. 60, no. 4, pp. 986–997, 2012.

[22] H. Shirani-Mehr and G. Caire, "Channel state feedback schemes for multiuser mimo-ofdm downlink," *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2713–2723, 2009.

[23] X. Rao and V. K. Lau, "Distributed compressive csit estimation and feedback for fdd multi-user massive mimo systems," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3261–3271, 2014.

[24] M. E. Eltayeb, T. Y. Al-Naffouri, and H. R. Bahrami, "Compressive sensing for feedback reduction in mimo broadcast channels," *IEEE Transactions on communications*, vol. 62, no. 9, pp. 3209–3222, 2014.

[25] T. T. Do, L. Gan, N. Nguyen, and T. D. Tran, "Sparsity adaptive matching pursuit algorithm for practical compressed sensing," in *2008 42nd Asilomar conference on signals, systems and computers*. IEEE, 2008, pp. 581–587.

[26] Z. Lu, J. Wang, and J. Song, "Multi-resolution csi feedback with deep learning in massive mimo system," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[27] M. Hussien, K. K. Nguyen, and M. Cheriet, "PRVNet: Variational autoencoders for massive MIMO CSI feedback," *arXiv*, 2020.

[28] Y. Sun, W. Xu, L. Fan, G. Y. Li, and G. K. Karagiannidis, "Ancinet: An efficient deep learning approach for feedback compression of estimated csi in massive mimo systems," *IEEE Wireless Communications Letters*, vol. 9, no. 12, pp. 2192–2196, 2020.

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-Resnet and the Impact of Residual Connections on Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[30] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end Optimized Image Compression," in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[31] J. Xie, L. Xu, and E. Chen, "Image Denoising and Inpainting with Deep Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 341–349, 2012.

[32] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[33] Z. Liu †, **M. del Rosario** †, and Z. Ding, "A Markovian Model-Driven Deep Learning Framework for Massive MIMO CSI Feedback," *IEEE Transactions on Wireless Communications*, vol. 21, no. 2, pp. 1214–1228, 2022.

[34] T. Hastie, R. Tibshiran, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2016. [Online]. Available: https://web.stanford.edu/~hastie/ElemStatLearn/

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[37] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *arXiv preprint arXiv:1710.09829*, 2017.

[38] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[39] K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[42] A. M. Sayeed, "Deconstructing multiantenna fading channels," *IEEE Transactions on Signal processing*, vol. 50, no. 10, pp. 2563–2579, 2002.

[43] Z. Liu, L. Zhang, and Z. Ding, "Exploiting Bi-Directional Channel Reciprocity in Deep Learning for Low Rate Massive MIMO CSI Feedback," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 889–892, 2019.

[44] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[45] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[46] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[47] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[48] E. Ogasawara, L. C. Martinez, D. De Oliveira, G. Zimbrão, G. L. Pappa, and M. Mattoso, "Adaptive normalization: A novel data normalization approach for non-stationary time series," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[49] S. Nayak, B. B. Misra, and H. S. Behera, "Impact of data normalization on stock index forecasting," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 6, no. 2014, pp. 257–269, 2014.

[50] X. Shao, "Self-normalization for time series: a review of recent developments," *Journal of the American Statistical Association*, vol. 110, no. 512, pp. 1797–1817, 2015.

[51] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Deep adaptive input normalization for time series forecasting," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3760–3765, 2019.

[52] R. Chopra, C. R. Murthy, and H. A. Suraweera, "On the Throughput of Large MIMO Beamforming Systems With Channel Aging," *IEEE Signal Processing Letters*, vol. 23, no. 11, pp. 1523–1527, 2016.

[53] K. Huber and S. Haykin, "Improved bayesian mimo channel tracking for wireless communications: incorporating a dynamical model," *IEEE transactions on wireless communications*, vol. 5, no. 9, pp. 2458–2466, 2006.

[54] K. S. Ali and P. Sampath, "Time domain channel estimation for time and frequency selective millimeter wave mimo hybrid architectures: Sparse bayesian learning-based kalman filter," *Wireless Personal Communications*, pp. 1–21, 2020.

[55] H. Kim, S. Kim, H. Lee, C. Jang, Y. Choi, and J. Choi, "Massive mimo channel prediction: Kalman filtering vs. machine learning," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 518–528, 2021.

