CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

PHD THESIS

Every time you send a message, stream a video, or call a friend, your words and data travel through an invisible, ever-changing medium. The state of this medium, known as the wireless channel can be described by channel state information (CSI), which acts like a real-time weather report of the airwaves, describing how signals twist, fade, and scatter as they move through space. CSI can be handled much like many other types of data: it can be analyzed, compressed, predicted, or enhanced. However, CSI is inherently stochastic and high-dimensional, and obtaining it reliably is challenging. AI has already transformed countless domains, from images and audio to language and medical data, showing its strength in extracting patterns from complex, high-dimensional information. With sufficient CSI data, AI becomes equally attractive in wireless communication. This thesis examines how CSI can be viewed as its own data modality and how AI techniques can be leveraged to acquire it more efficiently and accurately. Specifically, it explores how to:

- Estimate CSI when observations are noisy or incomplete.
- Compress CSI to reduce communication overhead while preserving fidelity.
- Predict CSI so that networks can adapt intelligently to fast-changing environments.

The central vision is that by viewing CSI not just as a physical parameter but as rich data in its own right, we can design communication systems that are faster, smarter, and more reliable, bringing us closer to a future of truly intelligent, seamless wireless connectivity.

MEHDI SATTARI • CSI Estimation, Compression, and Prediction Using Deep Learning • 2026

CSI Estimation, Compression, and Prediction Using Deep Learning

MEHDI SATTARI

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

CSI Estimation, Compression, and Prediction Using Deep Learning

Mehdi Sattari

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2026

CSI Estimation, Compression, and Prediction Using Deep Learning

Mehdi Sattari
ISBN 978-91-8103-377-9

Acknowledgments, dedications, and similar personal statements in this thesis reflect the author's own views.

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5834
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000

This work was supported in part by the SEMANTIC project funded by the EU's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 861165, and in part by the Hexa-X-II project, which has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under grant agreement No. 101095759.

Cover:
A futuristic banner showing a "CSI" letter at the center, with a modern smartphone on the left and a luminous AI circuit brain on the right, all connected by propagating electromagnetic waves.

Printed by Chalmers Digital Printing
Gothenburg, Sweden, Feburary, 2026

CSI Estimation, Compression, and Prediction Using Deep Learning

Mehdi Sattari
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Acquiring accurate channel state information (CSI) is essential for enabling reliable and efficient wireless transmission and reception. However, CSI is inherently stochastic, high-dimensional, and time-varying, which makes its acquisition particularly challenging. Motivated by the success of deep learning (DL) across many data modalities, this thesis explores DL-based solutions for CSI estimation, compression, and prediction.

First, we study CSI estimation in full-duplex (FD) multiple-input multiple-output (MIMO) systems, where strong self-interference (SI) complicates channel acquisition. To reduce the pilot and computational burden of estimating both SI and user channels, we propose a pilot-sharing strategy together with a convolutional neural network that jointly estimates these channels. We further introduce a neural mapping that enables CSI acquisition at the transmit chain.

Second, we investigate DL–based CSI compression and its limited robustness under distribution shifts. To address this issue, we adopt a full-model fine-tuning while explicitly accounting for model update signaling overhead. Specifically, we employ a spike-and-slab prior to promote sparsity in the model updates and fine-tune the pretrained network using a rate–distortion objective regularized by the update bit rate.

Third, we tackle CSI prediction using a diffusion-based generative framework. The method consists of a temporal encoder that extracts latent features from past CSI and a diffusion generator that synthesizes future CSI. We also study a simplified encoder-free design to reduce latency, compare autoregressive and sequence-to-sequence inference, and explore multiple architectures for both temporal encoding and diffusion generation.

Keywords: Channel estimation, CSI compression, CSI Prediction, Deep learning

To my family.

List of Publications

This thesis is based on the following publications:

[A] **Mehdi Sattari**, Hao Guo, Deniz Gündüz, Ashkan Panahi, Tommy Svensson, "Full-Duplex Millimeter Wave MIMO Channel Estimation: A Neural Network Approach". *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 1093-1108, 2024.

[B] **Mehdi Sattari**, Deniz Gündüz, Tommy Svensson, "Neural CSI Compression Fine-Tuning: Taming the Communication Cost of Model Updates ". Submitted to *IEEE Transactions on Machine Learning in Communications and Networking*, Feb, 2026.

[C] **Mehdi Sattari**, Javad Aliakbari, Alexandre Graell i Amat, Tommy Svensson, "CSI Prediction Using Diffusion Models". *IEEE Transactions on Wireless Communications*, major revision, Feb, 2026.

Other publications by the author, not included in this thesis, are:

[D] **Mehdi Sattari**, Deniz Gündüz, Tommy Svensson, "Dynamically Fine-Tuned Neural Compressor for FDD Massive MIMO CSI Feedback". *IEEE 26th International Workshop on Signal Processing and Artificial Intelligence for Wireless Communications (SPAWC)*, Surrey, United Kingdom, 2025, pp. 1-5, doi: 10.1109/SPAWC66079.2025.11143260..

[E] **Mehdi Sattari**, Javad Aliakbari, Alexandre Graell i Amat, Tommy Svensson, "CSI Prediction Using Autoregressive Conditional Diffusion Models". Accepted for publication in *IEEE International Conference on Communications (ICC)*, 2026..

Contents

Acknowledgments

As I come to the end of my PhD, I feel deeply grateful. These years were not only about research and results, but also about personal growth and the people around me. The challenges were real, but so was the support I received. I was fortunate to be surrounded by people who gave their time, advice, encouragement, and kindness. To everyone who stood by me during this journey, thank you. You made these years meaningful and truly memorable.

First and foremost, I would like to sincerely thank my supervisor, Professor Tommy Svensson, for his guidance, encouragement, and unwavering support throughout my PhD. Your insightful advice, broad vision, and genuine care have shaped both my research and personal growth. Beyond technical mentorship, you consistently inspired me to become a better thinker and a better person. You made the PhD journey much more pleasant and helped ease the inevitable pressure along the way.

I am deeply grateful to Professor Deniz Gündüz for his inspiring collaboration, thoughtful feedback, and for hosting me during my research visit to Imperial College London. Working with you was a valuable learning experience. Your clarity of thought, enthusiasm for research, and high standards have had a lasting impact on my development as a researcher.

My sincere thanks go to Javad Aliakbari for the many enlightening discussions, valuable collaborations, and warm friendship that enriched my PhD years. Beyond your technical brilliance, your positive energy and constant smile always bring encouragement to everyone in the ComSys group. I am also thankful to my collaborators, Dr. Ashkan Panahi, Professor Alexandre Graell i Amat, and Dr. Hao Guo for their insightful contributions.

To my colleagues at ComSys, thank you for creating such a supportive and inspiring work environment. I am especially grateful to Professor Fredrik Brännström, our group leader, for his strong leadership and clear guidance, and to Professor Erik Ström for his thoughtful coordination of the Communication, Antenna, and Optical research division within the E2 Department. I would also like to thank my colleagues in the Wireless Systems group–Charitha, Azadeh, Akshay, Morteza, Venkatesh, and Armitha–for their kindness, support, and the enjoyable Friday meetings. Special thanks to my officemates, Alireza, André, and Kaan, for making our office a calm and motivating place to work.

Life in Gothenburg would not have been the same without my dear friends.

Thank you for the laughter, the unforgettable moments, and the support that made this journey truly enjoyable. My family means more to me than words can express. To Mom and Dad, thank you for your unconditional love, sacrifices, and for always standing by me through every challenge. To my brother Davood and his wife Pouneh, and my lovely nephew Elman; and to my sister Nasrin and her husband Mikael, and my adorable nieces Zahra and Hanieh–your encouragement, love, and support have been my greatest strength. This thesis is for you.

Acronyms

2D:	two-dimensional
3D:	three-dimensional
5G:	fifth generation mobile network
6G:	sixth generation mobile network
ADC:	analog-to-digital converter
AE:	Autoencoder
AF:	amplify and forward
AI:	artificial intelligence
AoA:	angle of arrival
AoD:	angle of departure
AR:	autoregressive
AS:	angular spread
AWGN:	additive white Gaussian noise
BS:	base station
CABAC:	context-adaptive binary arithmetic coding
CDL:	clustered delay line
CME:	conditional mean estimator
CNN:	convolutional neural network
CP:	cyclic prefix
CSI:	channel state information
DDIM:	denoising diffusion implicit model

DDPM:	denoising diffusion probabilistic model
DFT:	discrete Fourier transform
DiT:	diffusion Transformer
DL:	deep learning
DNN:	deep neural network
DoA:	direction of arrival
DoD:	direction of departure
EE:	energy efficiency
EMA:	exponential moving average
ELBO:	evidence lower bound
FDD:	frequency-division duplexing
FEEL:	federated edge learning
FFN:	feed-forward network
FLOP:	floating point operation
FNN:	fully connected neural network
GAN:	generative adversarial network
GELU:	Gaussian error linear unit
GRU:	gated recurrent unit
GPU:	graphics processing unit
i.i.d:	independent and identically distributed
IQ:	in-phase/quadrature
IoT:	internet of things
ISI:	inter-symbol interference

LoS:	line-of-sight
LTE:	long term evolution
LSTM:	long short-term memory
LS:	least squares
MAP:	maximum a posteriori
MF:	matched filter
ML:	machine learning
MLE:	maximum likelihood estimation
MLP:	multilayer perceptron
MMSE:	minimum mean square error
MIMO:	multiple-input multiple-output
mmWave:	millimeter wave
MRC:	maximum-ratio combining
MSE:	mean squared error
NLoS:	non-line-of-sight
NMSE:	normalized mean square error
NN:	neural network
OFDM:	orthogonal frequency-division multiplexing
PDF:	probability density function
QuaDRiGa:	QUAsi Deterministic RadIo channel GenerAtor
RD:	rate–distortion
ReLU:	rectified linear unit
RF:	radio frequency

RNN:	recurrent neural network
Rx:	receiver
SDE:	stochastic differential equations
SE:	spectral efficiency
Seq2Seq:	sequence-to-sequence
SiLU:	sigmoid linear unit
SNR:	signal-to-noise ratio
STE:	straight-through estimator
TDD:	time-division duplexing
Tx:	transmitter
UE:	user equipment
ULA:	uniform linear array
UPA:	uniform planar array
UAV:	unmanned aerial vehicle
VAE:	variational autoencoder
ViT:	vision Transformer
ZF:	zero-forcing

Part I

Overview

CHAPTER

Introduction

1.1 Background

Wireless networks have transformed from simple voice communication systems into the backbone of today's digital society. Each new generation of mobile technology has not only improved data rates and coverage but also enabled entirely new services and applications. While fifth-generation (5G) networks are still being rolled out worldwide, research and standardization efforts for sixth-generation (6G) systems are already underway [1]–[3]. The Ericsson Mobility Report [4] predicts that mobile data traffic will more than triple by the end of the decade, driven by emerging applications such as immersive extended reality [5], autonomous systems, and massive Internet of Things (IoT) connectivity [6]. To support these growing demands, future wireless networks must deliver higher throughput, lower latency, and improved reliability, all while addressing critical challenges such as spectrum scarcity and energy consumption [7].

To meet the continuously growing demands for higher capacity, faster connectivity, and improved performance, a new generation of cellular networks has historically been introduced roughly every decade [8]. Each generation

not only boosts technical capabilities but also unlocks unforeseen business opportunities and enables entirely new applications. As a result, telecommunication networks must evolve in close alignment with emerging industrial trends and societal needs. This co-evolution requires both innovative technical solutions and a rigorous standardization process to ensure interoperability on a global scale [9]. With every generational leap, new functionalities and capacity-enhancing features are incorporated into the standards, ultimately shaping the foundation of the next-generation mobile platform.

With each new generation of mobile networks, technological progress introduces not only enhanced capabilities but also new challenges that must be overcome. Higher frequency bands, denser deployments, and advanced techniques such as massive multiple-input multiple-output (MIMO) promise substantial gains in throughput and efficiency, but they also push existing design paradigms to their limits. Meeting the demands for faster speeds, lower latency, and improved energy efficiency (EE) requires rethinking how networks are modeled, managed, and optimized [10]. As complexity grows, traditional approaches often struggle to keep pace, underscoring the need for innovative solutions that can adapt to rapidly changing environments.

A central aspect of this challenge lies in acquiring accurate and timely channel state information (CSI). CSI characterizes how wireless signals propagate through the environment, encapsulating the effects of reflections, scattering, fading, and interference [11]. It effectively provides a fingerprint of the radio channel at a given time and frequency, enabling the network to adapt transmissions intelligently through techniques such as adaptive modulation and coding, beamforming, and interference management. However, CSI is high-dimensional and stochastically complex, which makes it difficult to obtain reliably. As networks evolve toward ultra-dense architectures and massive antenna arrays, efficient methods for CSI estimation, compression, and prediction become indispensable to sustain performance and scalability in 5G, 6G and beyond [12].

Conventional approaches to solving wireless communication problems have largely relied on mathematical modeling, analytical derivations, and signal processing techniques grounded in simplifying assumptions. While these methods have been remarkably successful in earlier generations of mobile networks, they struggle to cope with the increasing complexity of modern systems. For example, accurate modeling of multipath propagation, hardware impairments,

and user mobility often leads to intractable analytical formulations or requires oversimplifications that limit practical applicability. Moreover, traditional algorithms are typically hand-crafted for specific scenarios, making them rigid and less adaptable to diverse or rapidly changing environments. As wireless networks expand to support massive connectivity, higher frequency bands, and advanced technologies such as massive MIMO, the limitations of purely model-based methods become more evident—creating the need for new paradigms that can efficiently learn, adapt, and generalize from data [13].

Deep learning (DL) has emerged as one such paradigm, attracting widespread attention across both academia and industry. DL models can automatically learn hierarchical representations directly from data. This capability allows them to capture complex patterns and dependencies that are otherwise difficult to model analytically. The success of DL is evident in a wide range of real-world applications, from computer vision [14] and speech recognition [15] to natural language processing [16], healthcare [17], and autonomous systems [18]. Its ability to scale with large datasets and computational power has enabled breakthroughs that redefine state-of-the-art performance across disciplines. Consequently, DL has become a powerful tool for addressing problems that are highly nonlinear, data-rich, and difficult to model analytically—making it an increasingly essential component of modern technological solutions.

In this thesis, we focus on the fundamental problem of acquiring CSI, which is indispensable for enabling the advanced functionalities of modern and future wireless networks. Specifically, we investigate methods for estimating, compressing, and predicting CSI to ensure that it can be obtained accurately, efficiently, and with overhead. These challenges share deep connections with problems studied in other fields of DL. For example, CSI estimation resembles tasks in computer vision where missing or noisy information must be reconstructed; CSI compression parallels techniques in data compression used to reduce redundancy while preserving essential features; and CSI prediction is closely related to sequence modeling problems in natural language processing, where future tokens are inferred from past context. By bridging ideas across these domains, this thesis explores how powerful data-driven approaches can be adapted and extended to address the unique challenges of wireless communication systems.

1.2 Notation

Throughout this thesis, the following notations are used. Matrices are denoted by bold uppercase letters (e.g., $\mathbf{A}$), while vectors are denoted by bold lowercase letters (e.g., $\mathbf{a}$). Scalars are written in regular font. For a generic matrix $\mathbf{A} \in \mathbb{C}^{n \times m}$, $[\mathbf{A}]_{ij}$ refers to the element in the (i, j)-th position, and $\text{vec}(\mathbf{A})$ denotes the column vector obtained by stacking the columns of $\mathbf{A}$ one below the other. The identity matrix of size $n \times n$ is represented by $\mathbf{I}_n$. The transpose, conjugate transpose (Hermitian), and inverse of a matrix are denoted by the superscripts $(\cdot)^T$, $(\cdot)^H$, and $(\cdot)^{-1}$, respectively. The expectation operator is denoted by $\mathbb{E}[\cdot]$. The vector or matrix norm is written as $\| \cdot \|$ and the Kronecker product is denoted by $\otimes$. The symbols $\lfloor \cdot \rceil$, $\lfloor \cdot \rfloor$, and $\lceil \cdot \rceil$ are used to define the round, floor, and ceil operations, respectively.

1.3 Thesis Outline

This thesis is structured as a collection of papers and is divided into two main parts. Part I provides the overall introduction and background, while Part II contains the included publications. The organization of Part I is as follows. Chapter 2 introduces the fundamentals of wireless communications, covering topics such as MIMO systems and duplexing schemes. Chapter 3 discusses the foundations of deep learning, with a focus on different neural network architectures. Chapter 4 addresses the core problems investigated in this thesis, namely CSI estimation, compression, and prediction. Finally, Chapter 5 summarizes the appended publications and outlines potential directions for future research.

CHAPTER 2

An Introduction to Wireless Communications

Wireless communication concerns the reliable transmission of information over the air through a medium that is inherently time-varying, unpredictable, and shared among many users. Over the past decades, the field has advanced from simple point-to-point links to highly complex networks that operate across multiple frequency bands and employ sophisticated signal processing techniques. With each new generation of cellular technology, the demands on capacity, latency, and reliability have become more stringent, driving research and industry toward new spectrum resources, advanced antenna architectures, and innovative transmission schemes.

This chapter introduces the fundamental principles of wireless communications that form the basis for the subsequent discussions on CSI estimation, compression, and prediction. We begin by presenting MIMO systems and their integration with orthogonal frequency-division multiplexing (OFDM), followed by an overview of massive MIMO technology. The chapter then examines duplexing schemes—time-division duplexing (TDD), frequency-division duplexing (FDD), and full-duplex (FD)—which determine how uplink and downlink transmissions are organized in practical wireless systems.

2.1 Multi-antenna Systems

2.1.1 MIMO

One of the most significant breakthroughs in wireless communications has been the development of MIMO technology. By employing multiple antennas at both the transmitter and receiver, MIMO systems exploit the spatial dimension of the wireless channel in addition to the conventional time and frequency resources. This enables three fundamental benefits: spatial diversity, which improves reliability by mitigating fading; spatial multiplexing, which enhances throughput by transmitting multiple data streams simultaneously; and beamforming, which increases link quality and reduces interference by steering energy in desired directions [13].

The input–output relation of a narrowband MIMO system with N_t transmit antennas and N_r receive antennas can be expressed as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{2.1}$$

where $\mathbf{y} \in \mathbb{C}^{N_\mathrm{r}\times 1}$ is the received signal vector, $\mathbf{x} \in \mathbb{C}^{N_\mathrm{t}\times 1}$ is the transmitted symbol vector, $\mathbf{H} \in \mathbb{C}^{N_\mathrm{r}\times N_\mathrm{t}}$ is the channel matrix whose entries capture the fading coefficients between transmit and receive antennas, and $\mathbf{n} \sim \mathcal{CN}(0, \sigma^2\mathbf{I}_{N_\mathrm{r}})$ denotes additive white Gaussian noise (AWGN).

To fully exploit the spatial gains offered by MIMO, transmission is typically accompanied by a precoding matrix at the transmitter and a combining matrix at the receiver. Let $\mathbf{s} \in \mathbb{C}^{N_\mathrm{s}\times 1}$ denote the data symbol vector with $N_\mathrm{s} \leq \min(N_\mathrm{t}, N_\mathrm{r})$ streams, $\mathbf{F} \in \mathbb{C}^{N_\mathrm{t}\times N_\mathrm{s}}$ the precoding matrix, and $\mathbf{W} \in \mathbb{C}^{N_\mathrm{r}\times N_\mathrm{s}}$ the combining matrix. The received signal after combining is then

$$\mathbf{y}_\mathrm{eff} = \mathbf{W}^H\mathbf{H}\mathbf{F}\mathbf{s} + \mathbf{W}^H\mathbf{n}, \tag{2.2}$$

which highlights how the design of $\mathbf{F}$ and $\mathbf{W}$ determines the system's ability to exploit spatial diversity, achieve spatial multiplexing gains, and perform directional beamforming.

The design of the precoding and combining matrices relies on accurate knowledge of the underlying wireless channel. In particular, beamforming and spatial multiplexing strategies are only effective if the transmitter and receiver have access to sufficiently precise CSI. CSI captures the fading characteristics between each transmit–receive antenna pair and thus determines

how signals should be spatially processed to maximize throughput, improve reliability, or suppress interference.

2.1.2 MIMO–OFDM

In practical wireless channels, especially at high carrier frequencies or in rich multipath environments, the assumption of frequency-flat fading no longer holds. Instead, the channel exhibits frequency selectivity due to multipath propagation, where signals arriving from different paths experience distinct delays and attenuations, leading to inter-symbol interference (ISI). To efficiently combat this impairment, OFDM has become a core modulation scheme in modern broadband standards such as IEEE 802.11, LTE, and 5G NR [8], [11], [19].

When combined with MIMO techniques, OFDM enables spatial and frequency domain processing, forming the foundation of high-throughput, spectrally efficient wireless systems. In a MIMO–OFDM system with N_{t} transmit antennas, N_{r} receive antennas, and N_{c} subcarriers, the frequency-selective channel is divided into N_{c} parallel flat-fading subchannels in the frequency domain. Assuming the use of a cyclic prefix (CP) longer than the maximum delay spread, the input–output relation at the k-th subcarrier ($k = 0, 1, \ldots, N_{\mathrm{c}} - 1$) can be expressed as

$$\mathbf{y}[k] = \mathbf{H}[k]\mathbf{x}[k] + \mathbf{n}[k], \tag{2.3}$$

where $\mathbf{y}[k] \in \mathbb{C}^{N_{\mathrm{r}} \times 1}$ and $\mathbf{x}[k] \in \mathbb{C}^{N_{\mathrm{t}} \times 1}$ denote the received and transmitted symbol vectors at subcarrier k, respectively. The matrix $\mathbf{H}[k] \in \mathbb{C}^{N_{\mathrm{r}} \times N_{\mathrm{t}}}$ represents the frequency-domain channel response, and $\mathbf{n}[k] \sim \mathcal{CN}(0, \sigma^2 \mathbf{I}_{N_{\mathrm{r}}})$ is AWGN at subcarrier k.

The combination of OFDM and MIMO thus merges the benefits of both domains: OFDM simplifies equalization by converting a frequency-selective channel into parallel subcarriers, while MIMO introduces spatial multiplexing and beamforming on each subcarrier. Consequently, wideband MIMO–OFDM systems achieve high spectral efficiency (SE), robustness against multipath fading, and flexible resource allocation across space, time, and frequency [20], [21].

2.1.3 Massive MIMO

As wireless networks have evolved, the idea of scaling up antenna arrays has given rise to massive MIMO, where the BS is equipped with tens or even hundreds of antennas. In this regime, the spatial domain can be exploited more aggressively, enabling simultaneous service of a large number of users in the same time–frequency resource. Massive MIMO benefits from two key phenomena that emerge when the number of antennas becomes large. The first is favorable propagation, whereby user channels tend to become nearly orthogonal, thus simplifying multiuser interference management. The second is channel hardening, in which the effects of small-scale fading average out across many antennas, resulting in deterministic effective channels with reduced uncertainty.

From a statistical viewpoint, both favorable propagation and channel hardening stem from the law of large numbers and concentration of measure in high dimensions. As the number of antennas grows, inner products between distinct user channels vanish (favorable propagation), while each user's channel power becomes nearly deterministic (channel hardening). Together, these effects allow relatively simple linear signal processing schemes, such as maximum-ratio combining (MRC) or zero-forcing (ZF), to deliver near-optimal performance with reduced computational burden [21].

The transition to massive MIMO also highlights the importance of high-dimensional channel models. When the number of antennas at the BS grows to the order of hundreds, the corresponding channel matrices become extremely large, often spanning dimensions of hundreds by tens in multi-user scenarios. While this increase in dimensionality creates opportunities for improved spatial multiplexing and interference suppression, it also poses significant challenges for channel acquisition. Accurate CSI is essential to fully exploit the potential of massive MIMO systems, yet the required pilot overhead and feedback grow prohibitively with the number of antennas. Nevertheless, high-dimensional channels are not entirely unstructured; they often exhibit rich correlation and sparsity across the spatial, frequency, and time domains. Exploiting these properties is essential for developing efficient estimation, compression, and prediction techniques.

The relevance of massive MIMO becomes even more pronounced in the context of millimeter-wave (mmWave) communications, operating typically in the 30–300 GHz frequency range. At such high frequencies, the free-space path

loss increases dramatically, and the use of large antenna arrays is necessary to achieve sufficient link budget through sufficient antenna array aperture and highly directional beamforming. Moreover, mmWave propagation differs significantly from that in sub-6 GHz bands. The channels are typically sparse, since only a limited number of propagation paths exist due to the poor penetration and reflection characteristics of mmWave signals. At the same time, mmWave channels are highly sensitive to blockage, leading to sudden degradation of the link when obstacles obstruct the line-of-sight (LoS) or strong non-line-of-sight (NLoS) paths. These characteristics impose new requirements on channel estimation and tracking, while also enabling the use of structured models and advanced DL approaches for CSI acquisition [22].

In summary, MIMO and especially massive MIMO technologies represent a cornerstone of modern and future wireless networks. Their combination with mmWave frequencies enables unprecedented spectral and energy efficiency, but also brings to the forefront the challenges of high-dimensional channel acquisition. Addressing these challenges is essential for unlocking the full potential of next-generation wireless communication systems.

2.2 Duplexing Schemes

A fundamental design aspect of wireless systems is the method used to separate uplink and downlink transmissions, determined by the duplexing scheme. The most common approaches are TDD and FDD, while FD has emerged as a promising yet challenging alternative. Each scheme presents unique trade-offs in terms of latency, SE, hardware complexity, and CSI acquisition. The time–frequency allocation of these duplexing schemes is illustrated in Figure 2.1. We also summarize the main characteristics of FDD, TDD, and FD systems, highlighting their advantages, disadvantages, and practical challenges in Table 2.1.

2.2.1 TDD

In TDD systems, uplink and downlink transmissions occupy different time slots within the same frequency band. A key advantage is that channel reciprocity can be exploited, allowing the BS to infer the downlink channel directly from uplink pilots without explicit CSI feedback. This makes TDD

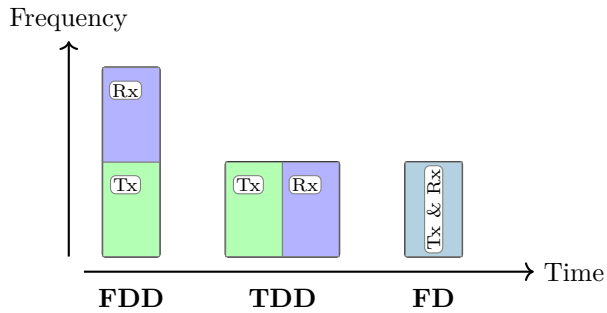


Figure 2.1: Illustration of FDD, TDD, and FD in the time–frequency plane.

Table 2.1: Comparison of key characteristics of FDD, TDD, and FD duplexing schemes.

	FDD	**TDD**	**FD**
Spectrum usage	High	Low	Low
Complexity	High	Low	Very high
Cost	High	Low	Very high
Latency	Low	High	Very low
Coverage	High	Low	Very low
Main challenge	CSI feedback	Synchronization	SI cancellation

highly scalable to massive MIMO and mmWave systems. Moreover, TDD supports flexible allocation of resources between uplink and downlink, which is valuable in scenarios with asymmetric traffic patterns.

The main drawbacks of TDD are related to timing and synchronization. Guard periods are required to separate uplink and downlink transmissions, which reduce SE and introduce latency, especially in wide-area deployments with long propagation delays. In addition, neighboring cells must be carefully synchronized to avoid cross-link interference, where one cell transmits in downlink while an adjacent cell is in uplink. Although the hardware complexity and cost of TDD are generally lower than FDD or FD, its coverage is more limited due to the guard periods and synchronization requirements.

2.2.2 FDD

In FDD systems, uplink and downlink use separate frequency bands and operate simultaneously. This scheme avoids guard periods and thus achieves very low latency, making it attractive for delay-sensitive applications. It also provides wide coverage, since uplink and downlink are continuously active and do not require tight synchronization across cells.

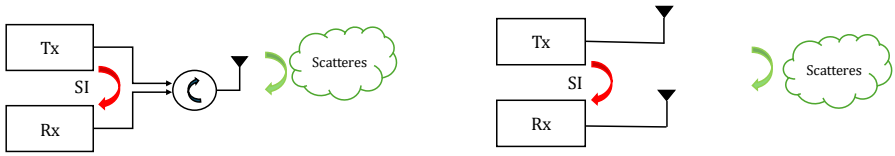
However, FDD has several limitations in modern high-dimensional systems. Because channel reciprocity does not hold across frequencies, CSI must be estimated at the UE and fed back to the BS, leading to significant overhead in massive MIMO deployments. Hardware complexity and cost are higher due to the need for duplex filters and diplexers that separate the uplink and downlink frequency bands. Furthermore, FDD lacks flexibility in allocating asymmetric uplink and downlink bandwidths, which may be inefficient in traffic scenarios dominated by downlink or uplink data.

2.2.3 FD

FD pushes the duplexing concept further by enabling simultaneous uplink and downlink transmission on the same frequency band. In theory, this can double SE compared to TDD and FDD, and it also minimizes latency since transmission and reception occur continuously. The main challenge of FD operation lies in self-interference (SI), where a device's own transmission can be several orders of magnitude stronger than the desired received signal. Although advanced analog and digital cancellation techniques can suppress a large portion of SI, residual interference remains a critical bottleneck. This makes FD systems highly complex and expensive, requiring sophisticated radio frequency (RF) circuitry and signal processing.

CSI estimation in FD systems is particularly challenging, as the SI channel must be estimated jointly with the users' channels, further aggravating the pilot overhead problem. Moreover, coverage tends to be more limited than in FDD or TDD systems, since the strong SI can easily overpower the weak signals arriving from distant users. Consequently, FD operation remains largely experimental, with ongoing research investigating its feasibility in massive MIMO deployments.

FD transceivers can be realized using two different antenna configurations as depicted in Figure 2.2: shared-antenna and separate-antenna. In the shared-



(a) Shared antenna configuration **(b)** Separate antennas configuration

Figure 2.2: Comparison of FD antenna configurations: (a) shared antenna and (b) separate antennas.

antenna configuration, the transmitter and receiver use the same antenna, typically separated by a circulator or a hybrid coupler. The main advantage of this design is compactness and hardware simplicity, since only one antenna array is required. However, the transmit leakage directly couples into the receive chain, resulting in extremely strong SI that is difficult to cancel completely. This places stringent requirements on analog cancellation and RF isolation.

In contrast, the separate-antenna configuration employs dedicated transmit and receive antennas, or sub-arrays. Physical separation between antennas provides inherent passive suppression of SI, which reduces the burden on subsequent analog and digital cancellation stages. The tradeoff, however, is increased hardware cost and size, as well as potential coupling effects when the arrays are closely spaced, especially in compact devices. Separate-antenna designs are often preferred in massive MIMO and BS deployments, where space and hardware cost are less restrictive, and higher SI suppression is needed.

Both configurations still require multi-stage SI suppression, combining passive isolation, analog cancellation, and digital cancellation, but the choice between shared and separate antennas strongly influences the achievable suppression level and the overall system complexity.

CHAPTER 3

A Preliminary to Deep Learning

In recent years, DL has become one of the most transformative branches of artificial intelligence (AI) and machine learning (ML). It has revolutionized numerous domains—such as computer vision, natural language processing, and speech recognition—by enabling machines to automatically learn complex patterns and hierarchical representations directly from data. In the context of communication systems, DL offers an alternative to conventional signal processing methods, which are typically built upon explicit mathematical models. Although such models have been highly effective in many scenarios, they necessarily rely on simplifying assumptions that only approximate the underlying reality. DL, by contrast, can learn directly from data and capture complex structures without imposing strong prior assumptions. This opens the door to tackling tasks that were previously constrained by oversimplified models and limited analytical tractability. This chapter introduces the key concepts, architectures, and generative AI models that form the foundation of this rapidly evolving field.

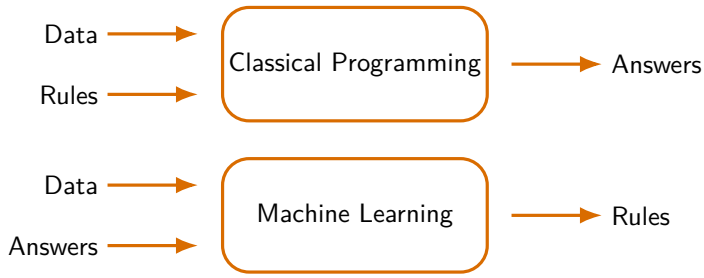


Figure 3.1: Comparison of classical programming and machine learning. In classical programming, rules and data are combined to compute answers, whereas in machine learning, data and answers are used to infer the rules.

3.1 AI, ML, DL

AI is the broadest field concerned with designing computational systems capable of performing tasks that normally require human intelligence, such as reasoning, planning, perception, and decision-making [23]. Within this hierarchy, ML represents a major subfield that focuses on algorithms which improve their performance by learning from data rather than relying solely on explicit rules [24]. DL, in turn, is a specialized branch of ML that uses multi-layer neural networks (NNs) to automatically extract hierarchical representations from raw or lightly processed inputs [25], [26].

While AI encompasses a wide range of symbolic and rule-based approaches, ML centers on statistical methods that enable systems to generalize from observed data. This fundamental distinction can be further understood by contrasting classical programming with ML, as illustrated in Figure 3.1. In classical programming, human experts define explicit rules which, together with data, are executed by the computer to produce answers. In ML, this process is inverted: both the data and the desired answers (labels) are provided to the algorithm, which then infers the underlying rules automatically. This paradigm shift underpins the power of ML, as systems no longer depend solely on hand-crafted rules but instead discover patterns directly from examples.

DL extends these capabilities by introducing highly expressive function classes: stacked layers of linear transformations and nonlinear activations that allow the learning of increasingly abstract features. This depth has proven essential for solving problems with high-dimensional and unstructured data.

The impact of DL has been most visible in areas such as computer vision, natural language processing, and speech recognition, where it has dramatically outperformed traditional hand-crafted feature methods. For example, convolutional networks underpin modern image recognition, recurrent and attention-based architectures have transformed machine translation, and end-to-end models now dominate speech-to-text applications. These successes illustrate DL's ability to model complex input–output relationships directly from raw data.

3.2 Fundamentals of NNs

At the core of DL lies the artificial NN, a parametric function inspired by the interconnected structure of biological neurons [25], [27]. The basic computational unit is the *perceptron*, which performs a weighted summation of its inputs followed by a nonlinear transformation. For an input vector $\mathbf{x} \in \mathbb{R}^d$, the output of a perceptron is expressed as

$$y = \sigma\big(\mathbf{w}^\top \mathbf{x} + b\big)\,, \tag{3.1}$$

where $\mathbf{w} \in \mathbb{R}^d$ is a learnable weight vector, $b \in \mathbb{R}$ is a bias term, and $\sigma(\cdot)$ denotes a nonlinear activation function. Popular choices include the logistic sigmoid, the hyperbolic tangent, and the rectified linear unit (ReLU), each introducing nonlinearity that enables the network to approximate nonlinear functions [26], [28].

By stacking multiple perceptrons, one obtains a *layer*. A network consisting of an input layer, one or more hidden layers, and an output layer is commonly referred to as a multi-layer perceptron (MLP), illustrated in Figure 3.2. The forward computation of an MLP with L layers can be expressed compactly as

$$\mathbf{h}^{(l)} = \sigma\Big(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\Big)\,, \quad l = 1, \ldots, L, \tag{3.2}$$

where $\mathbf{h}^{(0)} = \mathbf{x}$ is the input, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector of layer l, and $\mathbf{h}^{(L)}$ produces the network output. The nonlinearity applied at each layer is essential: without it, a stack of linear layers would collapse into a single linear transformation, offering no advantage over a simple linear model.

One of the most notable results in this area is the *universal approximation*

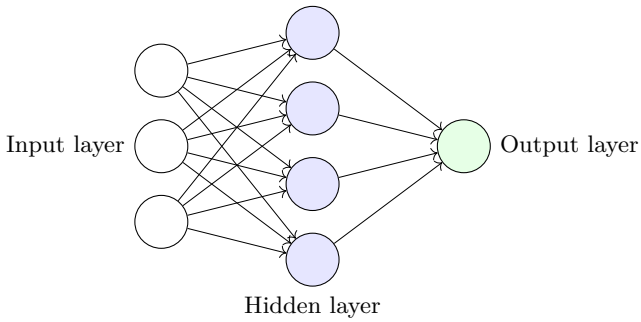


Figure 3.2: Schematic of a feedforward NN with one hidden layer.

theorem, which states that even a single hidden layer network with sufficient width can approximate any continuous function on a compact domain to arbitrary accuracy [29], [30]. While this theoretical guarantee is important, practical networks often employ many layers with moderate width to achieve scalability, representation learning, and computational efficiency. Depth allows networks to capture hierarchical structures in data: earlier layers extract low-level features, while deeper layers capture increasingly abstract patterns.

3.3 Training Deep Networks

NNs become effective only through the process of training, where their parameters are adjusted to minimize a task-specific objective. Training typically relies on a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ composed of input–output pairs. Depending on the nature of the target, supervised problems are generally categorized as either *regression* or *classification*. Regression tasks involve predicting a continuous-valued quantity, such as a channel coefficient or a physical parameter, whereas classification tasks involve assigning an input to one of several discrete classes.

In supervised learning, the goal is to find parameters $\boldsymbol{\theta}$ that minimize the discrepancy between the network predictions $f_{\boldsymbol{\theta}}(\mathbf{x}_i)$ and the corresponding labels y_i. This discrepancy is quantified by a *loss function*. For regression

tasks, a common choice is the mean squared error (MSE)

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \| f_{\boldsymbol{\theta}}(\mathbf{x}_i) - y_i \|_2^2 \,, \tag{3.3}$$

while classification problems typically employ the cross-entropy loss. For a model producing class probabilities $p_\theta(y_i \mid \mathbf{x}_i)$, the cross-entropy loss is defined as

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \log p_\theta(y_i \mid \mathbf{x}_i). \tag{3.4}$$

To minimize the loss, networks are trained using variants of stochastic gradient descent (SGD). Parameters are updated iteratively according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \, \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}), \tag{3.5}$$

where η is the learning rate controlling the step size, and $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ is the gradient of the loss with respect to the parameters. Gradients are computed efficiently using the backpropagation algorithm [31], which applies the chain rule across the network layers.

A variety of optimization algorithms build upon SGD, including momentum-based updates, RMSProp, and Adam [32]. These methods adapt learning rates or incorporate gradient history, often accelerating convergence in practice. Selecting appropriate hyperparameters (e.g., learning rate schedule, batch size, weight initialization) is crucial to achieving stable training.

Training also requires strategies to avoid overfitting, where the network memorizes the training data but fails to generalize to unseen examples. Common regularization techniques include weight decay, dropout [33], and batch normalization [34], all of which encourage more robust representations. Another essential aspect is the partitioning of data into training, validation, and test sets, ensuring that performance reflects generalization rather than memorization.

3.4 NN Architectures

While MLPs represent the most basic form of NN, they are limited in their ability to scale to complex, high-dimensional, and structured data. The evo-

lution of DL has therefore been marked by the development of specialized architectures designed to exploit particular structures present in the input. For example, many data modalities exhibit strong spatial, temporal, or sequential patterns that cannot be efficiently captured by fully connected layers alone.

Over the past decade, several families of architectures have become foundational across a wide range of DL applications. Convolutional neural networks (CNNs) leverage local connectivity and weight sharing to process spatially structured data, making them central to breakthroughs in computer vision. Recurrent neural networks (RNNs) and their gated variants, such as long short-term memory (LSTM) networks and gated recurrent units (GRUs), are tailored for sequential modeling and have proven highly effective in natural language processing and speech recognition. More recently, attention-based models such as the Transformer have reshaped the field by enabling efficient modeling of long-range dependencies, leading to state-of-the-art performance in tasks from machine translation to large-scale language modeling.

These architectures, while distinct in design, share the common goal of building representations that align with the structure of the data they process. By tailoring the inductive biases of the model to the modality of interest—whether spatial, temporal, sequential, graph, or generative—deep learning achieves scalability, efficiency, and state-of-the-art accuracy across domains such as vision, language, and audio processing. The remainder of this section provides a concise overview of these important architectures and the principles that underpin their success.

3.4.1 CNNs

CNNs are among the most influential architectures in DL. They were first popularized through their success in handwritten digit recognition [35] and later revolutionized computer vision with large-scale image classification [14]. The key idea is to leverage the local spatial structure of data by applying learnable filters (kernels) that slide across the input, as illustrated in Figure 3.3.

Mathematically, a convolutional layer applies a filter $\mathbf{K} \in \mathbb{R}^{k \times k}$ to an input feature map $\mathbf{X} \in \mathbb{R}^{H \times W}$ to produce an output feature map $\mathbf{Y} \in \mathbb{R}^{H' \times W'}$

$$Y_{i,j} = \sum_{m=1}^{k} \sum_{n=1}^{k} K_{m,n} \, X_{i+m-1,j+n-1}, \tag{3.6}$$

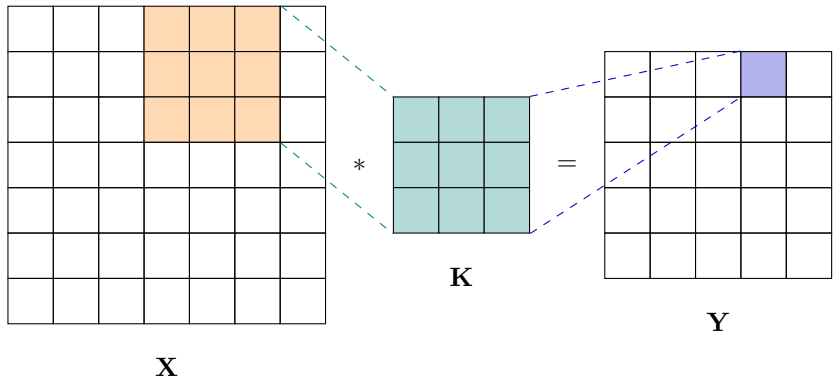


Figure 3.3: Illustration of a 2D convolution operation. The highlighted orange region in the input $\mathbf{X}$ corresponds to the receptive field covered by the convolution kernel $\mathbf{K}$. The resulting activation is placed in the blue-highlighted cell of the output feature map $\mathbf{Y}$. Dashed lines indicate how the kernel is applied to the input region and how the corresponding value propagates to the output.

where (i, j) indexes the spatial location of the output. In practice, multiple filters are learned, producing a set of feature maps that capture different local patterns. Each feature map is then passed through a nonlinear activation function, applied element-wise, to introduce nonlinearity. The convolution operation offers two key advantages: first, each neuron connects only to a small receptive field of the input, which significantly reduces the number of parameters compared to fully connected layers; second, the same filter is applied across all positions, yielding translation invariance and efficient parameter sharing.

By stacking multiple convolutional layers, the network learns a hierarchy of representations: early layers extract primitive features, while deeper layers combine them into more abstract features. Pooling layers, often interleaved with convolutions, reduce spatial resolution while retaining salient features, contributing to invariance against small input shifts.

In wireless communications, CSI matrices often resemble images, with antennas or subcarriers corresponding to spatial dimensions and fading coefficients acting as pixel intensities. CNNs are therefore naturally suited to process CSI by capturing local correlations in space or frequency. For example, neighboring subcarriers in an OFDM system experience correlated fading

due to multipath propagation, a structure that convolutional filters can efficiently exploit [36], [37]. This makes CNNs attractive for tasks such as channel estimation, denoising, and compression.

3.4.2 RNNs

Many types of data are inherently sequential: speech, text, time-series, and motion trajectories all consist of ordered observations where the present depends on the past. Standard feedforward networks process inputs independently and therefore cannot naturally capture such temporal dependencies. RNNs were introduced to address this limitation by incorporating a hidden state that evolves over time, allowing information from earlier steps in a sequence to influence later computations [38].

Given an input sequence $\{\mathbf{x}_t\}_{t=1}^{T}$, the hidden state $\mathbf{h}_t$ and output $\mathbf{y}_t$ of a simple RNN are updated as

$$\mathbf{h}_t = \phi(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)\,, \tag{3.7}$$

$$\mathbf{y}_t = f(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y)\,, \tag{3.8}$$

where $\mathbf{W}_{xh}, \mathbf{W}_{hh}, \mathbf{W}_{hy}$ are weight matrices, $\mathbf{b}_h, \mathbf{b}_y$ are bias terms, $\phi(\cdot)$ is a nonlinear activation such as tanh or ReLU, and $f(\cdot)$ is an output mapping. The recursive dependence on $\mathbf{h}_{t-1}$ enables the network to carry information across time steps.

Training RNNs typically involves backpropagation through time, which unfolds the recurrence into a deep computational graph as depicted in Figure 3.4. A major difficulty with this approach is the vanishing and exploding gradient problem [39], where gradients either shrink or grow exponentially with sequence length, preventing the model from capturing long-range dependencies. To address these limitations, more advanced architectures such as LSTMs and GRUs were proposed. These models incorporate gating mechanisms—such as forget, input, and output gates in LSTMs, and update and reset gates in GRUs—to better control the flow of information and mitigate gradient-related instabilities. While LSTMs and GRUs improve performance for moderately long sequences, they still rely on sequential processing, which limits their scalability to long sequences and makes them prone to error propagation. In particular, their training and inference times grow linearly with the sequence length, which can become a bottleneck for large-scale or real-time

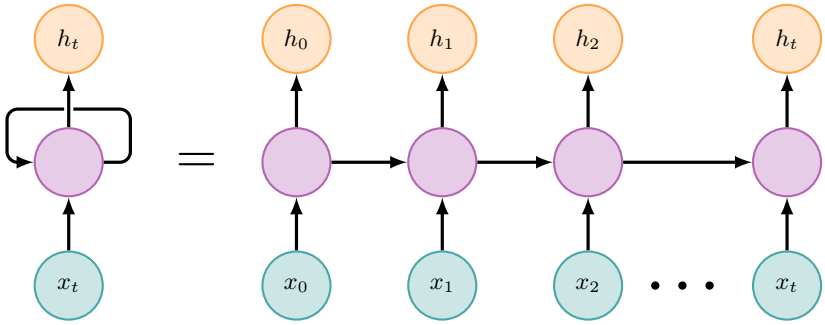


Figure 3.4: Unrolled recurrent neural network. Each hidden state $\mathbf{h}_t$ depends on the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$.

applications.

3.4.3 Autoencoders

Autoencoders (AEs) [26], [40], [41] constitute a class of NN architectures designed to learn compact representations of data through unsupervised training. The model is composed of two parts: an encoder, which maps the input to a lower-dimensional latent representation, and a decoder, which reconstructs the input from this representation. The central idea is to force the network to compress information while preserving essential structure, thereby learning features that capture the most salient characteristics of the data distribution.

Given an input $\mathbf{x} \in \mathbb{R}^d$, the encoder defines a mapping

$$\mathbf{z} = f_\phi(\mathbf{x}), \tag{3.9}$$

where $f_\phi : \mathbb{R}^d \to \mathbb{R}^k$ with $k < d$, parameterized by ϕ, and $\mathbf{z}$ is the latent representation. The decoder reconstructs the input from the latent variable via

$$\hat{\mathbf{x}} = g_\theta(\mathbf{z}), \tag{3.10}$$

where $g_\theta : \mathbb{R}^k \to \mathbb{R}^d$ with parameters θ. A schematic example of a simple autoencoder implemented with a few fully connected layers is shown in Figure 3.5. While this representation highlights the basic encoder–decoder structure with a latent bottleneck, in practice AEs can be designed with different layer types depending on the data modality. Despite these variations,

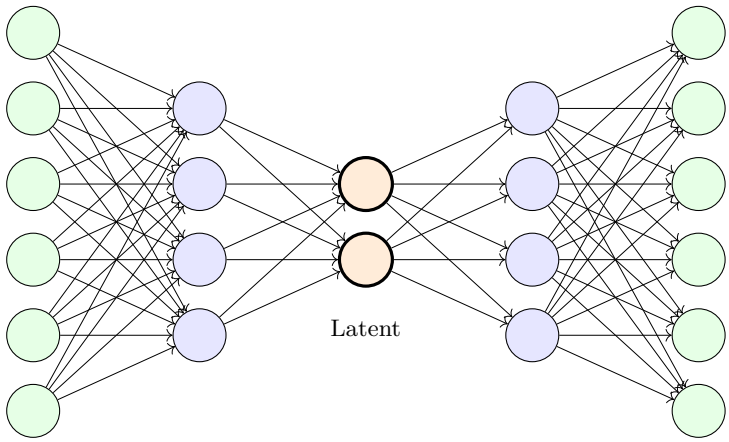


Figure 3.5: Architecture of a simple AE with three hidden layers. The input layer is mapped through two hidden transformations into a low-dimensional latent representation, which is then expanded back through a mirrored decoder to reconstruct the output.

the underlying principle remains the same: an encoder network compresses the input into a compact latent representation, and a decoder reconstructs the original input from this representation.

The bottleneck structure, where $k < d$, forces the model to discard redundancy and learn compressed features. This makes autoencoders useful not only for dimensionality reduction but also for denoising, anomaly detection, and as building blocks for more advanced generative models. Extensions of the basic autoencoder framework include denoising autoencoders [42], which are trained to reconstruct inputs from corrupted versions.

3.4.4 Attention

Attention mechanisms offer a flexible way for NNs to focus on the most informative parts of their input. Instead of relying on a fixed-size hidden state, as in recurrent models, or on a fixed receptive field, as in convolutional networks, attention retrieves information in a data-dependent manner. Given a query, the model computes how relevant each element of the input is and assigns weights accordingly. This ability to dynamically highlight important context has made attention a central tool in modern DL.

Self-attention, the core of the Transformer architecture [16], applies this principle within a sequence. Each element interacts with all others to build a contextualized representation that captures both local and long-range relationships. This is particularly advantageous because every position can directly access information from all others in a single step, enabling efficient modeling of global structure that would otherwise require many layers in RNNs or CNNs. Given an input sequence

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times d},$$

the model first projects it into three learned subspaces—query, key, and value:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V,$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_h}$ are learned parameters. The self-attention operation is then computed as

$$\text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\right)\mathbf{V},$$

where the dot products measure the similarity between each query–key pair, the softmax converts these similarities into attention weights, and the scaling factor $\sqrt{d_h}$ prevents overly large values that could destabilize training. Intuitively, each query compares itself with all keys to determine which positions are most relevant; the resulting weighted combination of value vectors gives each token direct access to information from the entire sequence.

To increase expressiveness, Transformers employ multi-head attention, which computes several attention operations in parallel using different learned projections. Each head can specialize in capturing certain patterns—such as short-range interactions, long-range dependencies, or structural relationships. The outputs of all heads are concatenated and linearly projected back to the model dimension, allowing the network to integrate diverse contextual cues.

A key challenge in Transformers is the quadratic complexity of self-attention in the sequence length, which limits scalability for long inputs. To address this, several efficient variants have been proposed. LinFormer [43] employs low-rank projections of keys and values to reduce computational burden, while Performer [44] introduces kernel-based approximations that make attention computation linear in N. Hierarchical designs such as the Swin Transformer [45]

restrict attention to shifted local windows, enabling scalable modeling of high-dimensional structured inputs. These developments highlight the adaptability of the Transformer framework across diverse domains and data modalities.

3.5 Generative AI

Generative AI refers to a class of ML methods that model the underlying probability distribution of data in order to synthesize new samples that resemble those drawn from the true distribution. Unlike discriminative models, which learn a conditional mapping $p(\mathbf{y} \mid \mathbf{x})$ from inputs to outputs, generative models learn the joint distribution $p(\mathbf{x}, \mathbf{y})$ or the marginal distribution $p(\mathbf{x})$, thereby enabling the creation of novel and realistic data. These models form the foundation for tasks such as image synthesis, text generation, speech synthesis, and representation learning, and have become a central paradigm in modern AI era [46], [47].

The fundamental goal of generative modeling is to approximate the data distribution $p_{\text{data}}(\mathbf{x})$ using a parameterized model distribution $p_\theta(\mathbf{x})$. Given a dataset $\{\mathbf{x}^{(i)}\}_{i=1}^N$, the model parameters θ are typically learned by maximum likelihood estimation (MLE)

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)}). \tag{3.11}$$

Since direct computation of the likelihood is often intractable, different families of generative models have been proposed depending on how the distribution is represented and optimized.

Generative modeling has given rise to several distinct families of methods, each grounded in different theoretical principles and offering complementary strengths. These models vary in how they represent probability distributions, how they are trained, and what trade-offs they make between likelihood estimation, sample quality, training stability, and computational efficiency. In the following subsections, we present four representative classes of generative models that have shaped modern deep learning.

3.5.1 AR Models

Autoregressive (AR) models are one of the earliest and most principled approaches to generative modeling. They rely on the chain rule of probability to factorize the joint distribution of a data vector $\mathbf{x} = (x_1, \ldots, x_d)$ into a product of conditional distributions

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{d} p_\theta(x_i \mid x_{<i}), \tag{3.12}$$

where $x_{<i} = (x_1, \ldots, x_{i-1})$. This decomposition allows exact likelihood computation and straightforward training via maximum likelihood estimation.

In practice, neural networks are used to parameterize the conditionals $p_\theta(x_i \mid x_{<i})$. For discrete data, such as text or images, these distributions are typically modeled with categorical softmax outputs. Prominent examples include PixelRNN and PixelCNN [48], which model images pixel by pixel, and modern large language models such as GPT [49], [50], which predict text token by token. The training objective is the negative log-likelihood of the data

$$\mathcal{L}_{\mathrm{AR}}(\theta) = -\sum_{i=1}^{d} \log p_\theta(x_i \mid x_{<i}). \tag{3.13}$$

This objective ensures consistent maximum likelihood estimation of the data distribution.

The main advantage of autoregressive models is their tractable likelihoods and stable training. However, sample generation is inherently sequential, requiring d steps to generate a data point of dimension d. This makes them computationally expensive for high-dimensional modalities such as high-resolution images or long sequences, despite their excellent modeling accuracy.

3.5.2 VAEs

variational autoencoders (VAEs) [51], [52] introduce a latent-variable framework for generative modeling. They assume that data $\mathbf{x}$ is generated from latent variables $\mathbf{z}$ through a likelihood model $p_\theta(\mathbf{x} \mid \mathbf{z})$, with a prior distribu-

tion $p(\mathbf{z})$, typically Gaussian. The marginal likelihood is

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}, \tag{3.14}$$

which is generally intractable to compute directly.

To address this, VAEs introduce a variational approximation $q_\phi(\mathbf{z} \mid \mathbf{x})$ to the true posterior $p_\theta(\mathbf{z} \mid \mathbf{x})$. Training maximizes the evidence lower bound (ELBO):

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z}))\,. \tag{3.15}$$

The first term encourages faithful reconstruction of the data given latent variables, while the second term regularizes the approximate posterior toward the prior.

A crucial innovation in VAEs is the reparameterization trick [51], which allows gradients to propagate through stochastic latent variables. Specifically, one samples

$$\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \tag{3.16}$$

where $\boldsymbol{\mu}_\phi$ and $\boldsymbol{\sigma}_\phi$ are encoder outputs.

VAEs are attractive for their elegant probabilistic formulation, stable training, and efficient inference. However, they often produce blurrier samples compared to GANs or diffusion models, due to the Gaussian assumptions in the likelihood and posterior distributions.

3.5.3 GANs

generative adversarial networks (GANs) [46] approach generative modeling through an adversarial game between two neural networks: a generator G_θ and a discriminator D_ϕ. The generator maps latent noise $\mathbf{z} \sim p(\mathbf{z})$ to the data space as $G_\theta(\mathbf{z})$, while the discriminator outputs the probability that a sample comes from the true data distribution rather than the generator.

The training objective is a minimax game:

$$\min_\theta \max_\phi \; \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \left[\log D_\phi(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log\left(1 - D_\phi(G_\theta(\mathbf{z}))\right)\right]. \tag{3.17}$$

At equilibrium, the generator recovers the data distribution and the discrim-

inator cannot distinguish between real and generated samples.

GANs have been remarkably successful in producing sharp and realistic images, with numerous architectural innovations such as DCGANs, StyleGANs, and BigGANs pushing the boundaries of sample fidelity [53]–[55]. However, GANs are notoriously difficult to train due to instability and mode collapse, where the generator produces limited diversity in samples. Various improvements have been proposed, including Wasserstein GANs with gradient penalty (WGAN-GP) [56], spectral normalization [57], and progressively growing architectures [58]. Despite these challenges, GANs remain a cornerstone of generative AI, particularly when high visual fidelity is required.

3.5.4 Diffusion Models

Diffusion models [47], [59], [60] represent a recent class of generative models that have achieved state-of-the-art performance in image, audio, and video synthesis. Their success stems from a simple yet powerful idea: learn to reverse a gradual noising process that destroys structure in the data. By sequentially denoising samples, the model can generate realistic data from random noise while capturing complex distributions.

A diffusion model is defined through two stochastic processes: a forward (diffusion) process and a reverse (denoising) process. In the forward process, Gaussian noise is added to the data in small increments over T steps. Given a clean sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, the forward process produces a noisy sample $\mathbf{x}_t$ at step t according to

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\Big(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\Big)\,, \tag{3.18}$$

where $\beta_t \in (0,1)$ is a small variance parameter controlling the noise level. After T steps, $\mathbf{x}_T$ approaches an isotropic Gaussian distribution, effectively discarding all information from the original sample.

The reverse process aims to learn the generative model that maps noise back into data. Specifically, a parameterized model p_θ is trained to approximate the reverse Markov chain:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))\,. \tag{3.19}$$

Training is performed by minimizing a variational bound on the negative log-

likelihood, which reduces to a denoising score-matching objective [47]. In practice, the model learns to predict the noise $\boldsymbol{\epsilon}$ that was added at each step, leading to the simple training loss

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0,t,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right\|^2\right], \tag{3.20}$$

where $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$, with $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$.

Sampling proceeds by iteratively applying the learned reverse transitions, starting from pure Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ and gradually denoising it to produce a data sample $\mathbf{x}_0$. While the original denoising diffusion probabilistic model (DDPM) [47] performs this process over hundreds or thousands of steps, subsequent work has proposed improvements such as deterministic sampling (DDIM) [61] and score-based generative modeling with stochastic differential equations (SDEs) [60].

A key strength of diffusion models lies in their flexibility. The forward noising process is fixed and independent of the data, while all learning is concentrated on the reverse process. This decoupling provides stable training dynamics compared to adversarial generative models such as GANs [46]. Furthermore, diffusion models can be conditioned on additional information, such as class labels, text prompts, or other modalities, enabling controlled generation in a wide range of applications.

CHAPTER 4

CSI Acquisition

The performance of physical-layer processing in wireless communication depends critically on accurate knowledge of the CSI, which enables efficient resource allocation, adaptive modulation and coding, and beamforming. However, acquiring high-dimensional CSI in practice is not trivial, as it involves dealing with noisy measurements, hardware limitations, and rapidly time-varying propagation conditions. This chapter explores the problems of CSI estimation, compression, and prediction. CSI estimation concerns the inference of channel coefficients from pilot signals, while compression focuses on reducing the dimensionality of CSI data, and prediction leverages temporal correlations in the wireless channel to anticipate future CSI states. By treating CSI as a data modality in its own right, these tasks can increasingly benefit from DL techniques. Throughout this chapter, we discuss classical approaches alongside recent data-driven methods, highlighting both theoretical foundations and practical considerations.

4.1 CSI Estimation

In general, CSI can be obtained either through *blind* or *pilot-based* estimation techniques. Blind estimation methods attempt to infer the channel characteristics directly from the received payload data without relying on known pilot symbols [62], [63]. Although they can reduce pilot overhead, these methods typically rely on strong statistical assumptions—such as signal independence, constant modulus, or sparsity—and often involve solving non-convex optimization problems. As a result, blind estimation is computationally demanding and rarely feasible for real-time implementations. Consequently, this thesis focuses on pilot-based CSI estimation, which is the standard approach adopted in practical communication systems [64], [65].

In pilot-based estimation, known pilot symbols are transmitted for channel learning. Let $\mathbf{X}_{\mathrm{p}} \in \mathbb{C}^{N_{\mathrm{t}} \times N_{\mathrm{p}}}$ denote the pilot matrix transmitted from the BS with N_{t} antennas over N_{p} pilot resources. The received pilot signal at the receiver equipped with N_{r} antennas is modeled as

$$\mathbf{Y}_{\mathrm{p}} = \mathbf{H}\mathbf{X}_{\mathrm{p}} + \mathbf{N}, \tag{4.1}$$

where $\mathbf{Y}_{\mathrm{p}} \in \mathbb{C}^{N_{\mathrm{r}} \times N_{\mathrm{p}}}$ is the received pilot matrix, $\mathbf{H} \in \mathbb{C}^{N_{\mathrm{r}} \times N_{\mathrm{t}}}$ denotes the MIMO channel matrix, and $\mathbf{N}$ represents additive noise. The goal is to obtain an estimate $\widehat{\mathbf{H}}$ of the true channel $\mathbf{H}$ given the known pair $(\mathbf{Y}_{\mathrm{p}}, \mathbf{X}_{\mathrm{p}})$.

The main challenges in CSI estimation include the limited pilot resources available in time and frequency, which constrain the number of pilots that can be transmitted; the high dimensionality of the channel, particularly in massive MIMO systems; and the stringent computational complexity and latency requirements of real-time processing. Moreover, the statistical properties of practical channels are often unknown or non-stationary, making it difficult to design accurate model-based estimators. Hardware impairments such as carrier-frequency offset, phase noise, in-phase/quadrature (IQ) imbalance, and low-resolution quantization further degrade estimation accuracy. Finally, estimators trained or optimized for a specific frequency band, antenna configuration, or propagation environment often generalize poorly when the deployment conditions differ from those seen during training.

In TDD systems, uplink pilots are commonly used to estimate the uplink channel, and under reciprocity calibration, the downlink CSI can be inferred from the uplink channel, assuming the transmit and receive radio chains are

properly calibrated. In contrast, FDD systems rely on downlink pilot transmission, where the user estimates the downlink channel and subsequently feeds the CSI back to the BS. This feedback incurs substantial overhead, particularly in massive MIMO systems with large antenna arrays [66].

In FD systems, in addition to estimating the users' channels, the SI channel must also be accurately estimated to enable effective digital SI cancellation. The received pilot signal at an FD transceiver is contaminated by the strong SI component when the transmit antenna arrays share the same pilot resources as the users. Moreover, the SI channel itself is typically a high-dimensional MIMO channel, and jointly estimating it together with the users' channels significantly increases the computational complexity and pilot overhead of CSI estimation in FD systems.

Under the MSE criterion, the optimal CSI estimator is the *conditional mean estimator* (CME), defined as [67]

$$\widehat{\mathbf{H}}^{*} = \mathbb{E}[\,\mathbf{H} \mid \mathbf{Y}_{\mathrm{p}}\,], \tag{4.2}$$

which minimizes the MSE among all possible estimators [68]. By Bayes' theorem, the CME can be expressed as

$$\widehat{\mathbf{H}}^{*} = \int \mathbf{H}\, p(\mathbf{H} \mid \mathbf{Y}_{\mathrm{p}})\, d\mathbf{H} = \frac{\int \mathbf{H}\, p(\mathbf{Y}_{\mathrm{p}} \mid \mathbf{H})\, p(\mathbf{H})\, d\mathbf{H}}{p(\mathbf{Y}_{\mathrm{p}})}, \tag{4.3}$$

where $p(\mathbf{Y}_{\mathrm{p}} \mid \mathbf{H})$ denotes the likelihood function and $p(\mathbf{H})$ is the prior distribution of the channel. The CME, however, is generally intractable in practice, since the exact channel prior is unknown and the integral in (4.3) is high-dimensional and computationally prohibitive. Therefore, practical systems employ suboptimal but tractable estimators that balance estimation performance with computational complexity.

A simple and widely used approach is the *least-squares (LS)* estimator,

$$\widehat{\mathbf{H}}_{\mathrm{LS}} = \mathbf{Y}_{\mathrm{p}} \mathbf{X}_{\mathrm{p}}^{\dagger}, \tag{4.4}$$

which requires no prior statistical knowledge of the channel but is sensitive to noise and interference. When second-order channel statistics are available, the *linear minimum mean-squared error (LMMSE)* estimator provides improved performance by incorporating the channel covariance matrix [64], [69]. Let $\mathbf{h} = \mathrm{vec}(\mathbf{H})$ denote the vectorized channel with covariance matrix $\mathbf{R}_{\mathbf{h}}$ and

noise variance σ^2. The LMMSE estimate is then given by

$$\widehat{\mathbf{h}}_{\mathrm{MMSE}} = \mathbf{R}_{\mathbf{h}}\mathbf{X}_{\mathrm{p}}^{H}\left(\mathbf{X}_{\mathrm{p}}\mathbf{R}_{\mathbf{h}}\mathbf{X}_{\mathrm{p}}^{H} + \sigma^2\mathbf{I}\right)^{-1}\mathrm{vec}(\mathbf{Y}_{\mathrm{p}}), \tag{4.5}$$

which achieves an optimal trade-off between bias and noise suppression by leveraging the second-order statistics of the wireless channel. From a Bayesian perspective, the LMMSE estimator is a special case of the CME under the assumption of a Gaussian channel prior, and is therefore the optimal estimator for channels following a Gaussian distribution.

While linear estimators are computationally efficient, their performance is fundamentally limited by the assumptions of linearity, Gaussianity, and accurate knowledge of the second-order channel statistics. In realistic propagation environments, wireless channels often exhibit complex spatial–frequency correlations, non-Gaussian fading, and hardware impairments that violate these assumptions. Consequently, conventional estimators such as LS and LMMSE may yield suboptimal performance.

Recent advances in DL have enabled data-driven CSI estimation methods that learn the complex mapping between the received pilots and the underlying channel directly from data without requiring explicit analytical knowledge. Unlike traditional estimators that rely on simplified statistical assumptions, NNs can implicitly capture rich propagation characteristics such as spatial correlation, path sparsity, and frequency selectivity, leading to superior estimation accuracy in challenging propagation environments [70]–[72].

More recently, the focus has shifted toward generative learning approaches that aim to explicitly model the underlying distribution of wireless channels rather than directly mapping pilots to estimates. By learning a generative prior $p_\theta(\mathbf{H})$, these models approximate the unknown channel distribution required in the Bayesian formulation of (4.3). Early studies employed VAEs to learn low-dimensional latent representations that capture essential channel statistics, while GANs focused on learning a generator that maps simple latent priors to realistic channel realizations through adversarial training. In contrast, diffusion-based generative models leverage score-based or denoising diffusion probabilistic modeling to implicitly capture complex and multimodal channel structures without relying on explicit analytical priors [73]–[77]. These advances underscore the promise of generative AI as a powerful new paradigm for channel estimation—effectively bridging the gap between Bayesian optimal inference and practical deep learning implementations.

Despite their advantages, DL–based CSI estimators introduce new challenges related to data collection, generalization, and computational cost. The networks must be trained with large, high-quality datasets that capture the variability of practical channels, and their performance may degrade when deployed in unseen environments. Moreover, NNs require additional computational resources and careful design to meet real-time inference constraints. Nevertheless, by approximating the optimal Bayesian estimator in a data-driven manner, DL–based CSI estimation has emerged as a powerful paradigm that complements and often surpasses classical model-based methods.

4.2 CSI Compression

As discussed in Section 2.2.2, in FDD systems, the downlink and uplink operate on different frequency bands, which breaks the channel reciprocity. Consequently, the BS must transmit downlink pilots, and each UE must estimate its downlink CSI locally and feed it back to the BS. This feedback process introduces substantial communication overhead, particularly in massive MIMO systems and wideband channels, where the downlink CSI is represented by high-dimensional matrices. To mitigate this overhead, the downlink CSI must be represented in a more compact form before transmission to the BS. This process, referred to as CSI compression, aims to reduce the number of bits required to represent the CSI while retaining as much relevant information as possible for accurate beamforming and precoding.

In general, data compression seeks to encode information using as few bits as possible while preserving the essential content. Compression techniques can be broadly classified as *lossless* or *lossy*. Lossless compression enables exact recovery of the original data, which is typically infeasible for high-dimensional and continuous-valued CSI due to its limited redundancy. In contrast, lossy compression allows small reconstruction errors in exchange for substantial bit-rate savings, making it the practical choice for CSI feedback. The fundamental trade-off between compression efficiency and reconstruction accuracy is captured by rate–distortion (RD) theory, which provides a theoretical framework for analyzing and optimizing lossy compression systems.

Building on this principle, CSI compression can be modeled as a *lossy transformation process* between two communicating entities. Specifically, an *encoder* f_ϕ at the UE compresses the estimated CSI into a compact latent rep-

resentation suitable for feedback, while a *decoder* g_θ at the BS reconstructs the CSI from the received representation. Mathematically, this process can be expressed as

$$\widehat{\mathbf{H}} = g_\theta\big(f_\phi(\mathbf{H})\big),$$

where $\widehat{\mathbf{H}}$ denotes the reconstructed channel matrix. For a given distortion measure $d(\mathbf{H}, \widehat{\mathbf{H}})$, e.g., normalized mean squared error (NMSE), the lossy CSI compression problem can be formulated as

$$\min_{f_\phi,\, g_\theta} \ \mathbb{E}\Big[d\Big(\mathbf{H},\, \widehat{\mathbf{H}}\Big)\Big] \quad \text{s.t.} \quad \mathbb{E}[\ell(f_\phi(\mathbf{H}))] \le R, \tag{4.6}$$

where $\ell(\cdot)$ denotes the number of feedback bits and R is the total feedback budget. The constrained problem in (4.6) can be equivalently expressed in its Lagrangian form as

$$\min_{f_\phi,\, g_\theta} \ \mathbb{E}\Big[d\Big(\mathbf{h},\, \widehat{\mathbf{h}}\Big)\Big] + \lambda\, \mathbb{E}[\ell(f_\phi(\mathbf{h}))]\,, \tag{4.7}$$

where $\lambda \ge 0$ controls the trade-off between feedback rate and reconstruction fidelity.

Several classical approaches have been proposed in the literature to reduce the feedback overhead in FDD systems by exploiting the inherent structure and sparsity of the CSI. Early methods relied on codebook-based quantization, where each UE selects the most suitable precoding vector or channel representation from a predefined finite codebook, such as those standardized in LTE and 5G NR [78], [79]. Seminal works on limited-feedback MIMO [80] analyzed the design of Grassmannian and random vector quantization codebooks, and later extensions considered multiuser and correlated-channel scenarios [81]. While conceptually simple and easy to implement, codebook-based schemes suffer from poor scalability in massive MIMO systems, since the required codebook size grows exponentially with the number of antennas and subcarriers.

To overcome this limitation, transform-domain and compressive sensing approaches were introduced. Transform-based methods exploit the sparsity of the CSI in certain domains, such as the angular–delay or beamspace domain, by applying a unitary transform, e.g., discrete Fourier transform (DFT), to decorrelate spatial and frequency components before quantization and feedback [82]. Compressive sensing techniques, on the other hand, model the CSI as a sparse signal and acquire only a small number of linear measurements at

the UE, followed by sparse recovery at the BS [83], [84]. These methods can substantially reduce the feedback load while preserving most of the channel information, provided that the channel exhibits sufficient sparsity and that the sensing matrix satisfies the restricted isometry property.

Despite these advances, classical approaches generally rely on linear signal models, hand-crafted transforms, or idealized sparsity assumptions. Consequently, their performance degrades in practical non-stationary, frequency-selective, or rich-scattering environments, where these assumptions do not hold. These limitations have motivated the development of data-driven and DL-based CSI compression frameworks [85], which learn compact representations directly from channel data without explicit modeling assumptions.

In DL-based CSI compression, the encoder produces a latent representation $\mathbf{z} = f_\phi(\mathbf{H}) \in \mathbb{R}^{d_z}$ that is quantized and entropy-coded before transmission:

$$\mathbf{q} = \mathcal{Q}(\mathbf{z}), \qquad \mathbf{b} = \mathrm{EC}(\mathbf{q}), \qquad \widehat{\mathbf{H}} = g_\theta(\mathbf{q}), \tag{4.8}$$

where $\mathcal{Q}(\cdot)$ denotes a (scalar or vector) quantizer and $\mathrm{EC}(\cdot)$ a lossless entropy coder. The expected bit length is upper-bounded by the cross-entropy under a learned prior p_ψ, given by

$$\mathbb{E}[\ell(\mathbf{b})] \;\approx\; \mathbb{E}[-\log_2 p_\psi(\mathbf{q})]\,, \tag{4.9}$$

where p_ψ is jointly optimized with the encoder–decoder network to model the distribution of quantized latent features [86], [87].

Since the quantization operation is non-differentiable, it is replaced during training with a differentiable approximation, enabling end-to-end gradient-based optimization. Two common approaches are: (i) additive uniform noise, which approximates quantization as $\widetilde{\mathbf{q}} = \mathbf{z} + \mathbf{u}$, where $\mathbf{u} \sim \mathcal{U}(-0.5, 0.5)$, and (ii) the straight-through estimator (STE), which passes gradients through the discrete rounding operation as $\partial \mathcal{Q}(\mathbf{z})/\partial \mathbf{z} \approx 1$ [88], [89]. The overall optimization objective can thus be expressed as

$$\min_{\phi,\,\theta,\,\psi} \; \mathbb{E}\big[d\big(\mathbf{H},\, g_\theta(\widetilde{\mathbf{q}})\big)\big] + \lambda\, \mathbb{E}\big[-\log_2 p_\psi(\widetilde{\mathbf{q}})\big]\,, \tag{4.10}$$

where $\widetilde{\mathbf{q}}$ denotes the relaxed quantized latent. This formulation jointly optimizes the encoder, decoder, and entropy model to achieve an optimal RD trade-off, directly aligning with the principles of RD theory [90].

A critical challenge in DL–based CSI compression is generalization. NNs trained under a specific environment often exhibit significant performance degradation when deployed in unseen conditions. This sensitivity arises because the encoder–decoder pair implicitly learns the statistical structure from the training data. To maintain performance across varying environments, one solution is to retrain or fine-tune the neural model online. However, updating an NN model introduces a new form of overhead: the updated model parameters must also be transmitted to the decoder side. Considering the large number of parameters in state-of-the-art neural architectures, this process can easily exceed the original CSI feedback cost, negating the compression gains. Therefore, efficient adaptation strategies are essential to achieve a good RD trade-off.

4.3 CSI Prediction

In modern wireless communication systems, CSI prediction has emerged as a solution to cope with the rapidly time-varying nature of wireless channels, especially in high-mobility and high-frequency scenarios such as vehicular and mmWave communications. Since accurate and timely CSI is essential for beamforming, resource allocation, and link adaptation, even small feedback or estimation delays can severely degrade system performance. CSI prediction aims to forecast future channel states based on past observations, effectively extending the usable coherence time of the channel and reducing pilot overhead. By exploiting the temporal correlation inherent in the wireless channel, prediction methods enable the transmitter to proactively adapt its transmission strategy to the anticipated channel conditions, thereby improving robustness and spectral efficiency under mobility.

Consider a wideband MIMO–OFDM system with N_{t} transmit antennas, N_{r} receive antennas, and N_{c} subcarriers. Let the discrete-time, frequency-domain CSI at time index n be

$$\mathbf{H}_n \in \mathbb{C}^{N_{\mathrm{r}} \times N_{\mathrm{t}} \times N_{\mathrm{c}}}, \tag{4.11}$$

where $\mathbf{H}_n(:,:,k)$ denotes the MIMO channel matrix on subcarrier k at time t. We assume access to an observation window of N_{p} past channel realizations

$$\mathbf{H}_{\mathrm{p}} \triangleq \left\{\mathbf{H}_{n-N_{\mathrm{p}}+1}, \ldots, \mathbf{H}_n\right\}, \tag{4.12}$$

and aim to predict a future horizon of N_{f} time steps,

$$\mathbf{H}_{\mathrm{f}} \triangleq \left\{\mathbf{H}_{n+1}, \ldots, \mathbf{H}_{n+N_{\mathrm{f}}}\right\}. \tag{4.13}$$

Under the MSE criterion, the optimal predictor $f^{\star}(\cdot)$ minimizes the expected prediction error between the true future CSI and its estimate, i.e.,

$$f^{\star}(\mathbf{H}_{\mathrm{p}}) = \arg\min_{f} \ \mathbb{E}\Big[\big\|\mathbf{H}_{\mathrm{f}} - f(\mathbf{H}_{\mathrm{p}})\big\|^2\Big]. \tag{4.14}$$

The optimal solution to this problem, denoted by $f^{\star}(\mathbf{H}_{\mathrm{p}})$, is the MMSE estimator, which coincides with the CME of the future CSI given the past:

$$\begin{aligned} f^{\star}(\mathbf{H}_{\mathrm{p}}) &= \mathbb{E}[\,\mathbf{H}_{\mathrm{f}} \mid \mathbf{H}_{\mathrm{p}}\,] \\ &= \int \mathbf{H}_{\mathrm{f}}\, p(\mathbf{H}_{\mathrm{f}} \mid \mathbf{H}_{\mathrm{p}})\, d\mathbf{H}_{\mathrm{f}}. \end{aligned} \tag{4.15}$$

In practice, however, the underlying channel distribution and its dynamics are unknown and highly complex to model. Consequently, model-based approaches often fail to achieve satisfactory performance. In contrast, an NN trained on a sufficiently large dataset can approximate the optimal predictor $f^{\star}$ by learning a parametric mapping $f_{\boldsymbol{\theta}}$ that captures the underlying channel dependencies.

Depending on the prediction strategy, NN–based models for CSI prediction can generally be categorized into AR and sequence-to-sequence (seq2seq) frameworks. In the AR formulation, the network is trained to perform one-step-ahead prediction and then recursively uses its own outputs as inputs for subsequent steps. Specifically, given past CSI observations $\mathbf{H}_{\mathrm{p}}$, the network predicts the next channel state as

$$\widehat{\mathbf{H}}_{n+1} = f_{\boldsymbol{\theta}}(\mathbf{H}_{\mathrm{p}}), \tag{4.16}$$

and the prediction for future steps is obtained iteratively as

$$\widehat{\mathbf{H}}_{n+\tau} = f_{\boldsymbol{\theta}}\Big(\{\mathbf{H}_{\mathrm{p}}, \widehat{\mathbf{H}}_{n+1}, \ldots, \widehat{\mathbf{H}}_{n+\tau-1}\}\Big), \quad \tau = 2, \ldots, N_{\mathrm{f}}. \tag{4.17}$$

This approach is flexible and memory-efficient but prone to error accumulation, as prediction errors at each step are propagated to subsequent ones.

During training, techniques such as teacher forcing or scheduled sampling [91] are commonly employed to stabilize learning and mitigate exposure bias.

Alternatively, in the seq2seq approach, the model directly maps the entire sequence of past CSI to the sequence of future CSI in a single forward pass:

$$\widehat{\mathbf{H}}_{\mathrm{f}} = f_{\boldsymbol{\theta}}(\mathbf{H}_{\mathrm{p}}). \tag{4.18}$$

Seq2seq models can jointly learn temporal dependencies across all future steps, often providing better long-term consistency and stability. However, they are generally less flexible with respect to varying observation windows and prediction horizons.

Motivated by the significant success of DNNs in time-series forecasting problems—such as stock market prediction [92], electricity load forecasting [93], and traffic flow prediction [94]—the use of DNNs for CSI prediction has attracted growing research interest [95]–[105]. Various NN architectures have been GRUs, as well as Transformer-based models. RNN-based architectures have been successfully employed for CSI prediction in [98], [99], [102], while Transformer-based approaches have been investigated in [103], [104], showing superior performance in modeling long-range temporal dependencies.

RNNs were among the earliest neural architectures designed to handle sequential data, where each element of the sequence is processed step by step and encoded into a hidden state. Although RNNs can theoretically capture long-term dependencies, they suffer from the vanishing and exploding gradient problems, which make it difficult to learn long-range temporal correlations in practice [106]. To address these limitations, more advanced architectures such as LSTMs and GRUs were proposed. These models incorporate gating mechanisms—such as forget, input, and output gates in LSTMs, and update and reset gates in GRUs—to better control the flow of information and mitigate gradient-related instabilities. While LSTMs and GRUs improve performance for moderately long sequences, they still rely on sequential processing, which limits their scalability to long sequences and makes them prone to error propagation. In particular, their training and inference times grow linearly with the sequence length, which can become a bottleneck for large-scale or real-time applications.

To overcome these limitations, the Transformer architecture was introduced. Unlike RNN-based models, Transformers employ self-attention mechanisms to process the entire input sequence in parallel, enabling more effective modeling

of long-range dependencies while achieving significantly higher computational efficiency [16]. This parallelism allows for faster training and inference, establishing Transformers as the state-of-the-art architecture in various sequence modeling tasks. Owing to these advantages, Transformer-based approaches for CSI prediction have gained increasing attention, and recent studies [103], [104] have demonstrated substantial improvements in NMSE compared to RNN-based models. Despite these advances, Transformers still demand considerable memory and computational resources, and their inference latency can become a bottleneck in real-time applications. To mitigate this issue, recent work [104] proposes replacing the self-attention mechanism with a linear projection module, significantly reducing the computational complexity of Transformer-based CSI prediction.

RNNs and Transformers have demonstrated strong performance in time-series forecasting tasks, including CSI prediction. However, these architectures typically learn a deterministic mapping from historical CSI to future CSI, which limits their ability to capture the inherent randomness and multimodality of wireless channel evolution. In realistic propagation environments, multiple plausible future trajectories can emerge from the same past observations due to factors such as mobility, scattering dynamics, and small-scale fading. In contrast, generative models aim to learn the full underlying probability distribution of CSI evolution rather than relying solely on point estimates. Despite their potential, the use of generative modeling for CSI prediction has been relatively unexplored. In this context, Paper C addresses this gap by employing diffusion-based generative models to capture the stochastic nature of channel dynamics and produce more realistic and uncertainty-aware predictions.

CHAPTER 5

Summary of included papers

This chapter provides a summary of the included papers.

5.1 Paper A

Mehdi Sattari, Hao Guo, Deniz Gündüz, Ashkan Panahi, Tommy Svensson
Full-Duplex Millimeter Wave MIMO Channel Estimation: A Neural Network Approach
IEEE Transactions on Machine Learning in Communications and Networking,
vol. 2, pp. 1093-1108, 2024
 DOI: 10.1109/TMLCN.2024.3432865 .

This paper investigates CSI estimation in FD mmWave MIMO systems. Different pilot dimension schemes are considered where transmit antennas and users share pilot resources. The SI and user channels are estimated using a CNN, and a subsequent channel mapping based on FNNs is introduced to infer the user–receive channels from the user–transmit channels. Extensive

simulations are conducted to analyze the NN-based channel estimators. The results show that NN-based estimation significantly outperforms traditional LS and MMSE methods, particularly in high-noise and high-interference scenarios.
MS and TS proposed the research topic, MS conducted the literature review, formulated the problem, and developed the proposed solution. MS also performed all simulations and wrote the paper. HG, DG, AP, and TS contributed to revising and improving the manuscript.

5.2 Paper B

Mehdi Sattari, Deniz Gündüz, Tommy Svensson
Neural CSI Compression Fine-Tuning: Taming the Communication Cost of Model Updates
Submitted to *IEEE Transactions on Machine Learning in Communications and Networking*, Feb, 2026
.

In this paper, we address the generalization problem of neural CSI compression schemes. A key challenge arises from the need to transmit updated model parameters to the decoder. To mitigate this issue, we employ a spike-and-slab prior for model updates and incorporate the model update bit rate into the RD optimization. Simulation results show that the proposed scheme achieves improved RD performance while requiring minimal feedback overhead.
DG and TS proposed the research topic, MS conducted the literature review, formulated the problem, and developed the proposed solution. MS also performed all simulations and wrote the paper. DG and TS contributed to revising and improving the manuscript.

5.3 Paper C

Mehdi Sattari, Javad Aliakbari, Alexandre Graell i Amat, Tommy Svensson
CSI Prediction Using Diffusion Models
IEEE Transactions on Wireless Communications, major revision, Feb, 2026.

.

This paper presents a diffusion-based framework for CSI prediction. The problem is decomposed into two components: a temporal encoder and a diffusion-based generator. The temporal encoder extracts latent representations from past CSI observations, while the diffusion generator synthesizes future CSI samples. We consider both AR and seq2seq inference strategies, employing different neural architectures for the temporal encoder and various backbone models for the diffusion generator. Furthermore, we propose a simplified design in which the diffusion generator directly predicts future CSI from past observations without an explicit temporal encoder. The results demonstrate that the proposed diffusion-based CSI predictor significantly outperforms state-of-the-art baselines.
MS and JA proposed the research topic, formulated the problem, and developed the proposed solution. MS conducted the literature review and wrote the paper. MS and JA performed all simulations. AGA and TS contributed to revising and improving the manuscript.

CHAPTER 6

Concluding Remarks and Future Work

This thesis has explored deep learning–based approaches for CSI estimation, compression, and prediction. Although deep learning offers substantial performance gains across a wide range of problems, its practical deployment in real-world wireless systems remains challenging due to high computational complexity and long inference times. While latency may be tolerable in many applications, it is a critical constraint in wireless communications, and meeting this requirement is essential for integrating deep learning into future wireless technologies. Moreover, the generalization problem of deep learning models must be carefully examined, as wireless environments are inherently dynamic, and factors such as carrier frequency, propagation conditions, and network topology can lead to widely varying CSI statistics.

FD transmission remains relatively underexplored, and existing SI channel models are not yet mature enough to accurately characterize near-field SI propagation. Developing accurate SI channel models is essential for effective SI estimation and cancellation. Further research is also needed to understand the extent to which precise SI channel estimation can enhance digital SI cancellation. Alternatively, intelligent beamforming and combining strategies may help mitigate the need for explicit SI channel estimation, facilitating practical

FD operation. However, when digital SI cancellation remains necessary using an estimated SI channel, careful pilot design and transmission strategies will be crucial to minimize pilot overhead while maintaining estimation accuracy.

The CSI compression framework presented in Paper B can be further extended to the multi-user scenario, where inter-user correlation and shared channel characteristics can be exploited to improve compression efficiency. Moreover, incorporating generative approaches such as diffusion models may enable more effective modeling of CSI distributions, enhancing reconstruction quality and robustness under limited feedback or noisy conditions. Future research could also focus on task-oriented compression, where the encoder preserves only the information relevant for downstream tasks, thereby achieving more efficient utilization of feedback resources.

While this thesis has demonstrated the potential of deep learning and diffusion models for CSI prediction, a fundamental question is when and where CSI prediction should be preferred over CSI estimation. Future studies should quantify the trade-offs between prediction accuracy, estimation overhead, and system latency under different mobility, channel dynamics, and SNR conditions. Such analysis would clarify the operational regimes where prediction provides tangible benefits over conventional estimation or tracking methods.

References

[1] M. Shafi, A. F. Molisch, P. J. Smith, *et al.*, "5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[2] K. David and H. Berndt, "6G Vision and Requirements: Is There Any Need for Beyond 5G?" *IEEE Vehicular Technology Magazine*, vol. 13, no. 3, pp. 72–80, 2018.

[3] T. L. Marzetta, E. G. Larsson, H. Yang, and H. Q. Ngo, *Fundamentals of Massive MIMO*. Cambridge University Press, 2016.

[4] Ericsson, *Ericsson Mobility Report*, Available: `https://www.ericsson.com/en/reports-and-papers/mobility-report`, Nov. 2023.

[5] X. Shen, J. Gao, M. Li, *et al.*, "Toward immersive communications in 6G," *Frontiers in Computer Science*, vol. Volume 4 - 2022, 2023, ISSN: 2624-9898.

[6] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[7] J. G. Andrews, S. Buzzi, W. Choi, *et al.*, "What Will 5G Be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[8] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[9] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5G Evolution: A View on 5G Cellular Technology Beyond 3GPP Release 15," *IEEE Access*, vol. 7, pp. 127 639–127 651, 2019.

[10] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive MIMO Networks: Spectral, Energy, and Hardware Efficiency," *Foundations and Trends® in Signal Processing*, vol. 11, no. 3-4, pp. 154–655, 2017, ISSN: 1932-8346.

[11] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[12] M. Biguesh and A. Gershman, "Training-based MIMO channel estimation: a study of estimator tradeoffs and optimal training signals," *IEEE Transactions on Signal Processing*, vol. 54, no. 3, pp. 884–893, 2006.

[13] R. W. Heath and A. Lozano, *Foundations of MIMO Communication*. Cambridge University Press, 2018.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[15] G. Hinton, L. Deng, D. Yu, *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[16] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention Is All You Need," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5998–6008.

[17] A. Esteva, B. Kuprel, R. A. Novoa, *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks.," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[18] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[19] F. Khan, *LTE for 4G Mobile Broadband: Air Interface Technologies and Performance*. Cambridge University Press, 2009.

[20] F. Rusek, D. Persson, B. K. Lau, *et al.*, "Scaling Up MIMO: Opportunities and Challenges with Very Large Arrays," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 40–60, 2013.

[21] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for Next Generation Wireless Systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, 2014.

[22] T. S. Rappaport, S. Sun, R. Mayzus, *et al.*, "Millimeter Wave Mobile Communications for 5G Cellular: It Will Work!" *IEEE Access*, vol. 1, pp. 335–349, 2013.

[23] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[24] T. Mitchell, *Machine Learning* (McGraw-Hill International Editions). McGraw-Hill, 1997, ISBN: 9780071154673.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[27] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[29] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[30] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

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

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[37] H. Ye, G. Y. Li, and B.-H. Juang, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, 2018.

[38] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[39] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[40] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," in *Science*, vol. 313, 2006, pp. 504–507.

[41] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[42] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[43] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, *Linformer: Self-Attention with Linear Complexity*, 2020.

[44] K. M. Choromanski, V. Likhosherstov, D. Dohan, *et al.*, "Rethinking Attention with Performers," in *International Conference on Learning Representations*, 2021.

[45] Z. Liu, Y. Lin, Y. Cao, *et al.*, "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 10 012–10 022.

[46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[47] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 6840–6851.

[48] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Conditional Image Generation with PixelCNN Decoders," *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[49] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language Models are Unsupervised Multitask Learners*, OpenAI Technical Report, 2019.

[50] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[51] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[52] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, 22–24 Jun 2014, pp. 1278–1286.

[53] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[54] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[55] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," in *International Conference on Learning Representations (ICLR)*, 2019.

[56] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[57] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[58] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," in *International Conference on Learning Representations (ICLR)*, 2018.

[59] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," in *International Conference on Machine Learning (ICML)*, 2015.

[60] Y. Song, J. Sohl-Dickstein, D. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations," in *International Conference on Learning Representations (ICLR)*, 2021.

[61] J. Song, C. Meng, and S. Ermon, "Denoising Diffusion Implicit Models," in *arXiv preprint arXiv:2010.02502*, 2020.

[62] A. Mezghani and A. L. Swindlehurst, "Blind Estimation of Sparse Broadband Massive MIMO Channels With Ideal and One-bit ADCs," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2972–2983, 2018.

[63] Y. Chi, L. L. Scharf, A. Pezeshki, and A. R. Calderbank, "Sensitivity to Basis Mismatch in Compressed Sensing," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2182–2195, May 2011.

[64] M. Biguesh and A. B. Gershman, "Training-Based MIMO Channel Estimation: A Study of Estimator Tradeoffs and Optimal Training Signals," *IEEE Transactions on Signal Processing*, vol. 54, no. 3, pp. 884–893, Mar. 2006.

[65] R. W. H. Jr., N. Gonzalez-Prelcic, S. Rangan, W. Roh, and A. M. Sayeed, "An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 3, pp. 436–453, Apr. 2016.

[66] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive MIMO Networks: Spectral, Energy, and Hardware Efficiency," *Foundations and Trends in Signal Processing*, vol. 11, no. 3–4, pp. 154–655, 2017.

[67] B. Fesl, "Generative model-aided channel estimation design and optimality analysis," Ph.D. dissertation, Technische Universität München, 2025.

[68] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*. Englewood Cliffs, NJ: Prentice Hall, 1993, ISBN: 9780133457117.

[69] T. L. Marzetta, "Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas," *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3590–3600, Nov. 2010.

[70] H. Ye, G. Y. Li, and B.-H. Juang, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, Feb. 2018.

[71] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep Learning-Based Channel Estimation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 652–655, Apr. 2019.

[72] C.-J. Chun, J.-M. Kang, and I.-M. Kim, "Deep Learning-Based Channel Estimation for Massive MIMO Systems," *IEEE Wireless Communications Letters*, vol. 8, no. 4, pp. 1228–1231, 2019.

[73] D. Neumann, T. Wiese, and W. Utschick, "Learning the MMSE Channel Estimator," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2905–2917, 2018.

[74] Y. Dong, H. Wang, and Y.-D. Yao, "Channel Estimation for One-Bit Multiuser Massive MIMO Using Conditional GAN," *IEEE Communications Letters*, vol. 25, no. 3, pp. 854–858, 2021.

[75] B. Fesl, M. Baur, F. Strasser, M. Joham, and W. Utschick, "Diffusion-Based Generative Prior for Low-Complexity MIMO Channel Estimation," *IEEE Wireless Communications Letters*, vol. 13, no. 12, pp. 3493–3497, 2024.

[76] M. Arvinte and J. I. Tamir, "MIMO Channel Estimation Using Score-Based Generative Models," *IEEE Transactions on Wireless Communications*, vol. 22, no. 6, pp. 3698–3713, 2023.

[77] X. Zhou, L. Liang, J. Zhang, P. Jiang, Y. Li, and S. Jin, "Generative Diffusion Models for High Dimensional Channel Estimation," *IEEE Transactions on Wireless Communications*, vol. 24, no. 7, pp. 5840–5854, 2025.

[78] "Study on 3D channel model for LTE (Release 12)," 3rd Generation Partnership Project (3GPP), Tech. Rep. TR 36.873, V12.7.0, Dec. 2018.

[79] "NR; Physical layer procedures for data (Release 18)," 3rd Generation Partnership Project (3GPP), Tech. Rep. TS 38.214, V18.3.0, Jun. 2024.

[80] D. J. Love, R. W. Heath, V. K. N. Lau, D. Gesbert, B. D. Rao, and M. Andrews, "An Overview of Limited Feedback in Wireless Communication Systems," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 8, pp. 1341–1365, Oct. 2008.

[81] N. Jindal, "MIMO Broadcast Channels With Finite-Rate Feedback," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 5045–5060, Nov. 2006.

[82] X. Rao and V. K. N. Lau, "Distributed Compressive CSIT Estimation and Feedback for FDD Multiuser Massive MIMO Systems," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3261–3271, Jun. 2014.

[83] A. Alkhateeb, O. E. Ayach, G. Leus, and R. W. Heath, "Channel Estimation and Hybrid Precoding for Millimeter Wave Cellular Systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 831–846, Oct. 2014.

[84] J. Lee, G. Gil, and Y. Lee, "Channel Estimation via Orthogonal Matching Pursuit for Hybrid MIMO Systems in Millimeter Wave Communications," *IEEE Transactions on Communications*, vol. 64, no. 6, pp. 2370–2386, Jun. 2016.

[85] C. Wen, W. Shih, S.-C. Lin, S.-Y. Chen, S.-C. Kuo, and K.-T. Feng, "Deep Learning for CSI Feedback in Massive MIMO Systems," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, Oct. 2018.

[86] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-End Optimized Image Compression," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[87] D. Minnen, J. Ballé, and G. Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[88] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy Image Compression with Compressive Autoencoders," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[89] Y. Bengio, N. Léonard, and A. Courville, *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*, 2013.

[90] M. B. Mashhadi, Q. Yang, and D. Gündüz, "Distributed deep convolutional compression for massive mimo csi feedback," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2621–2633, 2021.

[91] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[92] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018, ISSN: 0377-2217.

[93] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.

[94] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2018, pp. 3634–3640.

[95] C. Wu, X. Yi, Y. Zhu, W. Wang, L. You, and X. Gao, "Channel Prediction in High-Mobility Massive MIMO: From Spatio-Temporal Autoregression to Deep Learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 1915–1930, 2021.

[96] O. Stenhammar, G. Fodor, and C. Fischione, "A Comparison of Neural Networks for Wireless Channel Prediction," *IEEE Wireless Communications*, vol. 31, no. 3, pp. 235–241, 2024.

[97] G. Liu, Z. Hu, L. Wang, J. Xue, H. Yin, and D. Gesbert, "Spatio-Temporal Neural Network for Channel Prediction in Massive MIMO-OFDM Systems," *IEEE Transactions on Communications*, vol. 70, no. 12, pp. 8003–8016, 2022.

[98] W. Jiang and H. D. Schotten, "Recurrent Neural Network-Based Frequency-Domain Channel Prediction for Wideband Communications," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–6.

[99] W. Jiang and H. D. Schotten, "Deep Learning for Fading Channel Prediction," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 320–332, 2020.

[100] S. Fan, Z. Liu, X. Gu, and H. Li, "Csi-LLM: A Novel Downlink Channel Prediction Method Aligned with LLM Pre-Training," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 1–6.

[101] H. Kim, J. Choi, and D. J. Love, *Machine Learning for Future Wireless Communications: Channel Prediction Perspectives*, 2025.

[102] W. Xu, J. An, Y. Xu, C. Huang, L. Gan, and C. Yuen, "Time-Varying Channel Prediction for RIS-Assisted MU-MISO Networks via Deep Learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1802–1815, 2022.

[103] H. Jiang, M. Cui, D. W. K. Ng, and L. Dai, "Accurate Channel Prediction Based on Transformer: Making Mobility Negligible," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 9, pp. 2717–2732, 2022.

[104] Y. Jin, Y. Wu, Y. Gao, S. Zhang, S. Xu, and C.-X. Wang, "LinFormer: A Linear-based Lightweight Transformer Architecture For Time-Aware MIMO Channel Prediction," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2025.

[105] C. Jiang, J. Guo, C.-K. Wen, and S. Jin, "Enhancing Reliability in AI-Based CSI Prediction: A Proxy-Based Performance Monitoring Approach," *IEEE Transactions on Communications*, vol. 73, no. 4, pp. 2602–2615, 2025.

[106] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester, Eds., ser. Proceedings of Machine Learning Research, vol. 28, Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1310–1318.

Part II

Papers

PAPER A

Full-Duplex Millimeter Wave MIMO Channel Estimation: A Neural Network Approach

Mehdi Sattari, Hao Guo, Deniz Gündüz, Ashkan Panahi, Tommy Svensson

IEEE Transactions on Machine Learning in Communications and Networking,
vol. 2, pp. 1093-1108, 2024
 DOI: 10.1109/TMLCN.2024.3432865

The layout has been revised.

Abstract

Millimeter wave (mmWave) multiple-input-multi-output (MIMO) is now a reality with great potential for further improvement. We study full-duplex transmissions as an effective way to improve mmWave MIMO systems. Compared to half-duplex systems, full-duplex transmissions may offer higher data rates and lower latency. However, full-duplex transmission is hindered by self-interference (SI) at the receive antennas, and SI channel estimation becomes a crucial step to make the full-duplex systems feasible. In this paper, we address the problem of channel estimation in full-duplex mmWave MIMO systems using neural networks (NNs). Our approach involves sharing pilot resources between user equipments (UEs) and transmit antennas at the base station (BS), aiming to reduce the pilot overhead in full-duplex systems and to achieve a comparable level to that of a half-duplex system. Additionally, in the case of separate antenna configurations in a full-duplex BS, providing channel estimates of transmit antenna (TX) arrays to the downlink UEs poses another challenge, as the TX arrays are not capable of receiving pilot signals. To address this, we employ an NN to map the channel from the downlink UEs to the receive antenna (RX) arrays to the channel from the TX arrays to the downlink UEs. We further elaborate on how NNs perform the estimation with different architectures, (e.g., different numbers of hidden layers), the introduction of non-linear distortion (e.g., with a 1-bit analog-to-digital converter (ADC)), and different channel conditions (e.g., low-correlated and high-correlated channels). Our work provides novel insights into NN-based channel estimators.

A.1 Introduction

Current key enabling wireless technologies, such as massive multiple-input-multi-output (MIMO), millimeter-wave (mmWave) communication, and ultra-

dense networks have significantly improved the throughput of wireless communications [1]. Thanks to the huge bandwidth available in the mmWave band, we can meet the high data rate requirements for future cellular networks, and the high path loss at these frequencies can be compensated by utilizing the beamforming gain of MIMO antennas. Nonetheless, these technologies have been mainly studied for half-duplex communication, and the potential of utilizing full-duplex communication has been widely overlooked. In full-duplex transmission, a transceiver can simultaneously transmit and receive over the same carrier frequency, in principle doubling the spectral efficiency. Besides, the delay associated with half-duplex transmission is not present in full-duplex systems, thereby enabling the low latency requirements of 5G and beyond [2], [3].

To realize full-duplex transmission, self-interference (SI) caused by each transmit element over the receive array has to be canceled out. Consecutive cancellation procedures are usually considered in three domains: propagation, analog, and digital [4]. In propagation domain cancellation, SI power is suppressed before reaching the receive chain circuit. Analog domain cancellation is performed before the analog-to-digital converter (ADC) in the receiver analog chain. Finally, digital domain cancellation suppresses the SI power remaining from the propagation and analog domains [5]. Implementing full-duplex in mmWave systems poses unique challenges distinct from lower-frequency counterparts. Unlike lower-frequency systems, mmWave transceivers utilize dense antenna arrays and wide bandwidths, necessitating tailored solutions for handling the SI and exploiting spatial degrees of freedom. While lower-frequency full-duplex solutions often rely on analog SI cancellation, mmWave systems require novel approaches due to their unique transceiver architectures and propagation characteristics.

There are two main antenna configurations used for full-duplex systems: separate and shared. In the separate antenna configuration, separate antennas are dedicated to transmission and reception, yielding high isolation and relatively low interference. In contrast, in the latter configuration, a single antenna is utilized for both transmission and reception, facilitated by the use of circulators to separate the incoming and outgoing signals. This setup offers a simpler overall design and reduced hardware costs, albeit with potential challenges in managing interference and maintaining signal integrity.

Due to the large number of antenna elements in massive MIMO systems,

channel estimation is a complex process with a high pilot overhead. This is the case even in half-duplex systems, but is more pronounced in full-duplex systems due to the critical need for SI cancellation, which requires estimation of the large SI channel. Several studies have been conducted to address channel estimation in full-duplex systems. In [6], a channel estimation scheme based on least squares (LS) estimation is proposed for bidirectional communication between a pair of transceivers. The achievable sum rate was analyzed considering the limited dynamic range of the transmitter and receiver and channel estimation error. In [7], an LS-based scheme was utilized for a multi-pair two-way amplify and forward (AF) relaying system. Additionally, a low-pilot overhead scheme was proposed, where two user equipments (UEs) employ the same pilot sequence, while different UE pairs adopt orthogonal pilot sequences.

In [8], an expectation-maximization algorithm is utilized to address the channel estimation problem in a large-scale MIMO system with a full-duplex relay. The authors explore two different scenarios: in the first scenario, both the base station (BS) and the relay estimate their respective individual channels, while in the second scenario, only the BS estimates the cascaded channel between the transmitter, relay, and receiver. In [9], channel estimation is performed in both the radio frequency (RF) and baseband domains. In the RF domain, the authors employ the LS technique to estimate the SI channel. Subsequently, a subspace-based algorithm is employed in the baseband to estimate the residual SI and BS-UE channels. The study in [10] proposes a specific frame structure for full-duplex systems, for which the achievable rate and SI channel estimation are analyzed. Furthermore, the impact of the calibration period on SI channel estimation is investigated in [11].

In [12], the authors have studied the problem of hybrid beamforming with low-resolution phase shifters for full-duplex mmWave large-scale MIMO systems and optimized the sum spectral efficiency (SE) of uplink and downlink UEs, assuming perfect channel state information (CSI) knowledge. Two different SI cancellation approaches, namely spatial suppression and SI subtraction, have been compared in [13], taking into account channel estimation errors and channel spatial correlation. The uplink and downlink achievable rates have been obtained in [14], considering both the perfect CSI and imperfect CSI cases in full-duplex massive MIMO systems with low-resolution ADCs. A study has been conducted in [15] for full-duplex large-scale MIMO cellular systems, considering both non-cooperative and cooperative scenarios. Minimum

mean squared error (MMSE) channel estimation is employed for SI and UE channels. Analytical expressions have been derived for the ergodic achievable rate using linear filters such as matched filter (MF) and zero-forcing (ZF). Authors in [16] have introduced a hybrid beamforming design for mmWave full-duplex, addressing several practical aspects such as codebook-based analog beamforming and beam alignment.

A neural network (NN) is a specialized tool within the field of machine learning that has demonstrated remarkable effectiveness in solving a wide range of tasks, including computer vision, speech recognition, and autonomous driving [17]. Thanks to the versatility of NNs, they have also found applications in wireless communication, either in a block-structured manner or in an end-to-end scenario [18]. Research in the wireless communication community has explored optimizing traditional processing blocks, such as channel estimation [19], precoding [20], signal detection [21], and CSI feedback [22], using deep learning techniques. Moreover, in [23], an end-to-end communication system has been represented by an autoencoder. In [24], a communication system model is developed without prior information on the channel, with the assistance of conditional generative adversarial networks (GANs).

Numerous studies have focused on NN-based channel estimation. In [25], convolutional neural networks (CNNs) are employed to estimate the mmWave massive MIMO channel, demonstrating that CNNs can achieve superior performance compared to MMSE by effectively exploiting spatial and frequency correlation. In [26], the authors utilize a specifically designed CNN called "deep image prior" to denoise the LS estimated massive MIMO channel. Their findings indicate that deep learning can successfully estimate pilot-contaminated massive MIMO channels. In another work, [27], fully connected neural networks (FNNs) are utilized to estimate the massive MIMO channel from quantized received measurements with low-resolution ADCs. The study reveals that with sufficiently large antenna arrays, fewer pilots are required for channel estimation. For beamspace mmWave massive MIMO systems, a learned denoising-based approximate message passing network is used to solve the channel estimation problem [28]. Simulation results demonstrate that deep learning-based channel estimation outperforms compressed sensing (CS)-based schemes in this context.

As highlighted in several papers, such as [2], [3], and [12], the channel estimation problem in full-duplex transmission has not received extensive re-

search attention in the literature. The few existing works mentioned above are primarily focused on simple channel estimation techniques, like LS estimators, which may not ensure high-quality channel estimates in low signal-to-noise ratios (SNRs). On the other hand, the MMSE channel estimator requires knowledge of the channel correlation matrix and is burdened with high computational complexity. More importantly, classical channel estimation techniques require long pilot dimensions in order to perform well. But as we pointed out, in full-duplex systems, the SI channel is a large MIMO channel, and estimating this channel together with UE channels is extremely costly, rendering the realization of full-duplex systems impractical. Furthermore, interference or residual error from SI and UE signals can heavily disrupt the received pilot signal in full-duplex systems. Another challenge is acquiring the channel from transmit antenna (TX) arrays to the downlink UEs in separate antenna configurations.

In order to circumvent the aforementioned critical problems in channel estimation for full-duplex systems, we examine NNs to estimate both SI and UE channels. The main motivation for adopting NNs for this problem can be summarized as follows: 1) It has been shown in various studies that deep learning can provide remarkably good performance in channel estimation with a short pilot dimension, see e.g., [27], 2) Deep learning has demonstrated impressive performance in noisy conditions [26], holding promise for addressing the problem of heavy interference in the received pilot signal of full-duplex systems. 3) Finally, NNs have been successfully utilized to approximate the mapping of channels in frequency and spatial domains [29]. Therefore, NNs are a good candidate for obtaining the channel from TX arrays to downlink UEs by approximating the spatial mapping of the channel from downlink UEs to receive antenna (RX) arrays, to the channel of TX arrays and to downlink UEs.

The role of the number of hidden layers in NN-based channel estimators is not well understood. The general belief in the advantages of depth stems from experiments in machine learning on data modalities such as images or text, which can be far more complex than wireless channels. The latency constraints in wireless systems are also radically different from those in computer vision tasks. As such, the same advantages of depth may not hold for channel estimation. Additionally, the computational complexity and generalization capabilities of deep NNs are significant bottlenecks in implementing

NN estimators in practice. Hence, we concentrate on simple NNs ranging from zero to a few hidden layers. We aim to investigate how these simple NNs perform across various channel conditions and system scenarios compared to deep NNs. Additionally, to approximate the channel mapping from TX arrays to downlink UEs to the channel of RX arrays to downlink UEs, we employ FNNs with different numbers of hidden layers.

The summary of the contributions made in this paper is as follows:

- First, we examine the performance of the NN-based channel estimator with varying numbers of hidden layers and compare them to traditional LS and MMSE techniques, as well as state-of-the-art deep NNs. We consider architectures with 0, 1, 2, and 10 convolutional hidden layers, with the latter corresponding to the deep NN architecture described in [25]. To further explore NN-based estimators, we introduce 1-bit ADCs at the full-duplex BS to introduce nonlinear distortion to the received pilot signal. Additionally, to delve deeper into the problem of channel estimation with low-resolution ADCs, we consider cases with 2-bit and 3-bit ADCs. Furthermore, we investigate the performance of NNs under different channel spatial correlation regimes, namely, low-correlation and high-correlation regimes.

- In order to reduce the pilot overhead imposed by full-duplex transmission, we share pilot resources between TX arrays of BS and UEs and explore different pilot dimensions. First, we estimate the SI channel and then cancel out the SI signal from the received pilot signal to estimate the UE channels. We discuss how the SI power influences the channel estimates of UEs when pilot resources are shared between TX arrays and UEs. We further analyze the effect of SI cancellation during the pilot phase for UE channel estimation.

- To acquire channel estimates from transmit arrays to downlink UEs in separate antenna configurations, we seek a mapping from the channel of downlink UEs to the RX arrays to the channel of TX arrays to the downlink UEs. Due to the multi-path effect and random scattering environment, there is no simple mathematical formulation for this mapping. Instead, we utilize an NN to approximate it based on the universal approximation [30] theory. Accordingly, we estimate the channels of downlink UEs at the RX arrays as well and then map them to the

channel from TX arrays to the downlink UEs using NNs. Simulation results show that the NN can learn this mapping successfully.

- We analyze how our trained models perform when there is a significant distribution shift between the training and test phases. We utilize the DeepMIMO dataset [31] to generate channel samples that differ from the data used for training. As expected, the performance of the trained model degrades due to the distribution shift. Moreover, we observe that simpler NNs with fewer hidden layers exhibit better generalization ability compared to deeper NNs, since the number of trainable variables is much lower in shallow NNs. This underscores the advantages of utilizing simpler NN architectures for channel estimation problems.

- Finally, we compare the computational complexity of LS, MMSE, and NN channel estimators in terms of floating point operations (FLOPs). This comparison indicates that the number of FLOPs increases quadratically with the number of received antennas in MMSE estimation, whereas in LS and NN-based estimation, it grows linearly. Furthermore, the comparison of the computational complexity of NNs with different numbers of hidden layers indicates that NNs with a few hidden layers can significantly save computational resources while employing a deep NN poses a major challenge in practice.

The rest of this paper is organized as follows. Section A.2 introduces the channel models for SI and UEs, as well as pilot transmission schemes. We examine different channel estimation approaches such as LS and MMSE, and introduce the NN-based estimator for SI and UE channels in Section A.3, and the RX-TX mapping for separate antenna configurations. In Section A.4, we provide extensive numerical simulations and compare different channel estimators in terms of normalized mean squared error (NMSE) and computational complexity. Section A.5 concludes the paper.

Throughout the paper the following notations are used: Matrices are represented using bold uppercase letters, while bold lowercase letters denote vectors. Transpose, transpose conjugate (Hermitian), and matrix inversion operations are denoted by the superscripts $(\cdot)^T$, $(\cdot)^H$, and $(\cdot)^{-1}$, respectively. We use $\mathbb{E}$ to denote expectation, $||\cdot||_F$ to represent the Frobenius norm, and $\otimes$ to denote Kronecker product. The identity matrix of size $n \times n$ is represented by $\mathbf{I}_n$. For a generic matrix $\mathbf{A} \in \mathbb{C}^{n\times m}$, $[\mathbf{A}]_{ij}$ refers to the element in the

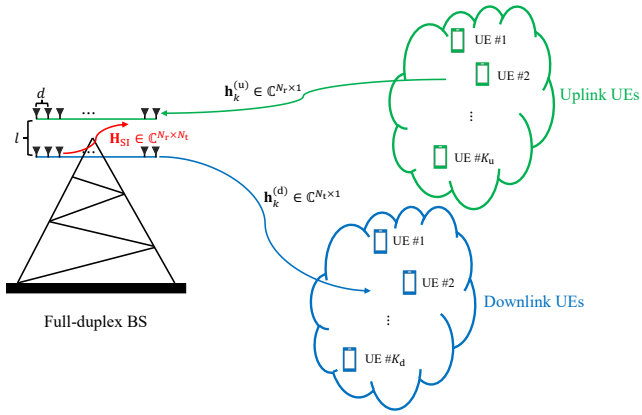


Figure 1: System model with a full-duplex BS serving K_u single-antenna UEs in the uplink and K_d single-antenna UEs in the downlink channel.

(i, j)-th position. The operation $vec(\mathbf{A})$ denotes a column vector obtained by stacking the columns of $\mathbf{A}$ one below the other.

A.2 System Model

A.2.1 Channel Modeling

We consider a single-cell system with one full-duplex mmWave MIMO BS, serving K_u uplink UEs and K_d downlink UEs as depicted in the Fig. 1. Both uplink and downlink UEs are half-duplex devices and the total number of UEs in the cell is $K = K_\mathrm{u} + K_\mathrm{d}$. We consider separate antenna configurations, where the BS is equipped with two uniform linear arrays to enable full-duplex transmission. The number of transmit and receive antennas are N_t and N_r, respectively, and they are linearly placed with a uniform distance.

In order to fully investigate different estimators within various scenarios, in this work, the UE channels follow a geometrical channel model as follows [32]:

$$\mathbf{h}_k^{(\mathrm{u})} = \sqrt{\frac{N_\mathrm{r}}{P}} \sqrt{\beta_{\mathrm{u},k}} \sum_{i=0}^{P-1} \alpha_{i,k} \mathbf{a}_\mathrm{a}(\theta_{i,k}), \quad k = 1, 2, \ldots, K_\mathrm{u}, \tag{A.1}$$

$$\mathbf{h}_k^{(\mathrm{d})} = \sqrt{\frac{N_\mathrm{t}}{P}}\sqrt{\beta_{\mathrm{d},k}}\sum_{i=0}^{P-1}\alpha_{i,k}\mathbf{a}_\mathrm{d}(\theta_{i,k}), \quad k = 1, 2, \ldots, K_\mathrm{d}, \tag{A.2}$$

where $\mathbf{h}_k^{(\mathrm{u})}(\mathbf{h}_k^{(\mathrm{d})})$ is the uplink (downlink) channel between the k-th uplink (downlink) UE and receive (transmit) antenna arrays at BS. Furthermore, P represents the number of multi-path components, $\alpha_{i,k} \sim \mathcal{CN}(0,1)$ are the small-scale fading factors of individual paths, and $\beta_{\mathrm{u(d)},k}$ stands for the large-scale fading

$$\beta_{\mathrm{u(d)},k} = \Gamma - 10\eta\log(r_{\mathrm{u(d)},k}) + \chi_k, \tag{A.3}$$

where Γ denotes the average channel gain in dB at a reference distance of 1 m and specific carrier frequency, $r_{\mathrm{u(d)},k}$ is the uplink (downlink) distance between transmitter and receiver, χ_k is the shadow fading with a lognormal distribution $\mathcal{LN}(0, \sigma_\mathrm{sf})$, where σ_sf is standard deviation, and η is the path loss exponent. Furthermore, $\mathbf{a}_\mathrm{a}(\theta)$ and $\mathbf{a}_\mathrm{d}(\theta)$ are, respectively, the uniform linear array (ULA) responses of the receive and transmit antennas, given by

$$\mathbf{a}_\mathrm{a}(\theta) = \sqrt{\frac{1}{N_\mathrm{r}}}\left[1, \exp\left(j2\pi\frac{d}{\lambda}(\sin\theta)\right), \ldots, \exp\left(j2\pi\frac{d}{\lambda}(N_\mathrm{r}-1)\sin\theta\right)\right]^T, \tag{A.4}$$

and

$$\mathbf{a}_\mathrm{d}(\theta) = \sqrt{\frac{1}{N_\mathrm{t}}}\left[1, \exp\left(j2\pi\frac{d}{\lambda}(\sin\theta)\right), \ldots, \exp\left(j2\pi\frac{d}{\lambda}(N_\mathrm{t}-1)\sin\theta\right)\right]^T, \tag{A.5}$$

where θ is the angle of arrival/departure (AoA/AoD) to/from the receive/transmit antenna arrays at the BS, and d and λ are the antenna spacing and wavelength, respectively. We assume that the AoA and AoD follow the local scattering model with a uniform distribution $\left[-\frac{\theta_\mathrm{AS}}{2}, \frac{\theta_\mathrm{AS}}{2}\right]$ [33], where θ_AS is the angular spread (AS) of the multi-path components.

In the full-duplex transmission, there is cross-link interference between uplink and downlink UEs that needs to be carefully modeled. Nonetheless, since the focus of the paper is channel estimation using uplink pilots, we are neglecting the cross-link interference in our study.

We denote the SI channel from the transmit antenna arrays to the receive antenna arrays as $\mathbf{H}_\mathrm{SI}$, which is modeled as a Rician fading channel due to the

presence of a strong line-of-sight (LoS) path. Hence, the SI channel consists of two components: the near-field and the far-field terms [2]:

$$\mathbf{H}_{\mathrm{SI}} = \sqrt{\epsilon_{\mathrm{SI}}}\sqrt{\frac{\kappa}{\kappa+1}}\mathbf{H}_{\mathrm{SI,NF}} + \sqrt{\frac{1}{\kappa+1}}\mathbf{H}_{\mathrm{SI,FF}}. \tag{A.6}$$

We assume that SI is sufficiently suppressed in the propagation domain and the received signal at the baseband does not exceed the dynamic range of the ADCs. We refer to the propagation domain SI suppression factor as ϵ_{SI}. Note that in the propagation domain, only the near field channel from the transmit antenna arrays to the receive antenna arrays can be suppressed. In the digital domain, the residual near-field and far-field SI channels can be further suppressed after estimating the SI channel. The near-field term $\mathbf{H}_{\mathrm{SI,NF}}$ arises from the close-in placement of the transmit and receive antennas, while the far-field term $\mathbf{H}_{\mathrm{SI,FF}}$ is a result of the reflections in the environment. Here, κ is the Rician factor and $\mathbf{H}_{\mathrm{SI,FF}}$ follows a similar geometrical channel modeling as the UE channels [32]:

$$\mathbf{H}_{\mathrm{SI,FF}} = \sqrt{\frac{N_{\mathrm{t}}N_{\mathrm{r}}}{P}}\sqrt{\beta_{\mathrm{SI}}}\sum_{i=0}^{P-1}\alpha_i\mathbf{a}_{\mathrm{a}}(\theta_i)\mathbf{a}_{\mathrm{d}}{}^{T}(\theta_i), \tag{A.7}$$

and β_{SI} follows a similar formulation to $\beta_{\mathrm{u(d)},k}$.

The near-field term does not satisfy the far-field condition and therefore requires a completely different channel modeling approach [34], [35]. In this paper, we adopt the model in [36], given by

$$[\mathbf{H}_{\mathrm{SI,NF}}]_{nm} = \frac{\rho}{r_{nm}}\exp\left(-2\pi j\frac{r_{nm}}{\lambda}\right), \tag{A.8}$$

where r_{nm} is the distance between the n-th receive antenna and the m-th transmit antenna, and ρ is a constant for power normalization.

A.2.2 Pilot transmission schemes

We consider various pilot dimensions for channel estimation in full-duplex MIMO systems. Since uplink and downlink transmissions happen at the same time and frequency in full-duplex systems, only uplink pilots will be transmitted from UEs. To estimate the channels of UEs and SI, both downlink and uplink UEs, along with transmit antennas at the BS, transmit pilot signals

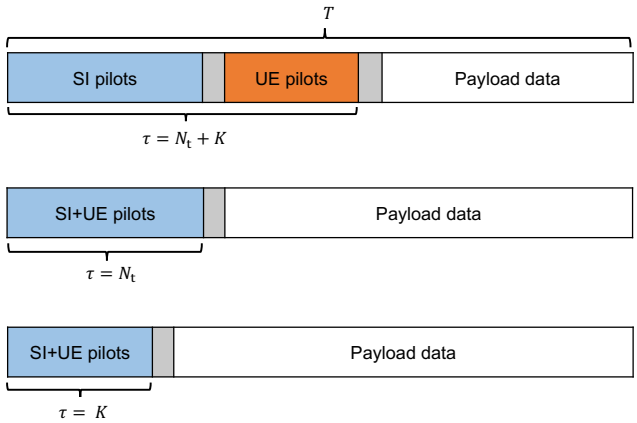


Figure 2: Different pilot dimensions. T is the frame dimension during one coherence block and τ is the pilot dimension for estimating SI and UE channels.

to the receive antennas at the BS. Specifically, we assume that all the K UEs transmit their pilot signals to the RX arrays in the channel estimation phase, whereas in the payload data transmission phase, a subset of K_{u} UEs will be in the uplink mode and K_{d} UEs will be in the downlink mode.

In the baseline scenario, we assume UEs and transmit antennas of the BS send their pilot signals at different pilot resources. Due to large antenna arrays, with this pilot transmission, the pilot overhead will be extremely high. In the second scenario, we assume the available pilot resources will be equal to the number of transmit antennas at the BS, and UEs will reuse the same pilot resources utilized by the BS to estimate BS-UE channels. Compared to the first scenario, the pilot overhead will be lower, but the received pilot signals will be contaminated by UEs/TX arrays when estimating SI/UE channels. To further reduce the pilot overhead, we reduce the number of pilots to the number of UEs. In this case, besides interference, the SI channel must be estimated with fewer pilots than the channel dimension, but with some performance loss, we can achieve a comparable pilot overhead as half-duplex systems. Three different pilot dimensions considered for estimating SI and UE channels are depicted in Fig. 2.

We assume that the received SI signal power remains stronger than the received UE signal power, even after SI cancellation in the propagation domain.

Hence, our approach involves initially estimating the SI channel and subsequently subtracting the estimated SI signal from the received pilot signal to estimate the UE channels.

A.2.3 SI channel estimation

For SI channel estimation, the received pilot signal at the RX arrays can be written as

$$\mathbf{Y}_{\mathrm{SI}} = \sqrt{\mathrm{SNR}_{\mathrm{SI}}}\mathbf{H}_{\mathrm{SI}}\mathbf{F}\mathbf{X}_{\mathrm{SI}} + \sqrt{\mathrm{SNR}_{\mathrm{UE}}}\mathbf{H}_{\mathrm{UE}}\mathbf{X}_{\mathrm{UE}} + \mathbf{N}, \tag{A.9}$$

where $\mathbf{Y}_{\mathrm{SI}} \in \mathbb{C}^{N_{\mathrm{r}} \times \tau}$ is the received pilot signal at receive antennas of BS during τ pilot transmissions. $\mathbf{X}_{\mathrm{UE}} \in \mathbb{C}^{K \times \tau}$ is the transmitted pilot signal from the K UEs, $\mathbf{X}_{\mathrm{SI}} \in \mathbb{C}^{\tau \times \tau}$ is a diagonal matrix whose diagonal elements are transmitted pilot signals from the transmit antennas at BS, $\mathbf{F} \in \mathbb{C}^{N_{\mathrm{t}} \times \tau}$ is the precoding matrix of transmit antennas of BS, and $\mathbf{H}_{\mathrm{UE}} = \left[\mathbf{h}_1^{(\mathrm{u})}, \mathbf{h}_2^{(\mathrm{u})}, \ldots, \mathbf{h}_{K_{\mathrm{u}}}^{(\mathrm{u})}, \mathbf{h}_1^{(\mathrm{d})}, \mathbf{h}_2^{(\mathrm{d})}, \ldots, \mathbf{h}_{K_{\mathrm{d}}}^{(\mathrm{d})}\right] \in \mathbb{C}^{N_{\mathrm{r}} \times K}$ is the concatenated uplink and downlink channel response matrix, where the k-th column corresponds to the channel between the k-th uplink or downlink UE and the receive antenna at the BS. In this paper, we assume fully digital beamforming at the full-duplex BS, and considering analog or hybrid precoding is left for future work. We assume all UEs have the same transmit power p_{UE}, and p_{SI} denotes the transmit power from transmit antenna arrays. We define $\mathrm{SNR}_{\mathrm{SI}}$ and $\mathrm{SNR}_{\mathrm{UE}}$ as

$$\mathrm{SNR}_{\mathrm{SI}} = \frac{p_{\mathrm{SI}}|\mathbf{H}_{\mathrm{SI}}|^2}{\sigma_{\mathrm{n}}^2}, \tag{A.10}$$

$$\mathrm{SNR}_{\mathrm{UE}} = \frac{p_{\mathrm{UE}}|\mathbf{H}_{\mathrm{UE}}|^2}{\sigma_{\mathrm{n}}^2}, \tag{A.11}$$

where σ_{n}^2 is the noise variance. The matrix $\mathbf{N} \in \mathbb{C}^{N_{\mathrm{r}} \times \tau}$ is the additive noise comprised of independent and identically distributed (i.i.d.) elements following a zero-mean unit variance Gaussian distribution.

Note that when the pilot dimension is $\tau = N_{\mathrm{t}} + K$, the SI and UE channel estimation happens at orthogonal pilot resources. Hence, the received pilot signal for SI channel estimation does not interfere with the pilots transmitted by UEs.

During pilot transmission, we apply an orthogonal pilot codebook for the

precoded SI signals from the transmit antenna arrays and pilot signals from UEs. Specifically, we utilize the discrete Fourier transform (DFT) pilot codebook, given by $\mathbf{F}\mathbf{X}_{\mathrm{SI}} = \mathbf{W}_{N_{\mathrm{t}}\times\tau}, \mathbf{X}_{\mathrm{UE}} = \mathbf{W}_{K\times\tau}$, where,

$$\mathbf{W}_{M\times\tau} = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{\tau-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(\tau-1)} \\ 1 & \omega^3 & \omega^6 & \cdots & \omega^{3(\tau-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{M-1} & \omega^{2(M-1)} & \cdots & \omega^{(M-1)(\tau-1)} \end{bmatrix} \tag{A.12}$$

and $\omega = e^{-2\pi j/\tau}$.

A.2.4 UE channel estimation

To estimate the UE channels, we use the estimated SI channel and cancel out the SI signal power from the received pilot signal. Therefore, the received pilot signal for UE channel estimation becomes

$$\mathbf{Y}_{\mathrm{UE}} = \sqrt{\mathrm{SNR}_{\mathrm{UE}}}\mathbf{H}_{\mathrm{UE}}\mathbf{X}_{\mathrm{UE}} + \sqrt{\mathrm{SNR}_{\mathrm{SI}}}\mathbf{E}_{\mathrm{SI}}\mathbf{F}\mathbf{X}_{\mathrm{SI}} + \mathbf{N}, \tag{A.13}$$

where $\mathbf{E}_{\mathrm{SI}} \triangleq \mathbf{H}_{\mathrm{SI}} - \hat{\mathbf{H}}_{\mathrm{SI}}$ is the SI cancelation error and $\hat{\mathbf{H}}_{\mathrm{SI}}$ is the estimated SI channel.

Similarly, for the pilot dimension $\tau = N_{\mathrm{t}} + K$, the received pilot signal does not contain the signal from the SI channel.

Typically, the channels of nearby UEs are correlated, and for the pilot dimension $\tau = N_{\mathrm{t}}$ and $\tau = K$, the residual error of SI cancellation introduces further correlation in the received pilot signals used for UE channel estimation. We estimate the entire UE channel matrix $\mathbf{H}_{\mathrm{UE}}$. In this way, we take advantage of the correlation resulting from imperfect SI cancellation and leverage the additional information for more accurate estimation of the UE channels.

A.3 LS, MMSE, NN-based Channel Estimation

In this section, we present three channel estimation methods: LS, MMSE, and NN-based estimators for both SI and UE channels. First, we will recall the LS

and MMSE channel estimators. Subsequently, we present channel estimation using NNs for estimating the SI and UE channels. For SI channel estimation, we correlate the received pilot signal with the pilot matrix transmitted from transmit antenna arrays, therefore, we define

$$\hat{\mathbf{Y}}_{\mathrm{SI}} = \mathbf{Y}_{\mathrm{SI}} \mathbf{W}^{H}_{N_{\mathrm{t}} \times \tau}. \tag{A.14}$$

Similarly for the UE channel estimation, we have

$$\hat{\mathbf{Y}}_{\mathrm{UE}} = \mathbf{Y}_{\mathrm{UE}} \mathbf{W}^{H}_{K \times \tau}. \tag{A.15}$$

A.3.1 LS channel estimator

The LS channel estimator is the most simple channel estimation technique that does not need any prior knowledge about channel statistics and finds the channel coefficients that minimize the mean square error (MSE) between the estimated channel and the received pilot signal. The LS channel estimator can be calculated as follows

$$\hat{\mathbf{H}}_{q,\mathrm{LS}} = \frac{1}{\tau \sqrt{\mathrm{SNR}_{\mathrm{q}}}} \hat{\mathbf{Y}}_{q}, \tag{A.16}$$

where $q \in \{\mathrm{SI}, \mathrm{UE}\}$.

A.3.2 MMSE channel estimator

MMSE is a Bayesian estimator that aims to minimize the MSE between the estimated channel and the true channel. It offers improved performance compared to the LS estimator by taking into account the statistical properties of the channel and noise. To derive the MMSE channel estimator, we rewrite the received pilot signal for SI and UE channel estimation in a vector form

$$vec(\mathbf{Y}_{\mathrm{SI}}) = \sqrt{\mathrm{SNR}_{\mathrm{SI}}} \tilde{\mathbf{X}}_{\mathrm{SI}} vec(\mathbf{H}_{\mathrm{SI}}) + \sqrt{\mathrm{SNR}_{\mathrm{UE}}} \tilde{\mathbf{X}}_{\mathrm{UE}} vec(\mathbf{H}_{\mathrm{UE}}) + vec(\mathbf{N}), \tag{A.17}$$

$$vec(\mathbf{Y}_{\mathrm{UE}}) = \sqrt{\mathrm{SNR}_{\mathrm{UE}}} \tilde{\mathbf{X}}_{\mathrm{UE}} vec(\mathbf{H}_{\mathrm{UE}}) + \sqrt{\mathrm{SNR}_{\mathrm{SI}}} \tilde{\mathbf{X}}_{\mathrm{SI}} vec(\mathbf{E}_{\mathrm{SI}}) + vec(\mathbf{N}), \tag{A.18}$$

where $\tilde{\mathbf{X}}_{\mathrm{SI}} = (\mathbf{F}\mathbf{X}_{\mathrm{SI}})^T \otimes \mathbf{I}_{\mathbf{N}_\mathrm{r}}$ and $\tilde{\mathbf{X}}_{\mathrm{UE}} = (\mathbf{X}_{\mathrm{UE}})^T \otimes \mathbf{I}_{\mathbf{N}_\mathrm{r}}$.

The channel covariance matrix is defined as follows

$$\mathbf{R}_q \triangleq \mathbb{E}\left[vec(\mathbf{H}_q)vec(\mathbf{H}_q)^H\right]. \tag{A.19}$$

The channel covariance matrix for the MIMO channel can be related to the transmit and receive covariance matrix via the Kronecker product

$$\mathbf{R} = \mathbf{R}_\mathrm{t} \otimes \mathbf{R}_\mathrm{r}, \tag{A.20}$$

where $\mathbf{R}_\mathrm{t}$ and $\mathbf{R}_\mathrm{r}$ are the transmit and receive covariance matrices.

The MMSE channel estimator for $\mathbf{H}_q$, $\hat{\mathbf{H}}_{q,\mathrm{MMSE}}$, minimizes the mean square error (MSE) $\mathbb{E}\left[||\mathbf{H}_q - \hat{\mathbf{H}}_{q,\mathrm{MMSE}}||^2\right]$. We have

$$vec(\hat{\mathbf{H}}_{q,\mathrm{MMSE}}) = \mathbf{R}_{\mathbf{H}_q\mathbf{Y}_q}\mathbf{R}_{\mathbf{Y}_q}^{-1}vec(\mathbf{Y}_q), \tag{A.21}$$

where

$$\mathbf{R}_{\mathbf{H}_q\mathbf{Y}_q} \triangleq \mathbb{E}\left[vec(\mathbf{H}_q)vec(\mathbf{Y}_q)^H\right], \tag{A.22}$$

$$\mathbf{R}_{\mathbf{Y}_q} \triangleq \mathbb{E}\left[vec(\mathbf{Y}_q)vec(\mathbf{Y}_q)^H\right]. \tag{A.23}$$

We can formulate the MMSE estimates of SI and UE channels as in equations (A.24) and (A.25) provided at the top of the next page, respectively. In (A.25), $\mathbf{R}_\mathrm{E}$ represents the error covariance matrix of SI cancellation, which can be calculated as in (A.26).

$$\begin{aligned} vec(\hat{\mathbf{H}}_{\mathrm{SI,MMSE}}) = {} & \sqrt{\mathrm{SNR}_{\mathrm{SI}}}\,\mathbf{R}_{\mathrm{SI}}\tilde{\mathbf{X}}_{\mathrm{SI}}^H \\ & \cdot \Big(\mathrm{SNR}_{\mathrm{SI}}\tilde{\mathbf{X}}_{\mathrm{SI}}\mathbf{R}_{\mathrm{SI}}\tilde{\mathbf{X}}_{\mathrm{SI}}^H \\ & + \mathrm{SNR}_{\mathrm{UE}}\tilde{\mathbf{X}}_{\mathrm{UE}}\mathbf{R}_{\mathrm{UE}}\tilde{\mathbf{X}}_{\mathrm{UE}}^H + \mathbf{I}_{N_\mathrm{t}N_\mathrm{r}}\Big)^{-1}\mathrm{vec}(\mathbf{Y}_{\mathrm{SI}}). \end{aligned} \tag{A.24}$$

$$\begin{aligned} vec(\hat{\mathbf{H}}_{\mathrm{UE,MMSE}}) = {} & \sqrt{\mathrm{SNR}_{\mathrm{UE}}}\,\mathbf{R}_{\mathrm{UE}}\tilde{\mathbf{X}}_{\mathrm{UE}}^H \\ & \cdot \Big(\mathrm{SNR}_{\mathrm{UE}}\tilde{\mathbf{X}}_{\mathrm{UE}}\mathbf{R}_{\mathrm{UE}}\tilde{\mathbf{X}}_{\mathrm{UE}}^H \\ & + \mathrm{SNR}_{\mathrm{SI}}\tilde{\mathbf{X}}_{\mathrm{SI}}\mathbf{R}_{\mathrm{E}}\tilde{\mathbf{X}}_{\mathrm{UE}}^H + \mathbf{I}_{N_\mathrm{t}N_\mathrm{r}}\Big)^{-1}\mathrm{vec}(\mathbf{Y}_{\mathrm{UE}}). \end{aligned} \tag{A.25}$$

$$\mathbf{R}_{\mathrm{E}} = \mathbf{R}_{\mathrm{SI}} - \mathrm{SNR}_{\mathrm{SI}}\mathbf{R}_{\mathrm{SI}}\tilde{\mathbf{X}}_{\mathrm{SI}}^{H} \cdot \Big(\mathrm{SNR}_{\mathrm{SI}}\tilde{\mathbf{X}}_{\mathrm{SI}}\mathbf{R}_{\mathrm{SI}}\tilde{\mathbf{X}}_{\mathrm{SI}}^{H} + \mathrm{SNR}_{\mathrm{UE}}\tilde{\mathbf{X}}_{\mathrm{UE}}\mathbf{R}_{\mathrm{UE}}\tilde{\mathbf{X}}_{\mathrm{UE}}^{H} + \mathbf{I}_{N_{\mathrm{t}}N_{\mathrm{r}}}\Big)^{-1}\tilde{\mathbf{X}}_{\mathrm{SI}}\mathbf{R}_{\mathrm{SI}}. \quad \text{(A.26)}$$

Note that for pilot dimension $\tau = N_{\mathrm{t}} + K$, the MMSE estimates of the SI and UE channels simplify to the following equations using Woodbury matrix identity [37],

$$vec(\hat{\mathbf{H}}_{\mathrm{SI,MMSE}}) = \sqrt{\mathrm{SNR}_{\mathrm{SI}}}\left(\mathbf{R}_{\mathrm{SI}}{}^{-1} + \tau\mathrm{SNR}_{\mathrm{SI}}\mathbf{I}_{N_{\mathrm{t}}N_{\mathrm{r}}}\right)^{-1}\tilde{\mathbf{X}}_{\mathrm{SI}}^{H}vec(\mathbf{Y}_{\mathrm{SI}}), \quad \text{(A.27)}$$

$$vec(\hat{\mathbf{H}}_{\mathrm{UE,MMSE}}) = \sqrt{\mathrm{SNR}_{\mathrm{UE}}}\left(\mathbf{R}_{\mathrm{SI}}{}^{-1} + \tau\mathrm{SNR}_{\mathrm{UE}}\mathbf{I}_{N_{\mathrm{t}}N_{\mathrm{r}}}\right)^{-1}\tilde{\mathbf{X}}_{\mathrm{SI}}^{H}vec(\mathbf{Y}_{\mathrm{SI}}). \quad \text{(A.28)}$$

A.3.3 NN channel estimator

Although conventional channel estimation techniques such as MMSE can leverage the spatial correlation present in large antenna arrays, they require prior knowledge of the channel covariance matrix and are computationally intensive due to matrix inversion involving a large matrix. In various fields of research, it has been demonstrated that NNs can utilize the inherent structure of data by solely observing a sufficient number of data points. Different network architectures, such as FNNs, CNNs, recurrent neural networks (RNNs), etc., can be utilized depending on the characteristics of the data. For instance, CNNs are suitable for addressing problems with feature correlation and local dependencies among them, while RNNs are more effective for handling memory-based features like time series [17].

It has been shown in various studies that the wireless channel with large antenna arrays is spatially correlated, see e.g., [38]. To exploit this spatial correlation for channel estimation, we employ CNNs, which are suitable for capturing the local dependency of features among data points.

The overall architecture of CNN for SI and UE channel estimation is depicted in Fig. 3. The input to the CNN is the correlated pilot signal, i.e., $\hat{\mathbf{Y}}_q$, and the output is the estimated channel, $\hat{\mathbf{H}}_{q,\mathrm{NN}}$, $q \in \{\mathrm{SI}, \mathrm{UE}\}$. In convolutional layers, we apply a 3×3 window size sliding through the whole input features with a unit stride size. Different numbers of hidden layers are em-

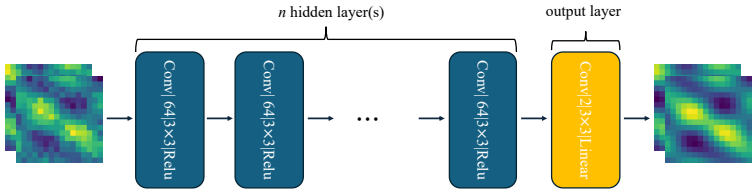


Figure 3: The CNN architecture employed for SI and UE channel estimation. we apply n ($n = 0, 1, 2, 10$) convolutional hidden layer(s) each with a 3×3 window size and 64 convolutional kernels.

ployed, where each layer applies 64 convolutional kernels to extract features from the successive windows of its input features. The effect of the number of hidden layers will be carefully examined in Section A.4.

To keep the dimensions of the output and input fixed, we utilize padding after convolution processing. We apply rectified linear unit (ReLU) as the activation function for the hidden layers, while for the output layer, linear activation is used. Since tensors do not support complex operations, the input to the CNN is converted to three-dimensional tensors, where the third dimension stores the real and imaginary parts of the complex data samples. Therefore, if we define $\mathbf{X}_{\text{tr}}$ and $\mathbf{Y}_{\text{tr}}$ as the input and labels of the CNN during training, we have

$$\begin{aligned} \mathbf{X}_{\text{tr}}[:,:,0] &\triangleq \Re\left\{\hat{\mathbf{Y}}_q\right\}, \\ \mathbf{X}_{\text{tr}}[:,:,1] &\triangleq \Im\left\{\hat{\mathbf{Y}}_q\right\}. \end{aligned} \tag{A.29}$$

$$\begin{aligned} \mathbf{Y}_{\text{tr}}[:,:,0] &\triangleq \Re\left\{\mathbf{H}_q\right\}, \\ \mathbf{Y}_{\text{tr}}[:,:,1] &\triangleq \Im\left\{\mathbf{H}_q\right\}. \end{aligned} \tag{A.30}$$

A training dataset comprising M_{tr} samples is generated, where the n-th sample is given by $\left(\hat{\mathbf{Y}}_q^{(n)}, \mathbf{H}_q^{(n)}\right)$. We employ supervised learning, where $\mathbf{H}_q$ is regarded as the label during training. To obtain the ground truth channel samples, $\mathbf{H}_q^{(n)}$, for SI and UE channels, we follow the channel model introduced in Section A.2. Each realization of data samples in the dataset is generated based on this channel model, where, in each realization, the small-scale fading coefficient, large-scale fading, and AoA/AoD take on a new realization based on their corresponding distributions. Specifically, we draw samples from $\mathcal{CN}(0, 1)$

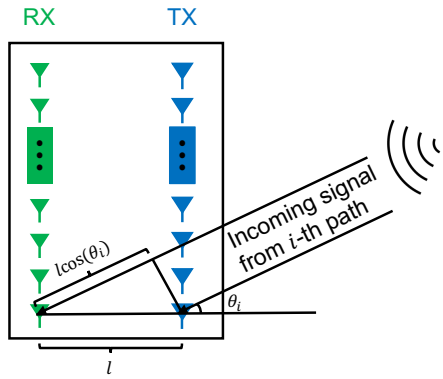


Figure 4: Phase shift between the RX and TX arrays in the separate antenna configuration in a full-duplex BS.

for every multi-path component for small-scale fading, $\mathcal{U}\left[-\frac{\theta_{\text{AS}}}{2}, \frac{\theta_{\text{AS}}}{2}\right]$ for every multi-path component for AoA/AoD, and $\mathcal{LN}(0, \sigma_{sf})$ for shadow fading in large-scale fading. After generating channel realizations, the received pilot signal for SI and UE channel estimation, $\hat{\mathbf{Y}}_q^{(n)}$, is generated by adding Gaussian noise to every realization and multiplying by the pilot matrix corresponding to different pilot dimensions. We apply min-max scaling to scale the dataset in the range $(0, 1)$. Such normalization is highly recommended for training machine learning models. We consider MSE as the loss function

$$\text{MSE} = \frac{1}{M_{\text{tr}}} \sum_{i=1}^{M_{\text{tr}}} ||\mathbf{X}_{\text{tr}} - \mathbf{Y}_{\text{tr}}||_F^2. \tag{A.31}$$

A.3.4 RX-TX channel mapping

Since in separate antenna configurations, two different antenna arrays are utilized for uplink reception and downlink transmission, uplink and downlink channels experience different channel realizations. So far we have assumed that RX arrays at BS receive pilot signals from both uplink and downlink UEs and estimate all channels from both uplink and downlink UEs to the RX arrays. For downlink transmission, the full-duplex BS needs the CSI from the TX arrays to downlink UEs. The TX arrays are not capable of receiving and processing pilot signals and implementing a separate receive RF chain and

ADC at the TX arrays for just pilot processing will be costly. As a result, we need to somehow map the channel from the downlink UEs to the RX arrays to the channel from the TX arrays to the downlink UEs. Due to the multi-path effect and random scattering environment, it is not straightforward to derive mathematically the relation of such mapping. More specifically, we assume that the RX-UE channel is

$$\mathbf{h}_{\mathrm{UE,RX}} = \sqrt{\frac{1}{N_{\mathrm{r}}}}\sqrt{\beta_{\mathrm{u}}}\sum_{i=0}^{P-1}\alpha_i \mathbf{a}_{\mathrm{a}}(\theta_i), \tag{A.32}$$

where a_i and θ_i are the amplitude and the AoA of i-th path, respectively. As shown in Fig. 4, TX and RX arrays are at a close distance in the order of a few wavelengths from each other; Therefore, the channel amplitude for each path is essentially the same for both the RX and TX arrays, while the antenna separation creates a delay depending on the AoA of each path. Thus, the TX-UE channel will be

$$\mathbf{h}_{\mathrm{UE,TX}} = \sqrt{\frac{1}{N_{\mathrm{t}}}}\sqrt{\beta_{\mathrm{d}}}\sum_{i=0}^{P-1}\alpha_i \mathbf{a}_{\mathrm{d}}(\theta_i) e^{-j\frac{2\pi}{\lambda} l cos\theta_i}. \tag{A.33}$$

The mathematical relationship between $\mathbf{h}_{\mathrm{UE,TX}}$ and $\mathbf{h}_{\mathrm{UE,RX}}$ is not well-defined due to the random AoA and the effects of multi-path propagation. On the other hand, based on universal approximation theory [30], feedforward NNs are capable of approximating any continuous function. This theory suggests that, given enough computational resources and data, it is possible to build NN models that can accurately approximate any function. Therefore, universal approximation theory inspires us to employ NNs to map the channel from the downlink UEs to the RX array to the channel from the TX array to the downlink UEs. Furthermore, recent studies [29] have demonstrated the existence of a spatial mapping function that can effectively map the channel from one set of antenna arrays to another. To accomplish this, we utilize an FNN with varying numbers of hidden layers. The input for FNN is the channel from the downlink UEs to the RX array, while the output is the channel from the TX array to the downlink UEs. Therefore, for the training of FNN, we collect M_{tr} samples of $\mathbf{h}_{\mathrm{UE,RX}}$ as the input and $\mathbf{h}_{\mathrm{UE,TX}}$ as the corresponding label. To generate data samples for RX-TX channel mapping, we first generate the ground truth channel realizations for $\mathbf{h}_{\mathrm{UE,RX}}$, and their corresponding

labels for $\mathbf{h}_{\mathrm{UE,TX}}$ are generated by adding its corresponding delay to each multi-path component. Again, we apply min-max normalization and MSE loss function for training. We create the following data samples as the input and label of FNN to work with real-valued tensors:

$$\begin{aligned}\mathbf{x}_{\mathrm{tr}}[0:N_{\mathrm{r}}] &= \Re\left\{\mathbf{h}_{\mathrm{UE,RX}}\right\},\\ \mathbf{x}_{\mathrm{tr}}[N_{\mathrm{r}}:2N_{\mathrm{r}}] &= \Im\left\{\mathbf{h}_{\mathrm{UE,RX}}\right\}.\end{aligned} \tag{A.34}$$

$$\begin{aligned}\mathbf{y}_{\mathrm{tr}}[0:N_{\mathrm{t}}] &= \Re\left\{\mathbf{h}_{\mathrm{UE,TX}}\right\},\\ \mathbf{y}_{\mathrm{tr}}[N_{\mathrm{t}}:2N_{\mathrm{t}}] &= \Im\left\{\mathbf{h}_{\mathrm{UE,TX}}\right\}.\end{aligned} \tag{A.35}$$

A.4 Simulation results

In this section, we present our simulation results for SI and UE channel estimation. We compare channel estimators that we introduced in previous sections, i.e., LS, MMSE, and NN for different pilot dimensions. The NMSE is considered the performance metric to compare the different channel estimators, and it is defined as

$$\mathrm{NMSE} \triangleq \mathbb{E}\left[\frac{||\mathbf{H}_{\mathrm{true}} - \mathbf{H}_{\mathrm{est}}||_F^2}{||\mathbf{H}_{\mathrm{true}}||_F^2}\right], \tag{A.36}$$

where $\mathbf{H}_{\mathrm{true}}$ and $\mathbf{H}_{\mathrm{est}}$ are the true and estimated channel matrices, respectively.

We consider an operating frequency is 28 GHz, corresponding to a wavelength of $\lambda = 10.71$ mm. We set the number of TX and RX arrays to 16, with a distance of 10λ between them, and antenna spacing in the transmit and the receive arrays is $d = \frac{\lambda}{2}$. The total number of downlink and uplink UEs is set to $K = 8$ and we assume all UEs are uniformly distributed within 20 m^2. The path loss parameters are based on experimental results from [39], where the path loss constant at the reference distance is $\Gamma = -72$ dB, the path loss exponent is $\eta = 2.92$, and the shadow fading standard deviation is $\sigma_{\mathrm{sf}} = 8.7$ dB. Unless otherwise claimed, the number of multi-path components is $P = 5$ and AS is $\theta_{\mathrm{AS}} = 60°$. The SI Rician factor is $\kappa = 40$ dB and we assume that the propagation SI cancellation is $\epsilon_{\mathrm{SI}} = -40$ dB [12].

The NN-based channel estimators are trained on Python 3.7.16 and implemented using the Keras libraries with a TensorFlow backend in the Jupyter Notebook environment. We employ Adam optimizer with a batch size of 512

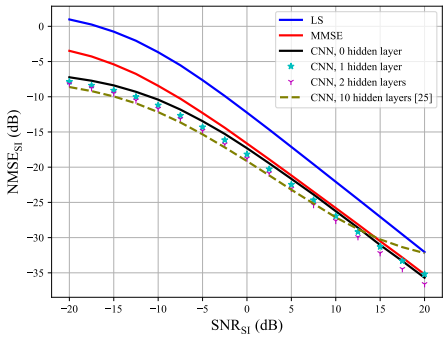


Figure 5: $\mathrm{NMSE}_{\mathrm{SI}}$ vs $\mathrm{SNR}_{\mathrm{SI}}$ with a varying number of hidden layers.

to update the network parameters. A dataset of $50,000$ samples is collected based on the channel model and it is split into $20,000$ samples for training, $20,000$ samples for validation, and $10,000$ samples for testing. The validation data is used to ensure that the model does not simply memorize the training data but learns meaningful aspects of the data for effective prediction.

A.4.1 Analysing NN-based channel estimator

Before examining the different channel estimators for various pilot dimensions discussed in subsection A.2.2, we conducted several experiments to understand the behavior of NN-based channel estimators under different channel conditions and NN architectures for training. The following results are specific to SI channel estimation with a pilot dimension of $\tau = N_{\mathrm{t}} + K$. For the sake of brevity, we have not included plots for other pilot dimensions and UE channel estimation. For the MMSE channel estimator, we obtain the empirical channel covariance matrix estimated through M channel realizations, i.e.,

$$\mathbf{R}_q = \frac{1}{M} \sum_{i=1}^{M} vec(\mathbf{H}_q^{(i)}) vec(\mathbf{H}_q^{(i)})^H, \tag{A.37}$$

where throughout our simulation, we set $M = 1000$.

Fig. 5 illustrates the $\mathrm{NMSE}_{\mathrm{SI}}$ versus $\mathrm{SNR}_{\mathrm{SI}}$ with different numbers of hidden layers. We can observe that NN-based channel estimator outperforms

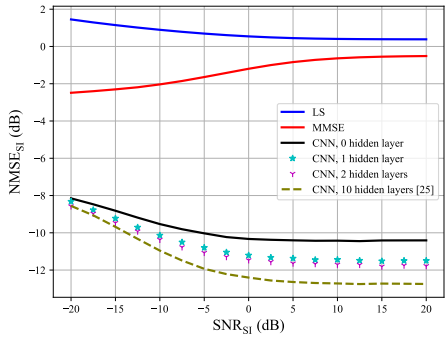


Figure 6: NMSE_{SI} vs SNR_{SI} with a varying number of hidden layers for a full-duplex BS with 1-bit ADCs.

the LS and MMSE estimation. Furthermore, utilizing hidden layers slightly improves the NMSE of channel estimates. For instance, CNN with 2 hidden layers decreases the NMSE by about 1 dB compared to CNN with no hidden layer. However, comparing CNN with 10 hidden layers (DeepCNN) with that of 1 or 2 hidden layers suggests that using very deep CNNs does not noticeably improve the channel estimates (about 0.3 dB NMSE improvement), while significantly increasing computational complexity. Furthermore, the erratic behavior of DeepCNN at high SNRs can be rectified through fine-tuning, for example, by using a different number of convolutional kernels, adjusting the learning rate, etc. However, for the sake of comparison, we applied the exact same parameters to all CNNs with different numbers of hidden layers. In practical scenarios, various non-linear distortions in hardware components exist, adding complexity to the problem of channel estimation. To further analyze the impact of the number of hidden layers, we consider the introduction of 1-bit ADCs at the BS to incorporate a non-linear distortion effect into the received pilot signal. When employing 1-bit ADCs, the received pilot signal becomes [27]

$$\mathbf{Y}_{\text{1-bit}} = \text{sgn}(\mathbf{Y}_{\text{SI}}), \tag{A.38}$$

where $\text{sgn}(\cdot)$ is the element-wise signum function.

Once again, we apply CNN with a varying number of hidden layers to address the channel estimation problem with 1-bit ADCs. In this case, the input

Table 1: Optimal uniform quantizer [40]

Resolution (b)	Step size (Δ_b)	Distortion (η_b)
1-bit	$\sqrt{\frac{8}{\pi}}$	$1-\frac{2}{\pi}$
2-bit	0.996	0.1188
3-bit	0.586	0.0374
4-bit	0.335	0.0115

to the CNN is $\hat{\mathbf{Y}}_{\text{1-bit}} = \mathbf{Y}_{\text{1-bit}}\mathbf{W}^{H}_{N_{\text{t}}\times\tau}$, and the output is the estimated channel. Fig. 6 presents the NMSE of LS, MMSE, and NN-based channel estimators. As observed in the figure, the addition of hidden layers improves the NMSE by approximately 1 dB, which is similar to the improvement observed in the case of infinite-bit ADCs in Fig. 5.

To delve deeper into the problem of channel estimation with low-resolution ADCs, we followed the study in [40]. We assume that ADCs apply b-bit uniform scalar quantization with a set of 2^b-1 thresholds denoted as $\{\tau_1,\ldots,\tau_{2^b-1}\}$. The thresholds of the uniform quantizer will be

$$\tau_l = (-2^{b-1}+l)\Delta_b, \quad l \in L = \{1,\ldots,2^b-1\}, \tag{A.39}$$

where Δ_b is the step size.

The quantization function is independently applied to the real and imaginary components of the signal, with its definition as follows:

$$Q_b(r) = \begin{cases} \tau_l - \frac{\Delta_b}{2} & \text{if } r \in (\tau_{l-1},\tau_l] \text{ with } l \in L, \\ (2^b-1)\frac{\Delta_b}{2} & \text{if } r \in (\tau_{2^b-1},\tau_{2^b}]. \end{cases} \tag{A.40}$$

Therefore, the received pilot signal after b-bit ADC follows:

$$\mathbf{Y}_{\text{b-bit}} = Q_b(\mathbf{Y}_{\text{SI}}), \tag{A.41}$$

The quantization step size is usually chosen to minimize the quantization MSE, and the optimal values are listed in Table 1 for their corresponding bits. The quantization distortion, η_b is defined as

$$\eta_b \triangleq \mathbb{E}\left[\frac{||Q_b(r)-r||_F^2}{||r||_F^2}\right], \tag{A.42}$$

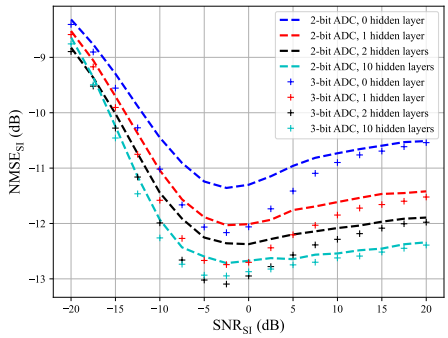
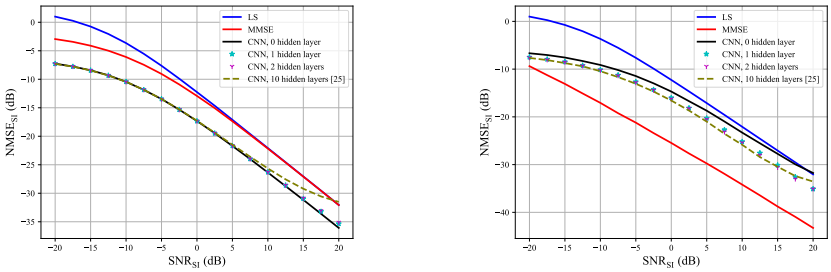


Figure 7: NMSE_{SI} vs SNR_{SI} with a varying number of hidden layers for a full-duplex BS with few-bit ADCs.

We conducted simulations for 2-bit and 3-bit ADCs, and similar results can be extrapolated for higher resolutions. From Fig. 7, we can observe that increasing the number of hidden layers improves estimation with 2-bit and 3-bit ADCs, similar to the results for the 1-bit case. Furthermore, as expected, the higher the resolution of ADCs, the better the NMSE of channel estimates.

Then, we examine the behavior of the NN-based channel estimator under two distinct channel correlation conditions: high-correlated and low-correlated scenarios. The plots in Fig. 8 illustrate how the LS, MMSE, and NN-based estimators perform under different spatial channel correlation strengths. Larger θ_{AS} corresponds to lower spatial correlations, and vice versa. In the high-correlated scenario, the MMSE estimator can explicitly leverage the channel covariance matrix, resulting in a significant improvement in estimation quality. This leads to a substantial gap between the LS and MMSE estimations. However, in the low-correlated channel, both the LS and MMSE estimations converge to the same NMSE at high SNRs. Comparing the NN-based estimation with the LS and MMSE estimations in both low-correlated and high-correlated channel conditions provides interesting insights into the behavior of the NN-based estimator. In the low-correlated scenario, the NN-based estimator outperforms both the MMSE and LS estimations. However, in highly correlated channel conditions, the MMSE estimator consistently outperforms the NN-based estimator. This observation suggests that the NN-based estimator, in a data-driven fashion, struggles to utilize the second-order statistics of



(a) $\theta_{\mathrm{AS}} = 360°$ **(b)** $\theta_{\mathrm{AS}} = 10°$

Figure 8: $\mathrm{NMSE}_{\mathrm{SI}}$ vs $\mathrm{SNR}_{\mathrm{SI}}$ for (a) low-correlated and (b) high-correlated channels.

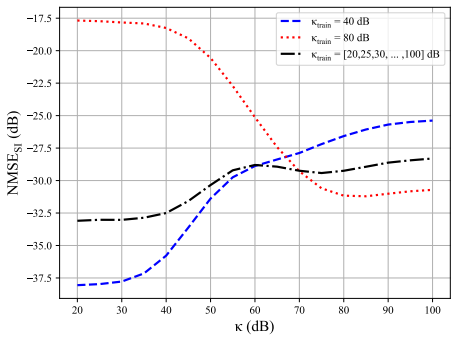


Figure 9: $\mathrm{NMSE}_{\mathrm{SI}}$ vs κ with different values of κ during training (κ_{train}).

the channel for estimation, as well as the MMSE estimator does. Furthermore, the plots suggest that in a low-correlated scenario, increasing the number of hidden layers does not yield any noticeable improvement. However, in a highly correlated channel condition, the NMSE decreases with the addition of more hidden layers to the NN architecture.

Therefore, the choice of the depth of NN architecture is subject to the channel condition and cannot be regarded as universally applicable. Nevertheless, this observation underscores the significance of channel statistics in determining the architecture of NN for the channel estimation problem.

Finally, Fig. 9 shows the NMSE of SI channel estimation versus κ with

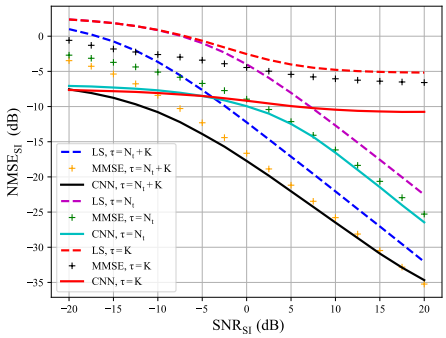


Figure 10: NMSE_{SI} vs SNR_{SI} for different pilot dimensions, $\text{SNR}_{\text{UE}} = 0$ dB.

different values of κ_{train} during training. We exclude the plots with 1, 2, and 10 hidden layers for enhanced clarity. As seen in this figure, the NMSE of SI channel estimates behaves differently depending on the value of κ_{train}. Specifically, when the NN is trained on low κ_{train}, the NMSE_{SI} increases as κ increases. Conversely, when the NN is trained on high κ_{train}, the NMSE_{SI} decreases as κ increases. However, training on different values of κ_{train} ranging from 20 to 100 dB shows that higher values of κ lead to higher NMSE_{SI}. While it seems that in the large κ regime, estimating the SI channel would be simpler for the NN because it is dominated by the near-field component $\mathbf{H}_{\text{SI,NF}}$ and thus more deterministic, we should highlight that we trained our NN with various values of κ in the black dash-dot curve. Therefore, the training data included the stochastic $\mathbf{H}_{\text{SI,FF}}$. During testing, the NN encounters a deterministic channel in the very large κ regime, which differs from the training conditions, leading to suboptimal performance compared to when κ is not very large. This behavior can be connected to the distribution shifts in the NNs that we will discuss in subsection A.4.3. Specifically, in the very large κ regime, the data distribution becomes more deterministic, whereas the training data included stochastic data from smaller κ values.

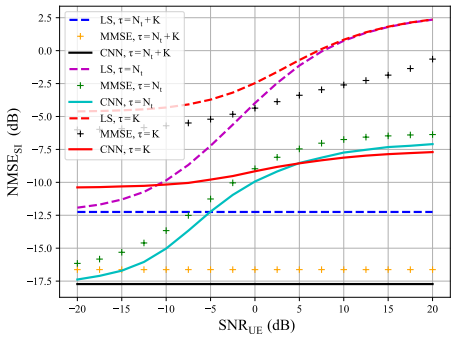


Figure 11: $\mathrm{NMSE}_{\mathrm{SI}}$ vs $\mathrm{SNR}_{\mathrm{UE}}$ for different pilot dimensions, $\mathrm{SNR}_{\mathrm{SI}} = 0$ dB.

A.4.2 SI and UE channel estimation for different pilot dimensions

Next, we examine the performance of SI channel estimation for different pilot dimensions, as described in Section A.2.2. For the following simulations, we applied a CNN with no hidden layers and set $\theta_{\mathrm{AS}} = 60°$.

The NMSE of SI channel estimation vs $\mathrm{SNR}_{\mathrm{SI}}$ and $\mathrm{SNR}_{\mathrm{UE}}$ are provided in Figs. 10 and 11 for different pilot dimensions, respectively. The results demonstrate that the NN-based estimator outperforms the LS and MMSE estimators across considered pilot dimensions. Moreover, from Fig. 11, it is evident that the NN-based approach is more resilient to interference from UEs when compared to the MMSE and LS estimation methods. Using fewer pilot dimensions leads to a significant reduction in the quality of channel estimates for all three approaches. However, the reduction in pilot dimensions can allow more resources to be allocated to transmitting payload data. The trade-off between the performance of channel estimates and the number of pilot dimensions used depends on the required accuracy threshold for channel estimates and the system data rate.

We have also generated similar plots for UE channel estimation, showing the NMSE with respect to $\mathrm{SNR}_{\mathrm{UE}}$ and $\mathrm{SNR}_{\mathrm{SI}}$. The plots are presented in Figs. 12 and 13, respectively, with varying pilot dimensions. We employ the MMSE estimator to cancel out the SI signal from the received pilot signal in LS, MMSE, and NN-based channel estimators for UE channel estimation,

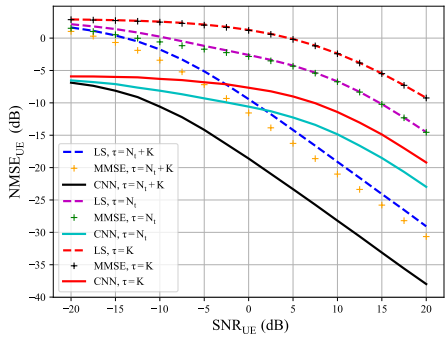


Figure 12: $\mathrm{NMSE}_{\mathrm{UE}}$ vs $\mathrm{SNR}_{\mathrm{UE}}$ for different pilot dimensions, $\mathrm{SNR}_{\mathrm{SI}} = 10$ dB.

specifically when the pilot dimensions are $\tau = N_{\mathrm{t}}$ and $\tau = K$. Note that we also utilize the error covariance matrix for the MMSE estimator when estimating UE channels as denoted in (A.25). From the results, it is evident that the NN-based approach using shorter pilot dimensions like $\tau = N_{\mathrm{t}}$ or $\tau = K$, outperforms LS channel estimation with a pilot dimension of $\tau = N_{\mathrm{t}} + K$. Even the MMSE channel estimator with longer pilot dimensions (e.g., $\tau = N_{\mathrm{t}} + K$), is unable to outperform the NN-based technique with shorter pilot dimensions (e.g., $\tau = N_{\mathrm{t}}$). These findings clearly demonstrate the superiority of NN in accurately estimating wireless channels, particularly in scenarios with lower SNRs and higher interference.

A higher $\mathrm{SNR}_{\mathrm{SI}}$ results in better SI channel estimation and, consequently, less error in SI cancellation for UE channel estimation. Conversely, a higher $\mathrm{SNR}_{\mathrm{UE}}$ increases the NMSE of SI channel estimation, leading to an increase in error for UE channel estimation. On the other hand, a higher $\mathrm{SNR}_{\mathrm{UE}}$ leads to improved UE channel estimates, while a higher $\mathrm{SNR}_{\mathrm{SI}}$ increases the power of interference during UE channel estimation. To understand the joint effect of $\mathrm{SNR}_{\mathrm{UE}}$ and $\mathrm{SNR}_{\mathrm{SI}}$ for UE channel estimation, we generated the color bar plots in Fig. 14 for LS, MMSE, and NN-based estimation considering a pilot dimension $\tau = K$. Based on the results in this figure, we can conclude that a lower SI signal power or higher SI cancelation will lead to better UE channel estimates.

To further analyze the effect of SI cancelation on UE channel estimation, we considered two cases: in the first case, we cancel out the effect of the SI

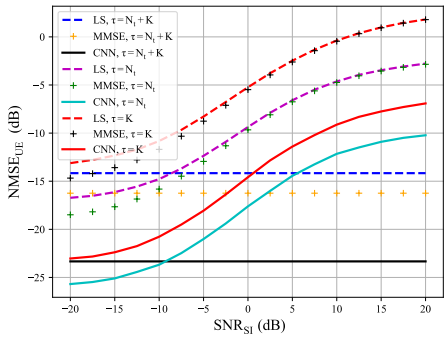
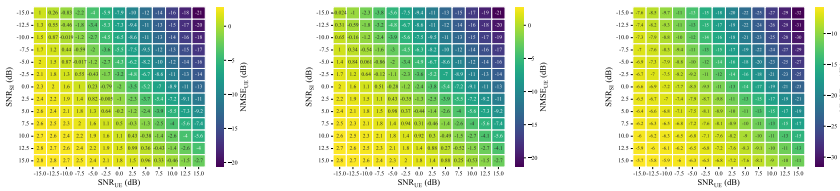


Figure 13: $\mathrm{NMSE_{UE}}$ vs $\mathrm{SNR_{SI}}$ for different pilot dimensions, $\mathrm{SNR_{UE}} = 5$ dB.



(a) LS **(b)** MMSE **(c)** NN

Figure 14: $\mathrm{NMSE_{UE}}$ vs $\mathrm{SNR_{SI}}$ and $\mathrm{SNR_{UE}}$ for (a) LS, (b) MMSE, and (c) NN channel estimators, $\tau = K$.

signal from the received pilot signal with the estimated SI channel, while in the second scenario, we estimate UE channels in the presence of interference from the SI channel. The results are shown in Fig 15. Note that for the MMSE estimation, in the case of SI cancellation, we incoporate the error covariance matrix of the estimated SI channel together with the covariance matrix of the UE channels, while for the scenario without SI cancellation, we exploit the covariance matrix of both SI and UE channels. We can observe that MMSE estimations for these two scenarios result in the same NMSE of UE channel estimates. By comparing the results for LS, MMSE, and NN channel estimators, we observe that SI cancellation during the pilot transmission phase does not provide significant improvement, while estimating the large SI channel matrix is costly.

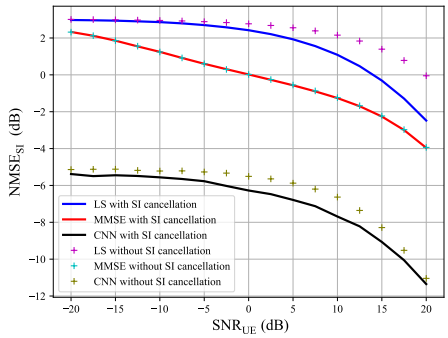


Figure 15: UE channel estimation with/without SI cancellation, $\tau = K$, $\text{SNR}_{\text{SI}} = 20$ dB.

To assess the performance of the NN-based channel mapping between RX and TX arrays at the BS in the separate antenna configuration, we present the NMSE of this mapping concerning varying angular spreads and two different numbers of antennas in Fig. 16. We trained an FNN with a varying number of hidden layers at angular spreads of $[10, 100, 190, 280]$ degrees during training. From this figure, we can conclude that an NN effectively predicts the channel from the TX arrays to downlink UEs, given the CSI from downlink UEs to the RX arrays. The prediction accuracy is higher in scenarios with lower angular spread, attributed to increased spatial correlation, and vice versa. Furthermore, the quality of prediction improves with an increased number of antennas, as angular resolutions increase with a larger array.

A.4.3 Distribution shift

When a learning model is trained on a particular dataset with a specific distribution of features, it learns to make predictions based on the patterns inherent in that dataset. However, deploying such models in real-world scenarios often involves testing them with datasets that may differ in distribution from the training data. This misalignment between training and test distributions can lead to a phenomenon known as distribution shift, where the model's performance deteriorates due to its inability to generalize effectively to unseen data and it poses significant challenges in testing the efficacy and reliability

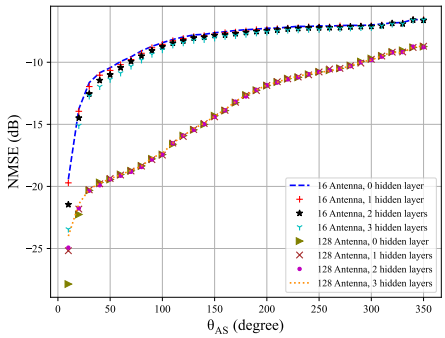


Figure 16: NMSE of channel mapping between RX and TX arrays vs angular spread.

of learning models.

To analyze the effect of distribution shift on our trained models, we utilize the DeepMIMO dataset in the test phase. This dataset constructs the MIMO channels based on ray-tracing simulation from Remcom Wireless InSite [41]. Specifically, we use the 'O1' scenario at 28 GHz with the parameters specified in Table 2. We generate 4977 channel realizations from the DeepMIMO dataset to test the NN model for UE channel estimation trained based on the channel data from geometrical channel models introduced in Section A.2.1. The same can be applied for SI channel estimation, but for brevity, we do not include the results for this case. As seen from Fig. 17, due to distribution mismatch, the NMSE degrades in the test phase. Furthermore, we observe that NNs with fewer hidden layers generalize better compared to deeper CNNs since the number of trainable parameters is much higher in deep CNNs. This underscores the benefits of utilizing simpler NN architectures in terms of generalization abilities.

A.4.4 Complexity Analysis

Finally, we compare the computational complexity of the LS, MMSE, and NN channel estimators in terms of FLOPs. The results are summarized in Table 3, where ζ represents the side length of the convolutional window, and f_{l-1} and f_l denote the numbers of input and output convolutional kernel of the l-th

Table 2: DeepMIMO dataset parameters

DeepMIMO parameters	**Value**
Number of paths	10
Active UEs	from row 550 to 1650
Active BS number	BS 3
Bandwidth	50 MHz
BS antennas	$N_x = 1$, $N_y = 16$, $N_z = 1$
UE antennas	$N_x = 1$, $N_y = 1$, $N_z = 1$

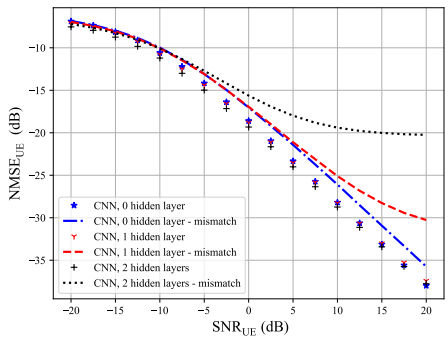


Figure 17: Performance of NN channel estimator for UE channel estimation with distribution shift.

layer. A bar plot is shown in Fig. 18 for SI channel estimation and pilot dimension $\tau = N_{\text{t}}$ based on the CNN architecture used in the simulation, i.e., $\zeta = 3$, $f_0 = f_L = 2$, and $f_i = 64$ for the remaining layers ($i = 1, 2, \ldots, L-1$). Note that we have ignored the computational complexity of the covariance matrix calculation for MMSE estimation, as the channel statistics do not typically change during several coherence blocks. For the same reasons, we also ignored the computational complexity of the matrices in the MMSE formula that depend on spatial correlation matrices, assuming they can be pre-computed and used until the channel statistics change substantially. According to table 3, the number of FLOPs exhibits quadratic growth with the number of received antennas in MMSE estimation, while for LS and NN-based estimation, it is linear. Furthermore, CNNs with no or one hidden layer require fewer FLOPs

Table 3: Number of FLOPs for SI and UE channel estimation using LS, MMSE, and NN estimators.

	LS	MMSE	CNN
SI	$N_{\mathrm{r}}N_{\mathrm{t}}\tau$	$N_{\mathrm{r}}^2N_{\mathrm{t}}\tau$	$N_{\mathrm{r}}N_{\mathrm{t}}\tau + N_{\mathrm{r}}N_{\mathrm{t}}\sum_{l=1}^{L}\zeta^2 f_{l-1}f_l$
UE	$N_{\mathrm{r}}K\tau$	$N_{\mathrm{r}}^2K\tau$	$N_{\mathrm{r}}K\tau + N_{\mathrm{r}}K\sum_{l=1}^{L}\zeta^2 f_{l-1}f_l$

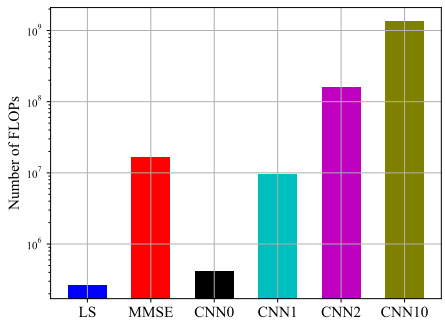


Figure 18: Number of FLOPs for SI channel estimation of different estimators. CNNx represents a CNN with x hidden layer(s).

than MMSE estimation, while increasing the number of hidden layers significantly increases the computational complexity. In particular, adding a hidden layer with 64 convolutional kernels and a 3×3 window size requires about 1.5×10^8 more FLOPs for estimating a MIMO channel with 16 antennas.

A.5 Conclusion

In this paper, we studied the channel estimation problem for full-duplex mmWave MIMO systems using NNs. We show that simple NNs with no or few hidden layers can achieve comparable NMSE to deep NNs. This conclusion holds to some extent even in the presence of few-bit ADC distortion, with additional hidden layers only slightly improving the NMSE of channel estimates. Furthermore, simpler NN architectures with fewer hidden layers demonstrate more robust generalization abilities compared to deep NNs. The comparison between the NN and MMSE channel estimators indicates a notable difference.

In the high-correlated regime, MMSE is the preferable channel estimator compared to LS and NN, as it can leverage the explicit channel covariance matrix. However, in scenarios where spatial correlation is not extremely high, our results demonstrate that NNs outperform MMSE estimation. Additionally, the NN-based channel estimator performs remarkably better than MMSE when fewer pilot resources are utilized for channel estimation. This performance allows for a reduction in pilot overhead in full-duplex systems. We applied FNNs to approximate RX-TX channel mapping for the separate antenna configurations in full-duplex systems. Simulation results demonstrate that NNs effectively map channels from downlink UEs to the receive arrays to the channel from the transmit arrays to downlink UEs, particularly in scenarios with high correlation and large antenna arrays. Finally, the complexity analysis reveals that NN-based estimation with fewer than two convolutional hidden layers requires fewer FLOPs compared to MMSE estimation.

References

[1] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[2] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan, and R. Wichman, "In-band full-duplex wireless: Challenges and opportunities," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 9, pp. 1637–1652, Jun. 2014.

[3] I. P. Roberts, J. G. