Efficient Deep Learning for Massive MIMO Channel State Estimation

Mason del Rosario

University of California, Davis

Davis, CA 95616

mdelrosa@ucdavis.edu

May 27, 2021

Contents

1 Introduction **4**

1.1 MIMO Channel Overview 4

1.2 Channel Model 6

1.3 Classical CSI Estimation 7

1.4 Deep Learning/CNNs for CSI Estimation 7

2 Spherical Normalization **10**

2.1 Related Work 12

2.2 Methods 12

2.2.1 Spherical Normalization 12

2.2.2 CsiNet-Pro 14

2.3 Results 15

3 Deep Differential Encoding **16**

3.1 Related Work 16

3.2 Methods 18

3.2.1 MarkovNet 19

3.3 Numerical Results 21

4 CsiNet-SoftQuant **23**

4.1 Related Work 24

4.2 A Vector-quantized Autoencoder for Trainable Codewords 25

4.2.1 Entropy-regularization 26

4.2.2 Softmax annealing 27

4.3 Entropy Estimation for CSI Matrices 28

4.3.1 Entropy Estimation of Quantized CSI 28

4.3.2 Differential Entropy Estimation of Quantized CSI 29

4.3.3 Rate-Distortion Estimation 29

4.4 Results 30

4.4.1 Results: Rate-Distortion 31

4.4.2 Results: CSI Entropy Estimation 32

4.5 Future Work: Rate-optimal Quantization 35

4.5.1 ROI CSI Compression 35

4.5.2 Differential Encoding with Trainable Feedback Quantization 36

5 Conclusion **37**

Bibliography **38**

Chapter 1

Introduction

This qualifying examination proposal details work in improving the accuracy and efficiency of deep learning methods for MIMO channel state information estimation. Section 1.1 provides an overview of the MIMO channel. Section 1.2 introduces the channel simulation used in this work, the COST2100 model. Section 1.3 discusses prior work in compressed sensing for CSI estimation. Section 1.4 provides a generic overview of deep learning and recent work in deep learning for CSI estimation in MIMO networks. 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 received

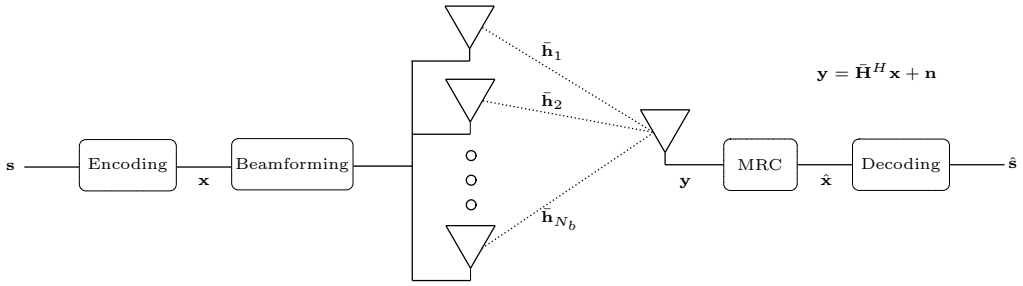


Figure 1.1: Example multi-antenna transmitter (BS, gNB) and single-antenna user equipment (UE) and relevant system values.

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
$\mathbf{H}_d$	$\mathbb{C}^{N_f \times N_b}$	Downlink impulse response on m-th subcarrier
$\mathbf{w}_{t,m}$	$\mathbb{C}^{N_b \times 1}$	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

symbols on the m-th subcarrier for the downlink and the uplink 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 $\bar{\mathbf{H}}_d$ [1]. 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 [2]. Downlink CSI estimation can be performed in time division duplex

(TDD) by using uplink pilots due to channel reciprocity [3–5]. In contrast, frequency domain duplex (FDD) does not admit channel reciprocity due to frequency-selective channels, and CSI estimates must be acquired using feedback.

Given their dimensionality, feeding back entire CSI matrices is impractical. Instead, we seek a compressed representation of a sparse transformation. The sparse representation we consider is the angular-delay representation of CSI matrices [6]. Denote the unitary DFT (inverse DFT) matrix $\mathbf{F} \in \mathbb{C}^{N_f \times N_f}$ ($\mathbf{F}^H \in \mathbb{C}^{N_b \times N_b}$), and denote the spatial-frequency CSI matrix as $\bar{\mathbf{H}}$. The angular-delay domain representation $\mathbf{H}$ is given as

$$\mathbf{H} = \mathbf{F}^H \bar{\mathbf{H}} \mathbf{F}.$$

The delay spread of the resulting $\mathbf{H}$ can typically be captured with a small number of delay elements, so we restrict our attention to the first R_d elements of $\mathbf{H}$, resulting in a truncated angular-delay matrix which we denote as $\mathbf{H}_d \in \mathbb{C}^{(R_d \times N_b)}$ for the downlink channel state.

1.2 Channel Model

For all CSI tests, we mainly rely on the COST2100 MIMO channel model [7]. 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 20m×20m coverage area.
2. **Outdoor** channels using a 300MHz downlink at 0.9 m/s UE velocity served by a gNB at center of a 400m×400m coverage area.

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

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

1.3 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 [8, 9]. The second category includes works which use compressed sensing, a technique which applies a random measurement matrix at the transmitter and the receiver [10,11]. Compressed sensing assumes matrices to be encoded and fed back meet certain sparsity requirements, and compressed sensing algorithms require iterative solvers [12] 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. Before describing these works, we first describe a few pertinent concepts from deep learning.

1.4 Deep Learning/CNNs for CSI Estimation

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 [13] for an overview).

Convolutional Neural Networks: 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., [14]), and given enough layers, such networks can achieve arbitrarily accurate functional approximation [15]. In recent years, neural networks with convolutional layers have established state-of-the-art performance in computer vision tasks such as image classification [16] and segmentation [17].

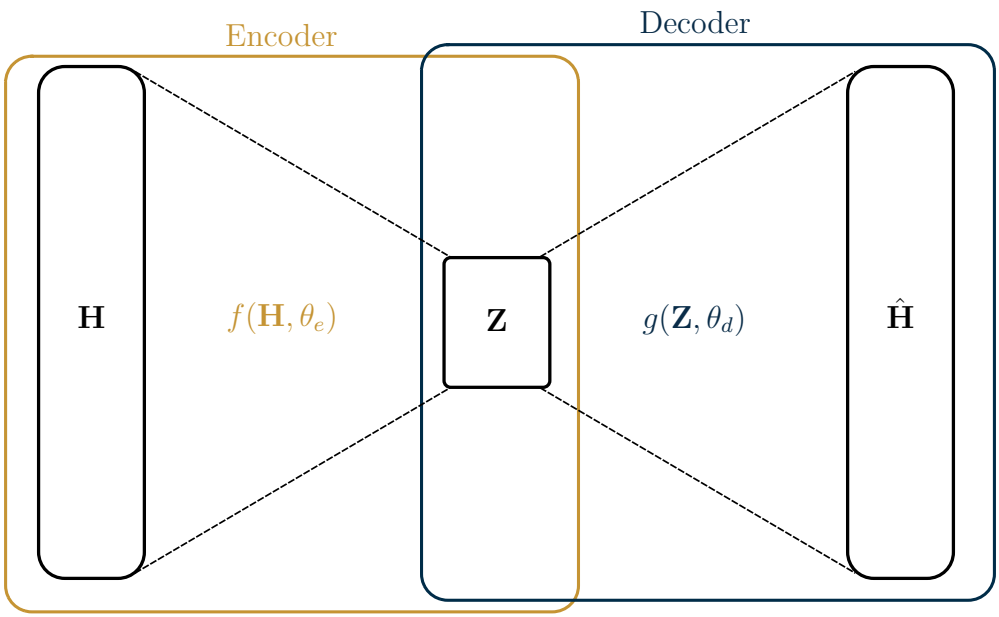


Figure 1.2: Abstract schematic for an autoencoder operating on CSI matrices $\mathbf{H}$. The encoder learns a latent representation, $\mathbf{Z}$, while the decoder learns to reconstruct estimates $\hat{\mathbf{H}}$.

A common architecture for deep unsupervised learning is the *autoencoder* (see Fig. 1.2 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,

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

We optimize network parameters $\vec{\theta_e}, \vec{\theta_d}$ by backpropagation and a stochastic optimization algorithm (e.g., stochastic gradient descent, ADAM).

Successful efforts in DL for CSI estimation have typically utilized convolutional neural networks (CNNs) in an autoencoder structure [18]. Variations on the CNN-based autoencoder have investigated different network architectures [19], variational training frameworks [20], and denoising modules [21]. Other works have exploited physical channel characteristics such as downlink/uplink reciprocity [22] and temporal coherence [23]. To make CNNs practical for actual feedback transmission, authors have experimented with networks using entropy encoding for bit stream feedback [24] and networks with complex latent elements for IQ symbol feedback [25].

Chapter 2

Spherical Normalization

Most work in deep learning for CSI estimation focuses on different neural network architectures, training frameworks, or hyperparameter tuning. Such works treat the real and imaginary elements of $\mathbf{H}$ as separate channels similar to color channels in images. The normalization method used in these works is typically the same – the extrema (i.e., the minimum and the maximum) of the real and imaginary channels are used to perform minmax scaling over the entire dataset. For the scalar $H_n(i,j)$, the minmax-scaled version of this element is

$$H_{n,\text{minmax}}(i,j) = \frac{H_n(i,j) - H_{\min}}{H_{\max} - H_{\min}},$$

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. 2.1).

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 (see Fig. 2.2) tends to have much lower variance than that of image data. 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}.

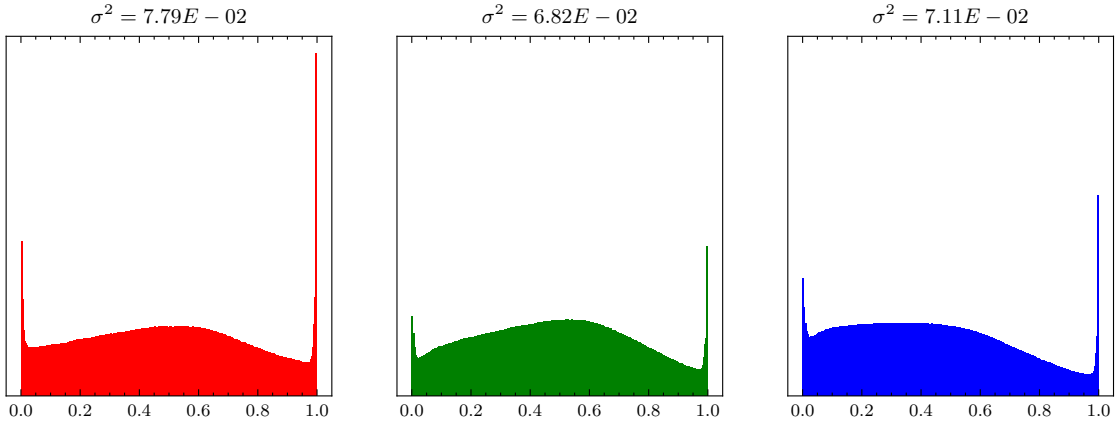


Figure 2.1: Distribution and variance of minmax-normalized ImageNet color channels ($N = 50000$) images.

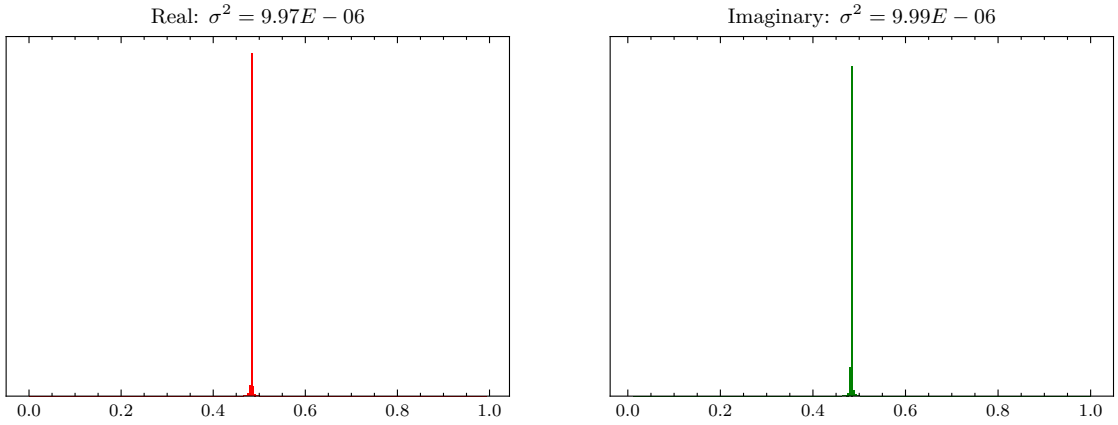


Figure 2.2: Distribution and variance of minmax-normalized COST2100 real/imaginary channels ($N = 99000$) images.

2.1 Related Work

Several works have investigated normalization techniques for deep learning such as batch normalization [26], instance normalization [27], layer normalization [28], and group normalization [29]. These normalization techniques scale the outputs of latent layers in neural networks, which helps to solve the problem of covariate shift [26] where the mean and variance of 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 [30–32]. In [33], 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.

2.2 Methods

Here, we discuss our work in spherical normalization (Section 2.2.1) and our optimized network architecture (Section 2.2.2) [34].

2.2.1 Spherical Normalization

Rather than apply minmax normalization, which is adversely impacted by outliers, 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

$$z = \frac{x - \mu}{\sigma^2}. \tag{2.1}$$

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 k-th downlink CSI matrix of the dataset as $\mathbf{H}_d^k$. 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[1] 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. 2.3 with Fig. 2.2).

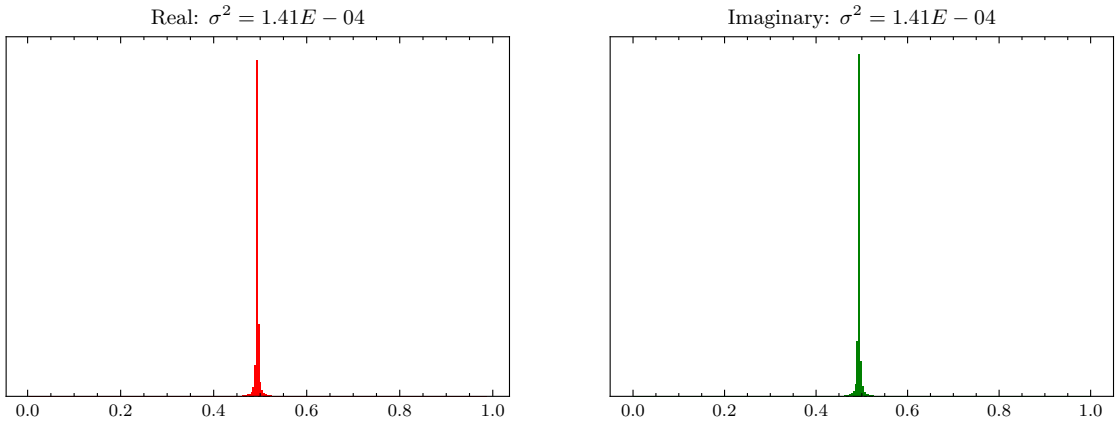


Figure 2.3: Distribution and variance of COST2100 real/imaginary channels under spherical normalization ($N = 99000$) images.

Beyond desirable properties in the input distribution, spherical normalization also results in an objective function which is better matched with the evaluation criterion. Neural

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

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 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.

2.2.2 CsiNet-Pro

In [34], 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 [18]).

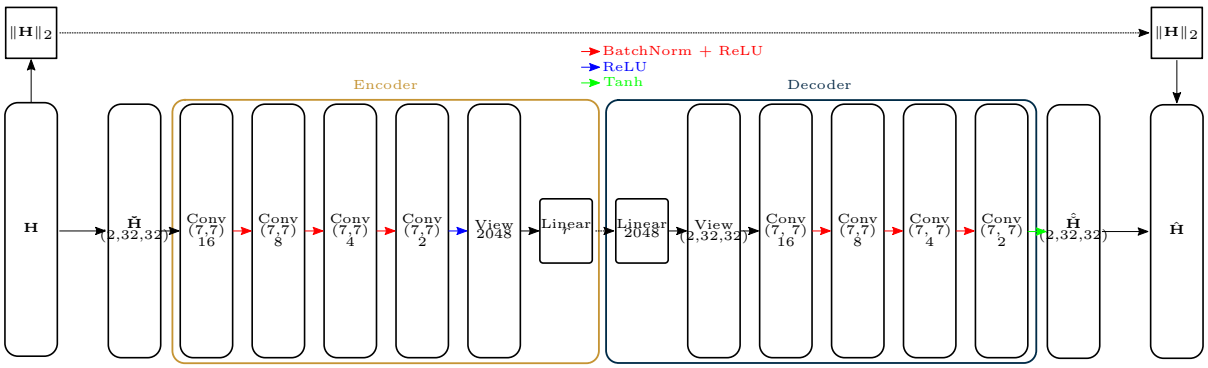
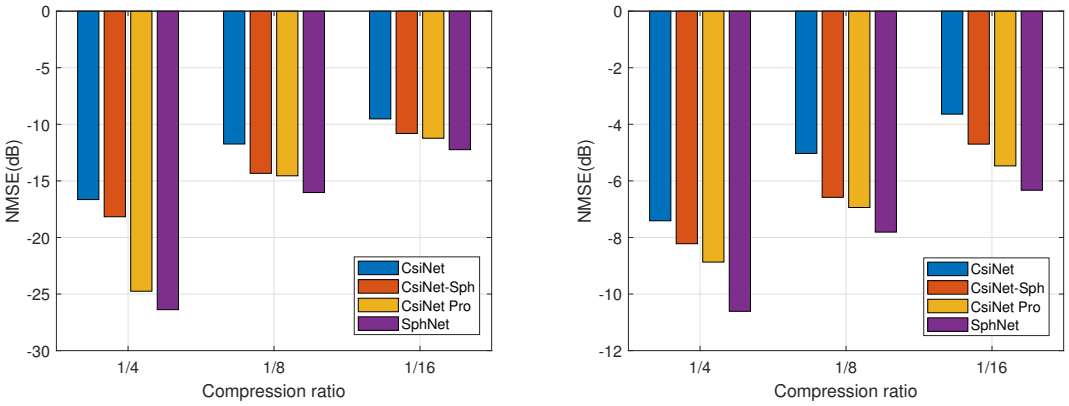


Figure 2.4: SphNet – CsiNetPro architecture with Spherical Normalization.

2.3 Results

Training on spherically normalized data and optimizing with respect to NMSE can yield better accuracy. Fig. 2.5 demonstrates this improvement for CsiNet and CsiNet-Pro on the 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 2.5: Reconstruction error for CsiNet [18] and CsiNet Pro with and without spherical normalization. SphNet combines CsiNet Pro with spherical normalization [34].

Chapter 3

Deep Differential Encoding

In this section, we consider methods for exploiting temporal correlation between CSI of subsequent timeslots. Assuming the channel coherence within a certain window of time, a reasonably accurate CSI estimate at time $t-1$ can be used to estimate the CSI at time t. Generically, we can write this estimator as

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

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

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

3.1 Related Work

Prior work in temporal correlation for CSI estimation utilized state-space methods such as the Kalman filter [35–37]. 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 [23, 38–41]. 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 [42] and can be used to predict future states of the process [43].

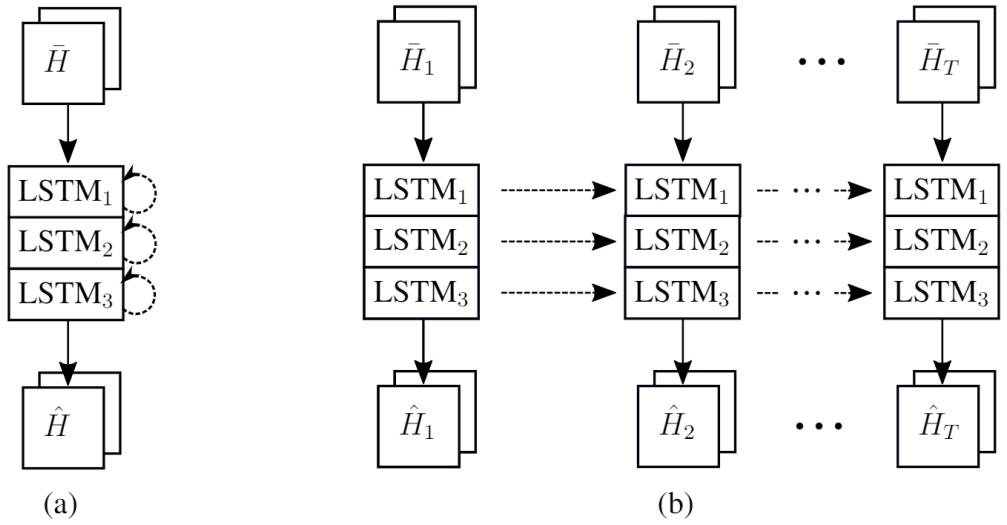


Figure 3.1: An example of LSTMs used for CSI estimation. (a) "Stacked" LSTM network of depth 3 shown with recurrent connections. (b) Same LSTM network "unrolled" into T timeslots

RNNs have been used extensively in natural language processing (NLP) for machine translation [44] and sentiment extraction [45]. For such works in NLP, authors have empirically found "stacked" or "deep" RNNs to be effective (e.g., Fig. 3.1), hypothesizing that having multiple recurrent layers allows the network to extract different semantic timescales [45, 46]. Works in CSI estimation have taken cues from this work in NLP, proposing CSI estimation networks with stacked LSTMs after a sequence of autoencoders [23]. While such work has demonstrated the utility of RNNs, the computational cost of LSTMs can be pro-

hibitively high. For example, the RNN portion of the network proposed in [23] accounts for 10^8 additional parameters. Since channel estimation should not place an undue computational burden on the communications system, LSTMs can be problematic.

3.2 Methods

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,

$$\grave{\mathbf{H}}_t = \hat{\mathbf{H}}_{t-1}\mathbf{W}$$

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

$$\mathbf{H}_t = \mathbf{H}_{t-1}\mathbf{W} + \mathbf{E}_t$$
$$\mathbf{H}_{t-1}^H\mathbf{H}_t = \mathbf{H}_{t-1}^H\mathbf{H}_{t-1}\mathbf{W} + \mathbf{H}_{t-1}^H\mathbf{E}_t$$

Under the principle of orthogonality, the error term $\mathbf{E}_t$ is orthogonal with the observed data, and the product $\mathbf{H}_{t-1}^H\mathbf{E}_t$ becomes a zero matrix. Denoting the cross correlation matrix as $\mathbf{R}_i = \mathbb{E}\left[\mathbf{H}_{t-i}^H\mathbf{H}_t\right]$.

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

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

$$\hat{\mathbf{R}}_i = \frac{1}{N}\sum_j^N \mathbf{H}_{t-i}^H(j)\mathbf{H}_t(j),$$

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

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

We can further simplify this estimator to a scalar, $\gamma \in \mathbb{R}$, as

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

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

$$\grave{\mathbf{H}}_t = \hat{\gamma}\hat{\mathbf{H}}_{t-1}. \tag{3.3}$$

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

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

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

$$\hat{\mathbf{H}}_t = \hat{\gamma}\hat{\mathbf{H}}_{t-1} + \hat{\mathbf{E}}_t \tag{3.4}$$

3.2.1 MarkovNet

In [47], 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 ($\mathbf{H}_1$). For all subsequent timeslots, t_i for $i \geq 2$, we use the MMSE estimator (3.3) 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 (3.4) to produce a refined estimate.

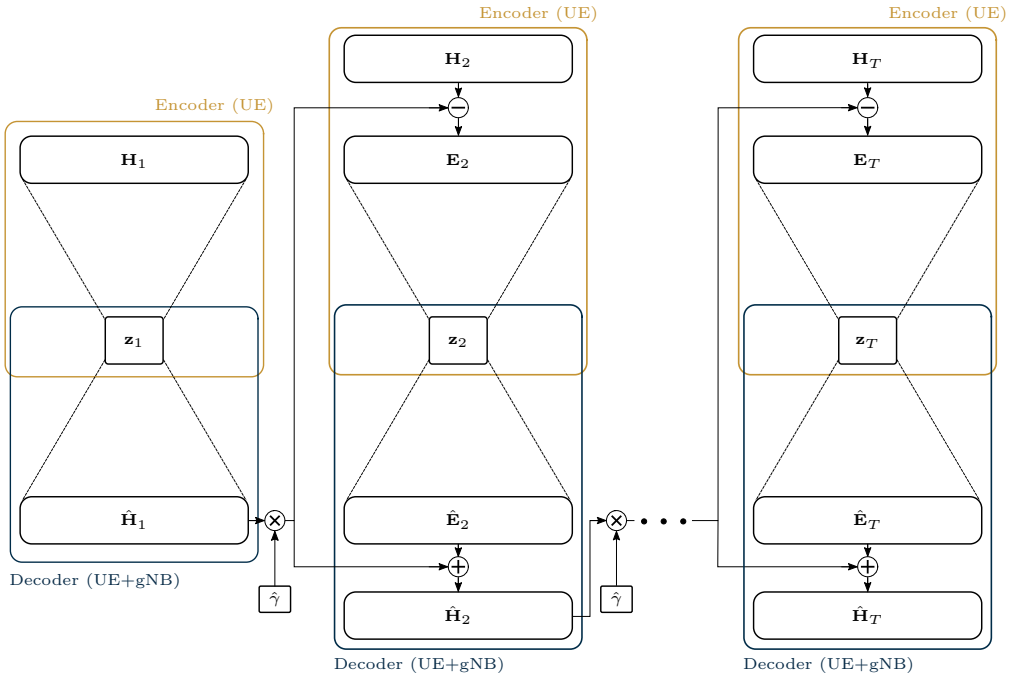


Figure 3.2: Abstract architecture for MarkovNet. Networks at t_i for $i \geq 2$ are trained to predict the estimation error, $\mathbf{E}_i$.

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 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.

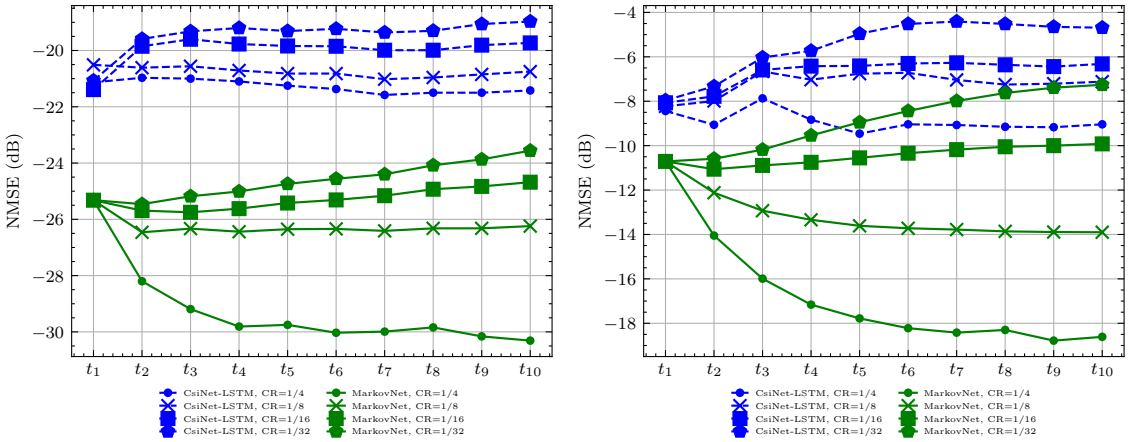
Table 3.1: Model size/computational complexity of tested temporal networks (CsiNet-LSTM, MarkovNet) and comparable non-temporal network (CsiNet). M: million.

	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

3.3 Numerical Results

We compare MarkovNet with CsiNet-LSTM [23] on the indoor and outdoor COST2100 datasets (for details, see Section 1.2). 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 γ using the training set. To compare the estimation accuracy of each network, we report the NMSE.

Figure 3.3 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 3.4 shows a random sample from the test set, $\mathbf{H}$, and the estimates produced 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).



(a) Indoor (b) Outdoor

Figure 3.3: NMSE comparison of MarkovNet and CsiNet-LSTM at various compression ratios (CR).

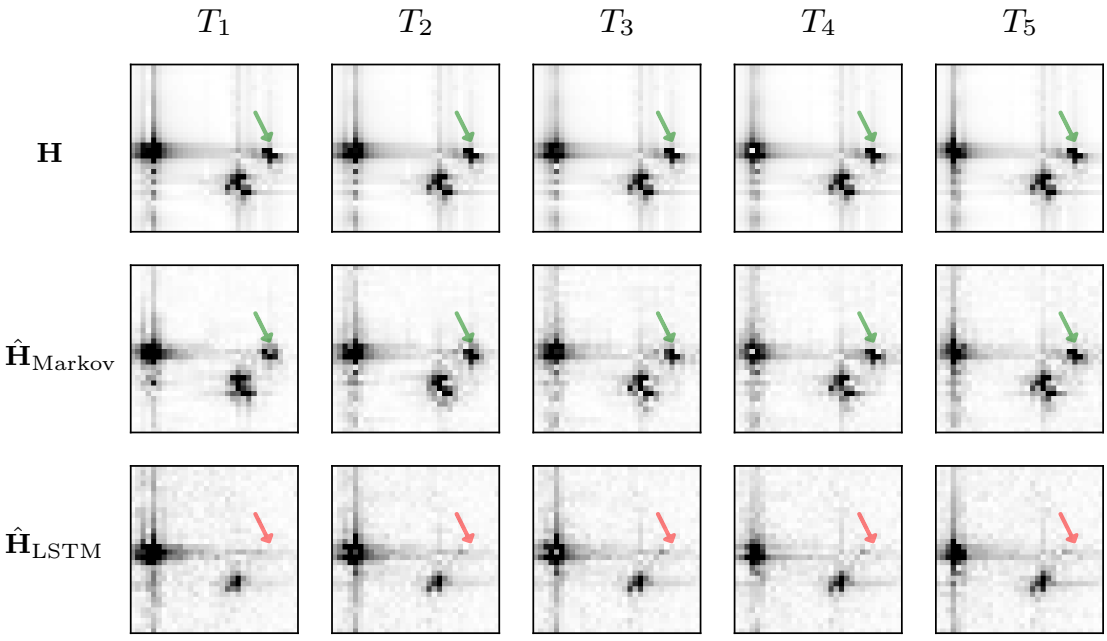


Figure 3.4: CSI ($\mathbf{H}$), MarkovNet estimates ($\hat{\mathbf{H}}_{\text{Markov}}$), and CsiNet-LSTM estimates ($\hat{\mathbf{H}}_{\text{LSTM}}$) across five timeslots (T_1 through T_5) on one outdoor channel sample from the test set, using CR $= \frac{1}{4}$.

Chapter 4

CsiNet-SoftQuant

While previous works investigated neural architectures for learned encoding-decoding of CSI data, such architectures rely on continuous valued latent representations (codewords). The use of such continuous codewords presents at least two issues:

1. **Quantization**: Modulation protocols require quantized data to accommodate bit string representations and IQ constellations. To make neural autoencoders compatible with common modulation schemes, trainable CSI compression techniques should incorporate quantized codewords in the learning process.

2. **Metrics for Compressibility**: Works in learnable CSI compression typically present a given network's reconstruction error a range of compression ratios (e.g., [18, 22]) but do not discuss the network's compatibility with coding schemes. Coding techniques such as arithmetic coding [48]) require probability estimates of the encoded symbols in order to operate [49]. Based on Shannon's coding theorem [50], the entropy of the encoded alphabet establishes the minimum transmission rate needed to encode a given symbol. To describe their compatibility with optimal coding schemes, neural CSI compression techniques should be presented with the entropy of their encoded

alphabets in order to assess the realized encoding distributions.

4.1 Related Work

Prior work has investigated feedback quantization in deep learning-based CSI compression. In [24], 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) [51]. 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 [24]. In [25], 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.

To achieve discrete codewords with valid probability distributions, we consider works in neural discrete representations. In [52], the authors propose *vector quantization (VQ)*, which partitions a r-dimensional latent space into d-dimensional vectors and quantizes the latent space based on a nearest neighbor assignment. In [53], the authors proposed soft-to-hard VQ (SHVQ), a softmax relaxation of VQ which enables a latent entropy term which can be used to regularize the loss function. Unlike the prior work, SHVQ allows for end-to-end training with feedback quantization.

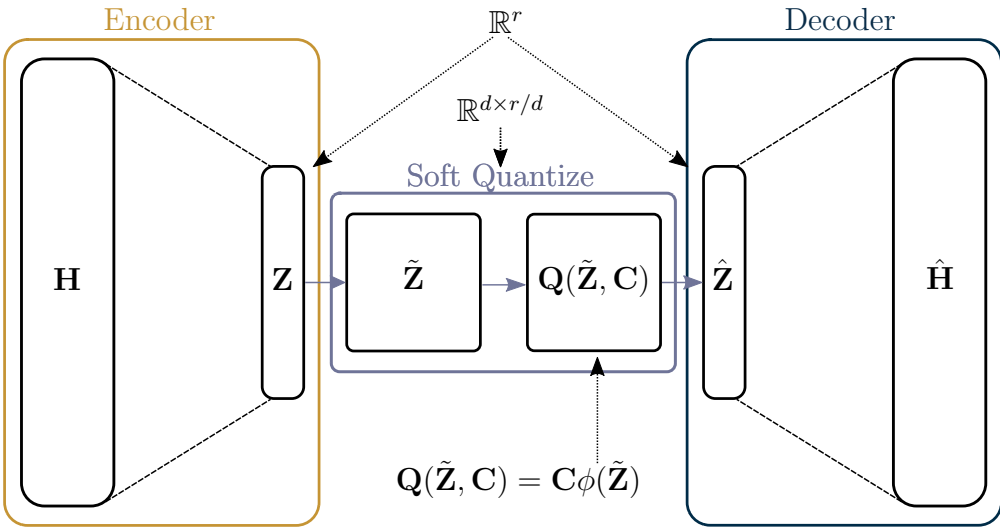


Figure 4.1: Abstract architecture for CsiNet-Quant. SoftQuantize layer ($Q(\tilde{\mathbf{Z}})$) is a continuous, softmax-based relaxation of a d-dimensional quantization of the latent layer $\mathbf{Z}$.

4.2 A Vector-quantized Autoencoder for Trainable Codewords

To incorporate discrete latent codewords in the learning process, we propose to use soft-to-hard vector quantization framework proposed in [53]. We choose a vector dimension, m, by which to partition the latent space $\mathbf{Z} = f(\mathbf{H}, \theta_e)$, 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

$$\phi(\tilde{\mathbf{z}}_j) = \left[\frac{\exp(-\sigma \|\tilde{\mathbf{z}}_j - \mathbf{c}_\ell\|^2)}{\sum_{i=1}^{L} \exp(-\sigma \|\tilde{\mathbf{z}}_j - \mathbf{c}_i\|^2)} \right]_{\ell \in [L]} \in \mathbb{R}^L \tag{4.1}$$

where (4.1) is typically referred to as the *softmax* function, a commonly adopted alternative to the max function which is fully differentiable. 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 $\mathbf{C} \in \mathbb{R}^{m \times L}$,

$$Q(\tilde{\mathbf{z}}_j, \mathbf{C}) = \phi(\tilde{\mathbf{z}}_j)\mathbf{C}^T. \tag{4.2}$$

The quantized version of the latent variable 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 4.1.

4.2.1 Entropy-regularization

To optimize the network with soft quantization, we adapt the loss function to resemble the canonical rate-distortion function by adding an entropy penalization term,

$$\underset{\theta_e, \theta_d, \mathbf{C}}{\operatorname{argmin}} \underbrace{\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{H}_i - g(Q(f(\mathbf{H}_i, \theta_e), \mathbf{C}), \theta_d)\|^2}_{\text{distortion loss}} + \underbrace{\lambda \left(\|\theta_e\|^2 + \|\theta_d\|^2 + \|\mathbf{C}\|^2 \right)}_{\ell^2 \text{ penalty}} + \underbrace{m\beta H(\phi)}_{\text{rate loss}}. \tag{4.3}$$

Where $H(\phi) = H(p, q)$ is the crossentropy based on the hard and soft probability estimates p and q, respectively. Before defining the estimates p and q, we briefly discuss the population probabilities of the latent codewords. Denote the symbol encoder/decoder pair as $E : \mathbb{R}^m \to [L]^m / D : [L]^m \to \mathbb{R}^m$. Denote the distribution of latent variables as Z such that $\mathbf{z} \sim \mathsf{Z}$ with the encoder $E(\mathsf{Z}) = \mathbf{e}$. The entropy of Z is given as

$$H(E(\mathsf{Z})) = -\sum_{\mathbf{e} \in [L]^m} P(E(\mathsf{Z}) = \mathbf{e}) \log_2(P(E(\mathsf{Z}) = \mathbf{e})).$$

In practice, the true population probabilities $P(E(\mathsf{Z}))$ are inaccessible, and we must estimate the probability masses via finite sampling over the encoder's outputs, $e(\mathbf{z})$. The hard

probability estimate p_j of the j-th codeword is

$$p_j = \frac{|\{e_l(\mathbf{z}_i)|l \in [m], i \in [N], e_l(\mathbf{z}_i) = j\}|}{mN}.$$

The soft assignments of ϕ admit valid probability masses, $q_j = \phi(\tilde{\mathbf{z}})$, over the codewords. Using histogram estimates p_j and the soft assignments q_j, the crossentropy term is written

$$H(\phi) := H(p, q) = -\sum_{j=1}^{L} p_j \log q_j = H(p) + D_{\mathrm{KL}}(p\|q)$$

where $D_{\mathrm{KL}}(p\|q) = -\sum_{j=1}^{L} p_j \log\left(\frac{p_j}{q_j}\right)$ is the Kullback Liebler (KL) divergence. Due to the nonnegativity of D_{KL}, $H(\phi)$ is an upper bound on $H(p)$, and so (4.3) is a valid optimization target.

4.2.2 Softmax annealing

During training, we gradually increase the value of the temperature parameter, σ, for the softmax function, resulting in a gradual transition from soft-to-hard quantization. Denote the training iteration t and the corresponding temperature σ_t. The annealing schedule

$$\sigma_t = \sigma_{t-1} + K_\sigma \mathrm{gap}(t)$$

Where K_σ is a constant and $\mathrm{gap}(t)$ is the difference in network mean squared error between soft and hard quantization. The mean squared error of the soft (hard) network is given as $e_S = \|\tilde{F}(\mathbf{H}) - \mathbf{H}\|^2$ ($e_H = \|\hat{F}(\mathbf{H}) - \mathbf{H}\|^2$), and the performance gap is given as $\mathrm{gap}(t) = e_H - e_S$.

4.3 Entropy Estimation for CSI Matrices

Entropy as defined by Shannon is a measure of the amount of information in a random variable [50]. While most works in CSI compression either list a range of compression ratios or compare the achieved bit rate of different networks, there is a gap in the literature with respect to the limits of compression. To assess the limits of compressibility of CSI, we seek to quantify the entropy of CSI matrices. Section 4.3.1 discusses an estimated upper bound based on quantized CSI matrices, and Section 4.3.2 proposes an upper bound based on the differential entropy of CSI matrices. Section 4.3.3 proposes a method for establishing a rate-distortion curve based on CSI matrices with additive noise.

4.3.1 Entropy Estimation of Quantized CSI

Given angular-delay domain CSI data, $\mathbf{H}$, we assume i.i.d. $\mathbf{H}_{(i,j)}$ for i-th (j-th) row (col) of $\mathbf{H}$. Denote the quantized CSI matrix, $\mathbf{H}^{\Delta}$, under b-bit quantization. The entropy of the (i,j)-th element is

$$H(\mathbf{H}^{\Delta}_{(i,j)}) = -\sum_{k}^{2^b} p(\mathbf{H}^{\Delta}_{(i,j)} = k) \log p(\mathbf{H}^{\Delta}_{(i,j)} = k),$$

where $p(H^{\Delta}_{(i,j)} = k)$ can be obtained as a histogram estimate over the entire dataset,

$$p(\mathbf{H}^{\Delta}_{(i,j)} = k) = \frac{|\mathbf{H}_{(i,j)} = k : i \in [R_d], j \in [n_T], k \in [2^b - 1]|}{N}.$$

A conservative upper bound on the entropy of the full CSI matrix is

$$H(\mathbf{H}^{\Delta}) \leq \sum_{i}^{R_d} \sum_{j}^{n_T} H(\mathbf{H}^{\Delta}_{(i,j)}) = H_{\text{UB}}(\mathbf{H}^{\Delta}). \tag{4.4}$$

4.3.2 Differential Entropy Estimation of Quantized CSI

Assume i.i.d. $\mathbf{H}_{(i,j)}$ for i-th (j-th) row (col). 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 [54] for each element in $\mathbf{H}$ and average over the elements,

$$h(\mathbf{H}) \leq \sum_{i}^{R_d}\sum_{j}^{n_T}\hat{h}(\mathbf{H}_{(i,j)}) = h_{\mathrm{UB}}(\mathbf{H}), \tag{4.5}$$

for KL estimator $\hat{h}$. Based on Theorem 8.3.1 from Cover [55], for sufficiently small quantization interval $\Delta = \frac{1}{2^n}$, the entropy of a quantized random variable is related to its differential entropy as,

$$H(\mathbf{H}^{\Delta}) = h(\mathbf{H}) + n, \tag{4.6}$$

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

4.3.3 Rate-Distortion Estimation

The ultimate goal of estimating the entropy of CSI matrices is to establish a rate-distortion curve. Such a curve will establish the limits of compression and accuracy for works in learnable feedback quantization. Based on either estimator outlined above, we can utilize

additive 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 can calculate the bounds $\hat{H}(\mathbf{H}_\sigma^\Delta)$ or $\hat{h}(\mathbf{H}_\sigma^\Delta)$ from (4.4) or (4.5) for different noise levels σ to establish a rate-distortion curve.

4.4 Results

We use SHVQ [53] to perform quantization on the tested CSI estimation networks. We used the COST2100 data introduced in Section 1.2. We train the network in three stages:

1. **Autoencoder pretraining**: Training the autoencoder ($\hat{\mathbf{H}} = g(f(\mathbf{H}, \vec{\theta}_e), \vec{\theta}_d)$) without latent quantization (1000 epochs). The autoencoder is trained with the MSE objective function.

2. **Center pretraining**: Training soft quantization layer to initialize centers, $\mathbf{C}$ (1000 epochs). Using $\vec{\theta}_e$ from stage 1, the soft quantizer is trained on $\mathbf{Z} = f(\mathbf{H}, \vec{\theta}_e)$ to minimize the cluster energy, $\underset{\mathbf{C}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\tilde{\mathbf{Z}} - Q(\tilde{\mathbf{Z}})\|^2$.

3. **SHVQ finetuning**: Using the results of stage 1 ($\vec{\theta}_e, \vec{\theta}_d$) and stage 2 ($\mathbf{C}$), we finetune the autoencoder and the soft quantization layer (50 epochs). The soft-quantized autoencoder is trained with the entropy-regularized MSE (Equation (4.3)).

We use a batch size of 200. We perform a training/testing split of 75k/25k sample. For stage 3, we sweep the parameter β to realize different latent entropy values $H(\phi)$. To visualize the rate-distortion of the proposed network, we show the network's NMSE versus

Table 4.1: Parameters/hyperparameters used for CsiNet-SoftQuant.

Symbol	**Values**	**Description**
L	1024	Number of centers/codewords for VQ.
d	4	Dimensionality of vectors in VQ.
r	$512, 256, 128$	Dimensionality of encoder's output/latent layer.

the bits per pixel (bpps),

$$\text{bpps} = \frac{rm}{2n_b R_d} H(\phi) = CR \times mH(\phi).$$

We also perform experiments with arithmetic encoding on the hard quantized centers, and we report the resulting bits per pixel as

$$\text{bpps} = \frac{\frac{1}{N}\sum_i^N b_{\text{AE},n}}{2n_b R_d}.$$

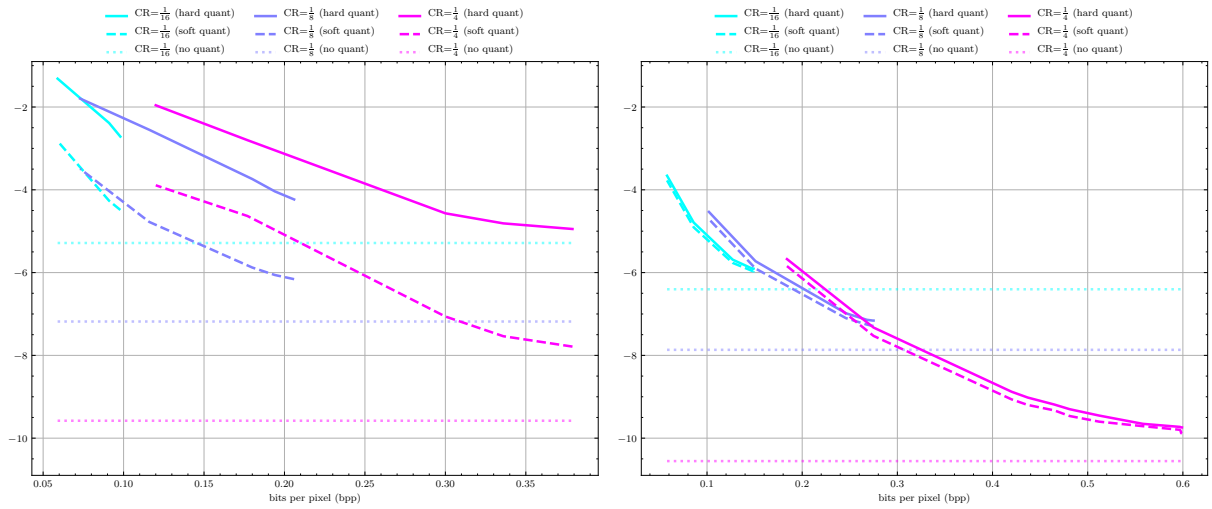
where $b_{\text{AE},n}$ is the bits per feedback message under arithmetic encoding for the n^{th} sample.

We demonstrate the performance of CsiNet and SphNet under latent quantization. Table 4.1 summarizes the parameters used in these tests.

4.4.1 Results: Rate-Distortion

Figures 4.2a and 4.2b show the performance of CsiNet-Quant under minmax and spherical normalization, respectively. We demonstrate the gap between the soft (dashed line) and hard quantization (solid line) of each tested compression ratio. In addition to achieving better NMSE performance overall, the network under spherical normalization has smaller performance gaps between soft and hard quantization.

Figure 4.3 shows the rate-distortion for both minmax and spherical normalization under hard quantization only. At comparable bit rates, the NMSE of SphNet is consistently



(a) Minmax normalization

(b) Spherical normalization

Figure 4.2: Rate distortion of CsiNet-Quant trained on the outdoor network under a) minmax normalization and b) spherical normalization using: $L = 1024$ centers, $d = 4$. Hard, soft, and no quantization performance shown for each CR.

lower than that of CsiNet-Pro, illustrating that spherical normalization enables better latent representations than minmax normalization.

4.4.2 Results: CSI Entropy Estimation

Figure 4.4 shows the estimated conditional entropy of the quantized CSI matrices ($\mathbf{H}^{\Delta}$) for the indoor and outdoor COST2100 channel samples. The conditional entropy is based on the following identity,

$$\hat{H}(\mathbf{H}^{\Delta}_{t_2}|\mathbf{H}^{\Delta}_{t_1}) = \hat{H}(\mathbf{H}^{\Delta}_{t_2}, \mathbf{H}^{\Delta}_{t_1}) - \hat{H}(\mathbf{H}^{\Delta}_{t_1})$$

for a feedback interval $t_{\text{interval}} = t_2 - t_1$. A feedback interval of $t_{\text{interval}} = \infty$ indicates no conditioning (i.e., $\hat{H}(\mathbf{H}^{\Delta}_{t_2})$). Both figures show the conditional entropy for different feedback intervals, and we see that the conditional entropy for longer feedback intervals is consistently higher than that of shorter feedback intervals.

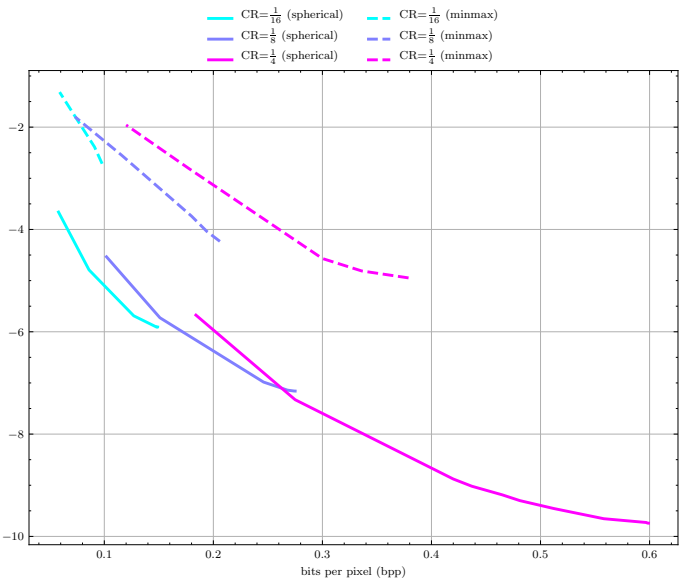
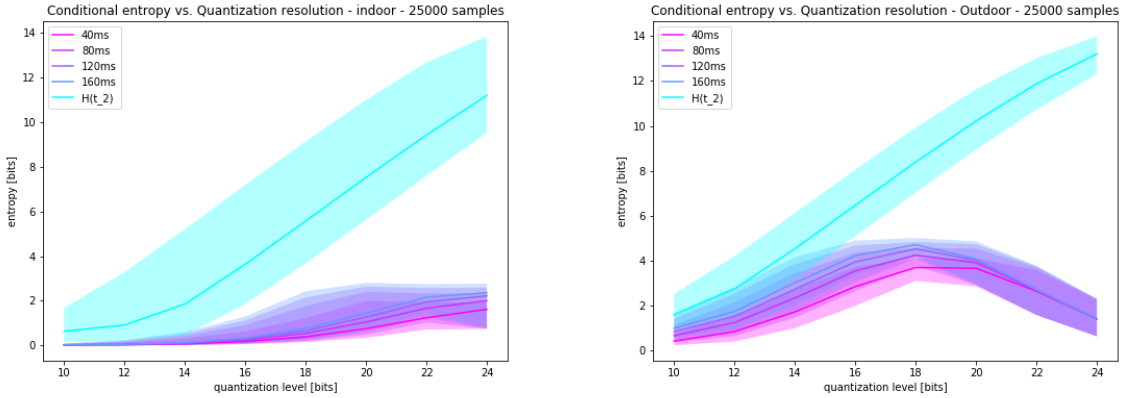


Figure 4.3: Rate distortion of CsiNet-Quant trained on the outdoor network under hard quantization for both minmax (dotted line) and spherical (solid line) normalization.



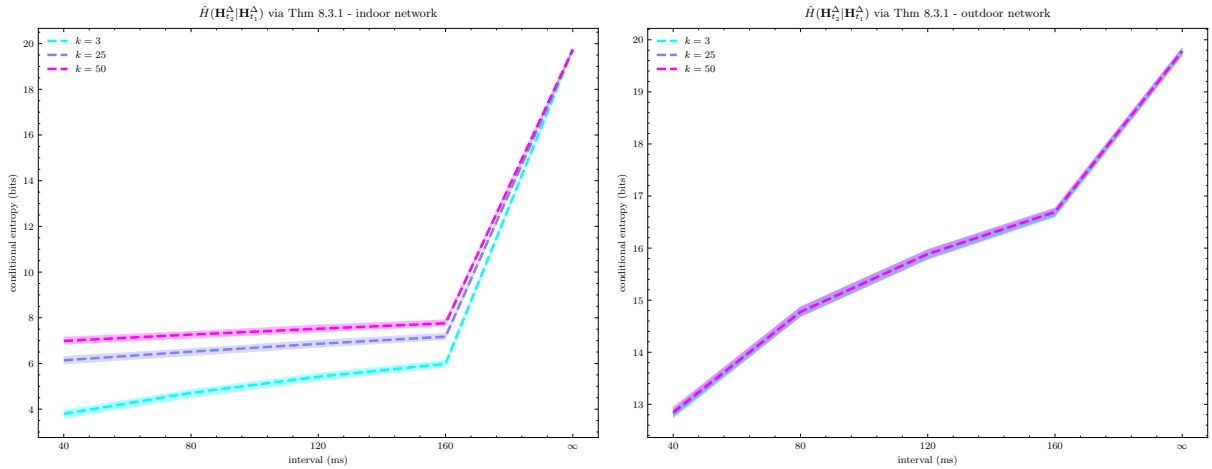
(a) Indoor (b) Outdoor

Figure 4.4: Mean entropy/conditional entropy estimates $H(\mathbf{H}^{\Delta})$ with 95% c.i. for quantized i.i.d COST2100 elements vs. quantization level (bits).

Figure 4.5 shows the estimated conditional entropy of the quantized CSI matrices ($\mathbf{H}^{\Delta}$) based on the differential entropy estimates ($\hat{h}(\mathbf{H})$). The conditional differential entropy is based on the following identity,

$$\hat{h}(\mathbf{H}_{t_2}|\mathbf{H}_{t_1}) = \hat{h}(\mathbf{H}_{t_2}, \mathbf{H}_{t_1}) - \hat{h}(\mathbf{H}_{t_1}),$$

and the entropy of the quantized CSI matrices is obtained via Theorem 8.3.1 from Cover [55] as defined in (4.6).



(a) Indoor (b) Outdoor

Figure 4.5: Mean entropy/conditional entropy estimates $\hat{H}(\mathbf{H}^{\Delta}) = \hat{h}(\mathbf{H}) + n$ with 95% c.i. for quantized i.i.d COST2100 elements vs. feedback interval (ms).

Using $\mathbf{H}^{\Delta}_{\sigma}$, we can draw the rate-distortion curve for the corrupted CSI, and we can compare this curve to the achieved rate-distortion the tested networks. Figure 4.6) demonstrates the achieved rate-distortion under spherical normalization for CsiNet-SoftQuant. Compared with uniform quantization, SHVQ can achieve similar NMSE at lower feedback rates, but based on the bound $\hat{H}(\mathbf{H}^{\Delta}_{\sigma})$, it is possible to achieve lower feedback bit rates at the given distortion levels. Note that the performance of CsiNet-SoftQuant was achieved with minimal hyperparameter optimization. The ultimate compression rate can likely be improved by changing the dimensionality of the codebook (L), the size of codewords (m), or training hyperparameters (see [53] for more details).

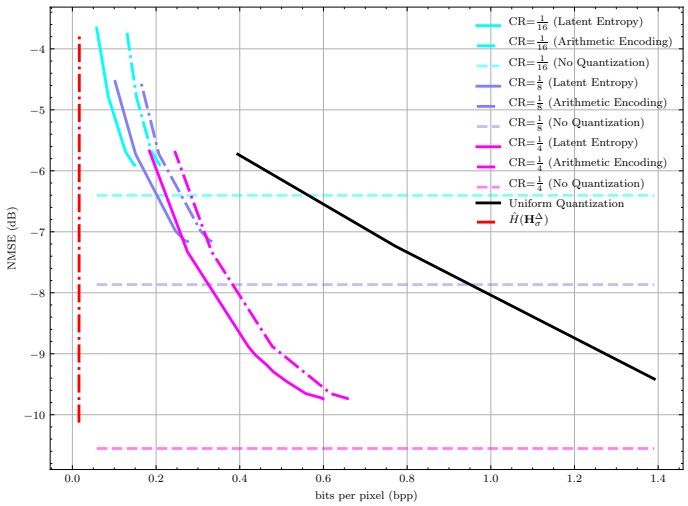


Figure 4.6: CsiNet-SoftQuant compared to uniform quantization of latent feedback (**black**) and compared to $\hat{H}(\mathbf{H}_\sigma^\Delta)$ (**red**)

4.5 Future Work: Rate-optimal Quantization

Moving forward, our work in trainable CSI quantization will seek to specify networks which achieve **rate-optimal quantization**. A rate-optimal network will achieve feedback entropy approaching the entropy of the underlying CSI matrices. In addition to hyperparameter optimization (as discussed in Section 4.4.2) and different quantization frameworks (e.g., MCR^2 [56]), we plan to investigate the following two avenues to achieve rate-optimal CSI compression.

4.5.1 ROI CSI Compression

Region of interest (ROI) based compression emphasizes the encoding and decoding of designated sections of a given signal. Deep learning-based ROI compression has demonstrated success in image compression tasks, demonstrating lower distortion at lower rates compared to non-ROI image codecs [57]. To apply ROI compression to MIMO CSI, we would need to consider how to define ground truth ROI masks based on either delay spread or the power distribution of $\mathbf{H}$.

4.5.2 Differential Encoding with Trainable Feedback Quantization

As highlighted by our entropy estimation experiments (Figures 4.4 and 4.5), the conditional entropy $\hat{H}(\mathbf{H}_{t_2}|\mathbf{H}_{t_1})$ is appreciably lower than the entropy $\hat{H}(\mathbf{H}_{t_2})$. The resulting entropy reduction implies a corresponding reduction in feedback rate for compressed CSI. Our future work will investigate the compatibility of differential encoding (see Section 3) with trainable feedback quantization. We anticipate that the achieved bit rates for quantized differential encoders should be substantially lower than non-temporal encoders.

Chapter 5

Conclusion

In this proposal, we discussed techniques for efficient MIMO channel state information (CSI) estimation. We outlined the author's prior work using spherical normalization, which scales each CSI sample by its power, and a deep learning framework for differential encoding, which exploits temporal correlation between CSI at subsequent timeslots. We presented initial results for neural network with trainable quantization for practical CSI feedback quantization, and we outline possible extensions which will enable rate-optimal encoding of CSI including ROI encoding and differential encoding.

Bibliography

[1] 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.

[2] 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.

[3] F. Kaltenberger, H. Jiang, M. Guillaud, and R. Knopp, "Relative channel reciprocity calibration in mimo/tdd systems," in *2010 Future Network Mobile Summit*, pp. 1–10, June 2010.

[4] 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.

[5] 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*, pp. 1–5, Aug 2010.

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

[7] 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, pp. 92–99, December 2012.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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*, pp. 581–587, IEEE, 2008.

[13] T. Hastie, R. Tibshiran, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2016.

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

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

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

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

[18] C. Wen, W. Shih, and S. Jin, "Deep Learning for Massive MIMO CSI Feedback," *IEEE Wireless Communications Letters*, vol. 7, pp. 748–751, Oct 2018.

[19] 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)*, pp. 1–6, 2020.

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

[21] 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.

[22] 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.

[23] 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, pp. 416–419, April 2019.

[24] 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)*, pp. 1–6, 2019.

[25] 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)*, pp. 8579–8583, 2020.

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

[27] 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*, pp. 1501–1510, 2017.

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

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

[30] 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)*, pp. 1–8, IEEE, 2010.

[31] 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.

[32] 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.

[33] 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.

[34] 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)*, pp. 1–6, 2020.

[35] 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.

[36] 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.

[37] 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.

[38] C. Lu, W. Xu, H. Shen, J. Zhu, and K. Wang, "MIMO Channel Information Feedback Using Deep Recurrent Network," *IEEE Commu. Letters*, vol. 23, pp. 188–191, Jan 2019.

[39] Y. Liao, H. Yao, Y. Hua, and C. Li, "CSI Feedback Based on Deep Learning for Massive MIMO Systems," *IEEE Access*, vol. 7, pp. 86810–86820, 2019.

[40] 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.

[41] 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.

[42] 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.), pp. 190–198, 2013.

[43] 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.

[44] 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.

[45] 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.

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

[47] Z. Liu †, **M. del Rosario †**, and Z. Ding, "A Markovian Model-Driven Deep Learning Framework for Massive MIMO CSI Feedback," *arXiv e-prints*, Sept. 2020. Submitted to IEEE Transactions on Wireless Communications.

[48] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[49] P. G. Howard and J. S. Vitter, "Arithmetic coding for data compression," *Proceedings of the IEEE*, vol. 82, no. 6, pp. 857–865, 1994.

[50] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[51] 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.

[52] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv preprint arXiv:1711.00937*, 2017.

[53] 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.

[54] 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.

[55] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[56] Y. Yu, K. H. R. Chan, C. You, C. Song, and Y. Ma, "Learning diverse and discriminative representations via the principle of maximal coding rate reduction," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[57] C. Cai, L. Chen, X. Zhang, and Z. Gao, "End-to-end optimized roi image compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 3442–3457, 2020.