[56] C. Lu, W. Xu, H. Shen, J. Zhu, and K. Wang, "MIMO Channel Information Feedback Using Deep Recurrent Network," *IEEE Commu. Letters*, vol. 23, no. 1, pp. 188–191, Jan 2019.

[57] Y. Liao, H. Yao, Y. Hua, and C. Li, "CSI Feedback Based on Deep Learning for Massive MIMO Systems," *IEEE Access*, vol. 7, pp. 86 810–86 820, 2019.

[58] X. Li and H. Wu, "Spatio-Temporal Representation With Deep Neural Recurrent Network in MIMO CSI Feedback," *IEEE Wireless Comm. Letters*, vol. 9, no. 5, pp. 653–657, 2020.

[59] Y. Jang, G. Kong, M. Jung, S. Choi, and I. Kim, "Deep Autoencoder Based CSI Feedback With Feedback Errors and Feedback Delay in FDD Massive MIMO Systems," *IEEE Wireless Comm. Letters*, vol. 8, no. 3, pp. 833–836, 2019.

[60] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges and et al., Eds., 2013, pp. 190–198.

[61] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *2nd International Conference on Learning Representations*, no. March 2014, 2014.

[62] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3104–3112, 2014.

[63] O. Irsoy and C. Cardie, "Opinion mining with deep recurrent neural networks," *Proc. 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 720–728, 2014.

[64] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009.

[65] H. Asplund, D. Astely, P. von Butovitsch, T. Chapman, M. Frenne, F. Ghasemzadeh, M. Hagström, B. Hogan, G. Jöngren, J. Karlsson, F. Kronestedt, and E. Larsson,

"Chapter 8 - 3GPP Physical Layer Solutions for LTE and the Evolution Toward NR," in *Advanced Antenna Systems for 5G Network Deployments*. Academic Press, 2020, pp. 301–350. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128200469000083

[66] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep ADMM-Net for Compressive Sensing MRI," in *Proceedings of the 30th international conference on neural information processing systems*, 2016, pp. 10–18.

[67] J. Zhang and B. Ghanem, "ISTA-Net: Interpretable Optimization-inspired Deep Network for Image Compressive Sensing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1828–1837.

[68] Y. Sun, W. Xu, L. Liang, N. Wang, G. Y. Li, and X. You, "A Lightweight Deep Network for Efficient CSI Feedback in Massive MIMO Systems," *IEEE Wireless Communications Letters*, 2021.

[69] L. Gan, "Block compressed sensing of natural images," in *2007 15th International Conference on Digital Signal Processing*, 2007, pp. 403–406.

[70] L. Kozachenko and N. N. Leonenko, "Sample Estimate of the Entropy of a Random Vector," *Problemy Peredachi Informatsii*, vol. 23, no. 2, pp. 9–16, 1987.

[71] T. M. Cover, *Elements of Information Theory*. John Wiley & Sons, 1999.

[72] Q. Yang, M. B. Mashhadi, and D. Gündüz, "Deep Convolutional Compression For Massive MIMO CSI Feedback," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019, pp. 1–6.

[73] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, 2003.

[74] M. B. Mashhadi, Q. Yang, and D. Gündüz, "Cnn-based analog csi feedback in fdd mimo-ofdm systems," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8579–8583.

[75] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. Van Gool, "Soft-to-hard Vector Quantization for End-to-end Learning Compressible Representations," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. NIPS, pp. 1142–1152, 2017.

[76] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

Appendix A

Computational Complexity of Common Layers

Based on the measures of computational complexity in Chapter 2 Section 2.1, i.e. FLOPs and parameters, we provide the corresponding formulas for common layers and operations used in the networks described in this dissertation. Note that any arithmetic operation (e.g., addition, multiplication) or non-linearity consumes a single FLOP, and certain non-linearities require a single parameter (e.g., the negative slope of a Leaky ReLU).

A.1 Matrix Multiplication

Denote two matrices $\mathbf{A} \in \mathbb{R}^{M \times P}$ and $\mathbf{B} \in \mathbb{R}^{P \times N}$. Matrix multiplication between $\mathbf{A}$ and $\mathbf{B}$ is denoted as

$$\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{M \times N},\ c_{ij} = \sum_{k=1}^{P} a_{ik} b_{kj}\ \forall\ i \in [1, \ldots, M], i \in [1, \ldots, N].$$

FLOPs: Each element in $\mathbf{C}$ involves P multiplications and $P - 1$ additions, and $\mathbf{C}$ includes $M \times N$ elements. Thus, the amount of FLOPs involved in matrix multiplication is

$$\text{FLOPs}_{\text{matmul}} = (2P - 1)MN$$

Parameters: The number of 'parameters' in matrix multiplicatiion depends on which matrix is considered the parameter matrix. If $\mathbf{A}$ is the parameter matrix, then the number of parameters is

$$P_{\text{matmul}} = MP.$$

Alternatively, if $\mathbf{B}$ is the parameter matrix, then the number of parameters is

$$P_{\text{matmul}} = PN.$$

A.2 Complex Matrix Multiplication

Denote two complex matrices $\mathbf{A} \in \mathbb{C}^{M \times P}$ and $\mathbf{B} \in \mathbb{C}^{P \times N}$. Complex matrix multiplication between $\mathbf{A}$ and $\mathbf{B}$ is denoted as

$$\begin{aligned}
\mathbf{C} &= \mathbf{AB} \in \mathbb{R}^{M \times N}, \\
c_{ij} &= \sum_{k=1}^{P} [\text{Re}(a_{ik})\text{Re}(b_{kj}) - \text{Im}(a_{ik})\text{Im}(b_{kj})] \\
&\quad + j[\text{Re}(a_{ik})\text{Im}(b_{kj}) + \text{Im}(a_{ik})\text{Re}(b_{kj})] \\
&\quad \forall\, i \in [1, \ldots, M], i \in [1, \ldots, N].
\end{aligned}$$

FLOPs: Each element in $\mathbf{C}$ involves P complex multiplications and $P - 1$ complex additions. Complex multiplication involves 4 (real) multiplications and 2 (real) additions, totaling 6 FLOPs total. Complex addition involves 2 (real) additions. Since $\mathbf{C}$ includes $M \times N$ complex elements, the amount of FLOPs involved in matrix multiplication is

$$\text{FLOPs}_{\text{matmul}} = (8P - 2)MN$$

Parameters: The number of parameters in complex matrix multiplication is identical to real matrix multiplication with a factor of 2. If $\mathbf{A}$ is the parameter matrix, then the number of parameters is

$$P_{\text{matmul}} = 2MP.$$

Alternatively, if $\mathbf{B}$ is the parameter matrix, then the number of parameters is

$$P_{\text{matmul}} = 2PN.$$

A.3 Linear Layer

Denote parameter matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ and the input vector $\mathbf{b} \in \mathbb{R}^{N}$. The output of a linear layer, $\mathbf{o} \in \mathbb{R}^{M}$, is given as

$$\mathbf{o} = \mathbf{Ab}.$$

If the linear layer contains a bias term, then an additional column $\mathbf{a}_0$ is appended to the end of the matrix $\mathbf{A}$ as

$$\mathbf{A}_{\text{bias}} = \begin{bmatrix} \mathbf{A} & \mathbf{a}_0 \end{bmatrix},$$

and correspondingly, the input vector is padded with a single one,

$$\mathbf{b}_{\text{bias}} = \begin{bmatrix} \mathbf{b} & 1 \end{bmatrix}.$$

FLOPs: A linear layer has the same number of FLOPs as a matrix-vector multiplication, i.e.

$$\text{FLOPs}_{\text{linear}} = M(N-1)$$

Parameters: The size of the matrix $\mathbf{A}$ determines the number of parameters in the layer, i.e.,

$$P_{\text{linear}} = MN$$

A.4 Convolutional Layer

In a convolutional layer, the complexity is driven by the number of kernels (N_k), the height/width of kernel (H_k/W_k), and the channels/height/width of the input data ($C/H/W$). Denote the input data as $\mathbf{I} \in \mathbb{R}^{C\times H\times W}$, the kernel as $\mathbf{K} \in \mathbb{R}^{N_k\times H_k\times W_k}$, and the output image as $\mathbf{O} \in \mathbb{R}^{N_k\times H\times W}$. The convolution operation (with no bias term) assigns a value to each element of the output using the following equation,

$$\mathbf{O}(i,j,k) = \sum_{c=1}^{C}\sum_{h=1}^{H_k}\sum_{w=1}^{W_k} K(i,j+h,k+w)I\left(c, j+h-\left\lfloor\frac{H}{2}\right\rfloor, k+w-\left\lfloor\frac{W}{2}\right\rfloor\right), \quad \text{(A.1)}$$

where the i,j,k index over the output channels, rows, and columns (respectively). On the right-hand side of (A.1), the first summation $\left(\sum_{c=1}^{C}\right)$ sums over the channels of the input image while the following two summations $\left(\sum_{h=1}^{H_k} / \sum_{w=1}^{W_k}\right)$ sum over the height/width of the kernel. Whenever the height or width indices of $\mathbf{I}$ are out of the range $[1,\ldots,H]$ or $[1,\ldots,W]$, the elements involved in the convolution are determined by the padding (e.g., zero padding, reflected padding, etc.).

FLOPs: The number of FLOPs in a convolutional layer is given by the following formula,

$$\text{FLOPs}_{\text{conv}} = C\times H\times W\times N_k\times H_k\times W_k$$

Parameters: The number of parameters in a convolutional layer is given by the following formula,

$$P_{\text{conv}} = N_k\times H_k\times W_k$$

A.5 Soft Threshold Function

The soft threshold function used in ISTANet+ [67] is given in (4.10), replicated below as,

$$\text{soft}(x,\theta) = \text{sign}(x)\text{ReLU}(|x|-\theta),$$

where θ is the threshold function.

FLOPs: We consider a single soft threshold to consume one FLOP.

Parameters: The soft threshold function requires one parameter to be stored, θ.

Appendix B

Autoregressive Markov Models

In Chapter 3 Section 3.2, we introduced a differential encoding network based on an autoregressive Markov model that was A) one-step and B) scalar. In this appendix, we show how we can generalize both the number of steps and the dimension of the Markov model.

Recall the truncated delay domain CSI at the t-th timeslot, $\mathbf{H}_t \in \mathbb{C}^{R_d \times N_b}$. Rather than the one-step, scalar model $\hat{\gamma} \in \mathbb{R}^+$ (see (3.5) in Section 3.2), we can derive a p-step, multivariate predictor. A p-step autoregressive model for $\mathbf{H}_t$ can be written generally as a function of p previous timeslots,

$$\hat{\mathbf{H}}_t = f(\mathbf{H}_{t-1}, \mathbf{H}_{t-2}, \ldots, \mathbf{H}_{t-p}).$$

Given p prior CSI samples, the mean-square optimal predictor $\hat{\mathbf{H}}_t$ is a linear combination of these the prior CSI samples,

$$\hat{\mathbf{H}}_t = \mathbf{H}_{t-1}\mathbf{W}_1 + \cdots + \mathbf{H}_{t-p}\mathbf{W}_p + \mathbf{E}_t. \tag{B.1}$$

Where $\mathbf{W}_i$ is the coefficient matrix for the i-th timeslot with dimension $\mathbb{C}^{N_b \times N_b}$, and $\mathbf{E}_t$ is the error term at time t, which is uncorrelated with the CSI samples (i.e. $\mathbf{H}_{t-i}^H \mathbf{E}_t = 0$ for all $i \in [0, \ldots, p]$). To solve for the matrices $\mathbf{W}_{t-1}, \ldots, \mathbf{W}_{t-p}$, we first pre-multiply by $\mathbf{H}_{t-i}^H$,

$$\begin{aligned} \mathbf{H}_{t-i}^H \hat{\mathbf{H}}_t &= \mathbf{H}_{t-i}^H \mathbf{H}_{t-1}\mathbf{W}_1 + \cdots + \mathbf{H}_{t-i}^H \mathbf{H}_{t-p}\mathbf{W}_p + \mathbf{H}_{t-i}^H \mathbf{E}_t \\ &= \mathbf{H}_{t-i}^H \mathbf{H}_{t-1}\mathbf{W}_1 + \cdots + \mathbf{H}_{t-i}^H \mathbf{H}_{t-p}\mathbf{W}_p. \end{aligned} \tag{B.2}$$

Denote the correlation matrix $\mathbf{R}_i = \mathbb{E}[\mathbf{H}_{t-i}^H \mathbf{H}_t]$. If temporal coherence between timeslots is maintained, then we may assume that CSI matrices arise from a stationary process, implying the following properties:

(1) $\mathbf{R}_i = \mathbb{E}[\mathbf{H}_{t-i}^H \mathbf{H}_t] = \mathbb{E}[\mathbf{H}_t^H \mathbf{H}_{t+i}]$

(2) $\mathbf{R}_i = \mathbf{R}_{-i}^H$

With these properties in mind, we take the expectation of (B.2), resulting in a linear

combination of $\mathbf{R}_i$ matrices,

$$\mathbb{E}\left[\mathbf{H}_{t-i}^H \hat{\mathbf{H}}_t\right] = \mathbb{E}\left[\mathbf{H}_{t-i}^H \mathbf{H}_{t-1}\mathbf{W}_1\right] + \cdots + \mathbb{E}\left[\mathbf{H}_{t-i}^H \mathbf{H}_{t-p}\mathbf{W}_p\right]$$
$$\mathbf{R}_{i+1} = \mathbf{R}_i\mathbf{W}_1 + \cdots + \mathbf{R}_{i-p+1}\mathbf{W}_p.$$

For p CSI samples (i.e., for $i \in [0, 1, \ldots, p-1]$), we write a system of p equations, admitting the following,

$$\begin{bmatrix}\mathbf{R}_1\\ \mathbf{R}_2\\ \cdots\\ \mathbf{R}_p\end{bmatrix} = \begin{bmatrix}\mathbf{R}_0 & \mathbf{R}_1^H & \ldots & \mathbf{R}_{p-1}^H\\ \mathbf{R}_1 & \mathbf{R}_0 & \ldots & \mathbf{R}_{p-2}^H\\ \vdots & & \ddots & \vdots\\ \mathbf{R}_{p-1} & \mathbf{R}_{p-2} & \ldots & \mathbf{R}_0\end{bmatrix}\begin{bmatrix}\mathbf{W}_1\\ \mathbf{W}_2\\ \cdots\\ \mathbf{W}_p\end{bmatrix}. \tag{B.3}$$

Finally, we solve for the coefficient matrices by inverting the Toeplitz matrix,

$$\begin{bmatrix}\mathbf{W}_1\\ \mathbf{W}_2\\ \cdots\\ \mathbf{W}_p\end{bmatrix} = \begin{bmatrix}\mathbf{R}_0 & \mathbf{R}_1^H & \ldots & \mathbf{R}_{p-1}^H\\ \mathbf{R}_1 & \mathbf{R}_0 & \ldots & \mathbf{R}_{p-2}^H\\ \vdots & & \ddots & \vdots\\ \mathbf{R}_{p-1} & \mathbf{R}_{p-2} & \ldots & \mathbf{R}_0\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{R}_1\\ \mathbf{R}_2\\ \cdots\\ \mathbf{R}_p\end{bmatrix}. \tag{B.4}$$

Thus, the solution to multi-step, multivariate Markov model is a function of the correlation matrices.

Since the distributions of $\mathbf{H}_i$ are not known, we cannot calculate the expectations in the correlation matrices, $\mathbf{R}_i$. Instead, we estimate the sample correlation matrices with the training data, i.e.,

$$\hat{\mathbf{R}}_i = \frac{1}{N_{\text{train}}}\sum_{j}^{N_{\text{train}}} \mathbf{H}_{t-i}(j)^H \mathbf{H}_t(j),$$

where j indexes over the training data. Now, we write the estimators, $\hat{\mathbf{W}}_i$, with respect to the sample correlation matrices,

$$\begin{bmatrix}\hat{\mathbf{W}}_1\\ \hat{\mathbf{W}}_2\\ \cdots\\ \hat{\mathbf{W}}_p\end{bmatrix} = \begin{bmatrix}\hat{\mathbf{R}}_0 & \hat{\mathbf{R}}_1^H & \ldots & \hat{\mathbf{R}}_{p-1}^H\\ \hat{\mathbf{R}}_1 & \hat{\mathbf{R}}_0 & \ldots & \hat{\mathbf{R}}_{p-2}^H\\ \vdots & & \ddots & \vdots\\ \hat{\mathbf{R}}_{p-1} & \hat{\mathbf{R}}_{p-2} & \ldots & \hat{\mathbf{R}}_0\end{bmatrix}^{-1}\begin{bmatrix}\hat{\mathbf{R}}_1\\ \hat{\mathbf{R}}_2\\ \cdots\\ \hat{\mathbf{R}}_p\end{bmatrix}. \tag{B.5}$$

If a more simple model is desired, then we can construct scalar estimators for each timeslot. Denote the mean of all elements in the correlation matrix as,

$$r_i = \mathbb{E}\left[\sum_{k}^{R_d}\sum_{\ell}^{N_b} \mathbf{R}_i^{(k,\ell)}\right]$$

where $\mathbf{R}_i^{(k,\ell)}$ is the element from the k-th row and ℓ-th column of $\mathbf{R}_i$. The resulting system of equations for scalar model is given as

$$\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \cdots \\ \gamma_p \end{bmatrix} = \begin{bmatrix} r_0 & r_1^* & \cdots & r_{p-1}^* \\ r_1 & r_0 & \cdots & r_{p-2}^* \\ \vdots & & \ddots & \vdots \\ r_{p-1} & r_{p-2} & \cdots & r_0 \end{bmatrix}^{-1} \begin{bmatrix} r_1 \\ r_2 \\ \cdots \\ r_p \end{bmatrix}. \tag{B.6}$$

Following the same logic as before, we utilize the sample versions of $\hat{r}_i$,

$$\hat{r}_i = \frac{1}{N_{\text{train}}} \sum_{j}^{N_{\text{train}}} \sum_{k}^{R_d} \sum_{\ell}^{N_b} \hat{\mathbf{R}}_i^{(k,\ell)}(j),$$

which admits the system of equations,

$$\begin{bmatrix} \hat{\gamma}_1 \\ \hat{\gamma}_2 \\ \cdots \\ \hat{\gamma}_p \end{bmatrix} = \begin{bmatrix} \hat{r}_0 & \hat{r}_1^* & \cdots & \hat{r}_{p-1}^* \\ \hat{r}_1 & \hat{r}_0 & \cdots & \hat{r}_{p-2}^* \\ \vdots & & \ddots & \vdots \\ \hat{r}_{p-1} & \hat{r}_{p-2} & \cdots & \hat{r}_0 \end{bmatrix}^{-1} \begin{bmatrix} \hat{r}_1 \\ \hat{r}_2 \\ \cdots \\ \hat{r}_p \end{bmatrix}. \tag{B.7}$$

For the one-step scalar estimator, we can derive equation (3.3) from Chapter 3 of Section 3.2,

$$\begin{aligned} \hat{\gamma}_1 &= \frac{\hat{r}_1}{\hat{r}_0} \\ &= \frac{\text{Trace}(\hat{\mathbf{R}}_1}{\sum_k^{R_d} \sum_l^{N_b} \hat{\mathbf{R}}_0(k,l)}. \end{aligned}$$

Appendix C

Matrix Regularization

In this dissertation, we encounter a few situations where we have a matrix to invert, $\mathbf{A}$, but the matrix $\mathbf{A}$ may be nearly singular. For example, in Section 4.1 of Chapter 4 we had the pseudoinverse $\mathbf{Q}_i^{\#} = \mathbf{Q}_i^T \left(\mathbf{Q}_i \mathbf{Q}_i^T\right)^{-1}$, where we would denote our matrix to invert as $\mathbf{A} = \mathbf{Q}_i \mathbf{Q}_i^T$.

As another example from Appendix B, we discussed the multivariate, p-step Markov model for CSI which was a function of the correlation matrices, $\hat{\mathbf{R}}_i$. Per equation (B.5), we denote $\mathbf{A}$ as the Toeplitz matrix populated by the $\hat{\mathbf{R}}_i$ matrices.

In either of the cases described above, the stability of the matrix inverse can be improved by regularizing the matrix. Here, we briefly describe one method for regularizing nearly singular matrices, off-diagonal regularization (ODIR).

Denote A_{ij} as the element in the i-th row and j-th column of the matrix $\mathbf{A}$. We select a non-negative real scaling factor $\delta \in \mathbb{R}^+$ to scale down the off-diagonal elements of $\mathbf{R}$. The elements of the resulting ODIR matrix, $\mathbf{A}_{\text{ODIR}}$, are written as

$$A_{ij,\text{ODIR}} = \begin{cases} A_{ij} & \text{if } i = j \\ \frac{A_{ij}}{1+\delta} & \text{if } i \neq j. \end{cases} \tag{C.1}$$

Appendix D

Compressive sensing

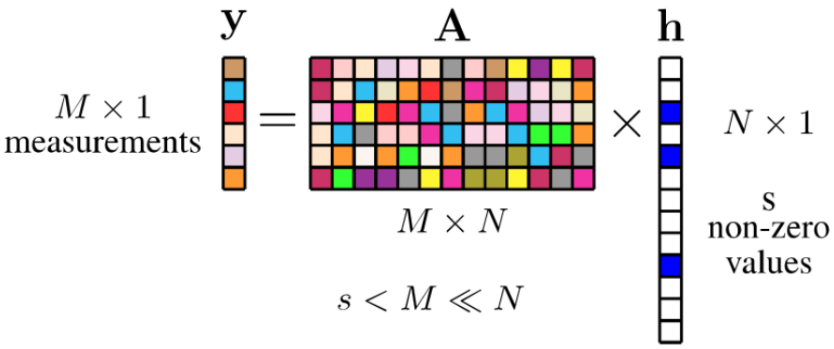


Figure D.1: Illustration of sampling/measurement process in compressive sensing [5]. The notation used here differs slightly from the notation adopted in this appendix (i.e., $\mathbf{A}$ and $\mathbf{h}$ in the figure correspond to $\mathbf{\Phi}$ and $\mathbf{x}$, respectively)

.

In Section 4.2 of Chapter 4, we introduced a heterogeneous differential encoder which utilized deep compressive sensing networks. This appendix provides a brief overview of the relevant theory from compressive sensing based on the excellent survey [5].

Given a signal $\mathbf{x} \in \mathbb{R}^N$, denote a random measurement of this signal as

$$\mathbf{y} = \mathbf{\Phi}\mathbf{x}$$

where $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ is referred to as the *measurement matrix* and $\mathbf{y} \in \mathbb{R}^M$ is a low-dimensional measurement (i.e., $M << N$). The goal of compressive sensing is to recover the original signal $\mathbf{x}$ based on the low-dimensional measurement $\mathbf{y}$. The least-squares solution to this problem is given as

$$\min_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}}\|_2 \text{ subject to } \|\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}\|_2^2 < \epsilon \tag{D.1}$$

where $\epsilon > 0$ is some error tolerance. By construction, the matrix Φ has more columns than rows (i.e., $N >> M$), and consequently, the linear system (D.1) is underdetermined. Furthermore, the least-squares solution cannot return a sparse vector, and instead, the recovery

of $\mathbf{x}$ is typically framed as a sparsity-constrained least-squares estimation problem, i.e.

$$\min_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}}\|_0 \text{ subject to } \|\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}\|_2^2 < \epsilon \tag{D.2}$$

Under certain constraints, the original signal $\mathbf{x}$ can be perfectly reconstructed. However, this perfect reconstruction requires a combinatoric search over $\binom{N}{s}$ (where s is the number of non-zero elements in $\mathbf{x}$). Instead, the problem is often relaxed to use the ℓ_1 norm,

$$\min_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}}\|_1 \text{ subject to } \|\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}\|_2^2 < \epsilon, \tag{D.3}$$

which is referred to as the least absolute shrinkage and selection operation (LASSO).

D.1 The ISTA algorithm

To solve the LASSO, it is possible to use proximal gradient methods from convex optimization [76]. The gradient method that we focus on in this appendix is the iterative shrinkage threshold algorithm (ISTA), which is the namesake of the ISTANet algorithm discussed in Section 4.2 of Chapter 4. For the ℓ_1 regularized problem, gradient-based methods solve

$$\min \{f(\mathbf{x}) + \lambda\|\mathbf{x}\|_1\}$$

using the iterative steps

$$\begin{aligned}\mathbf{x}_k &= \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}_{k-1}) + \langle \mathbf{x} - \mathbf{x}_{k-1}, \nabla f(\mathbf{x}_{k-1})\rangle + \frac{1}{2t_k}\|\mathbf{x} - \mathbf{x}_{k-1}\|^2 + \lambda\|\mathbf{x}\|_1 \right\} \\ &= \arg\min_{\mathbf{x}} \left\{ \frac{1}{2t_k}\|\mathbf{x} - (\mathbf{x}_{k-1} - t_k\nabla f(\mathbf{x}_{k-1}))\|^2 + \lambda\|\mathbf{x}\|_1 \right\},\end{aligned}$$

where $t_k > 0$ is the stepsize for the algorithm. Note that the second line is admitted by ignoring the constant terms. The ℓ_1 term is separable, and consequently, computing the iteration $\mathbf{x}_k$ can be done by solving a one-dimensional minimization problem for each component of $\mathbf{x}_k$,

$$\mathbf{x}_k = \mathcal{T}_{\lambda t_k}(\mathbf{x}_{k-1} - t_k\nabla f(\mathbf{x}_{k-1})).$$

Note the shrinkage operator, $\mathcal{T}_\alpha : \mathbb{R}^n \to \mathbb{R}^n$,

$$\mathcal{T}_\alpha(\mathbf{x})_i = \text{soft}(x_i, \alpha) = (|x_i| - \alpha)_+(x_i)$$

When this gradient method is applied to the compressive sensing problem, we have $f(\mathbf{x}) := \|\mathbf{\Phi}\mathbf{x} - \mathbf{y}\|$, and the resulting iterative algorithm is known as ISTA, where the iterations take the form,

$$\mathbf{x}_k = \mathcal{T}_{\lambda t_k}(\mathbf{x}_{k-1} - t_k\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}_{k-1} - \mathbf{y})) \tag{D.4}$$

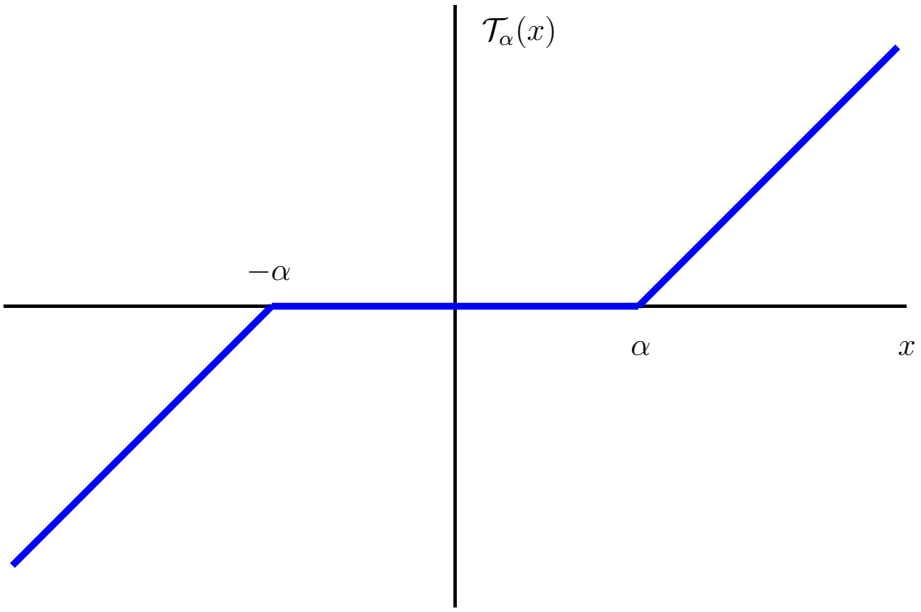


Figure D.2: Soft-threshold function used in the ISTA algorithm.

These iterations of the ISTA algorithm correspond to the gradient step and the proximal step of (4.8) and (4.9) for ISTANet+ (see Chapter 4, Section 4.2). Note that ISTANet+ differs from the vanilla formulation of ISTA in a few key ways, namely:

- **Learnable sparse basis**: ISTANet+ assumes that the ℓ_1 penalty term is imposed on a sparse transform of the input data (i.e., $\|\mathbf{\Psi x}\|_1$). Furthermore, the sparse transform and its inverse (i.e., $\mathbf{\Psi}$ and $\mathbf{\Psi}^{-1}$) are learned using convolutional layers (i.e., $\mathcal{G}$ and $\tilde{\mathcal{G}}$), and the 'soft' operator is applied to sparse transform.
- **Residual connections**: ISTANet+ uses the sparse transform and the 'soft' operator on the residual of the estimate rather than the estimate itself.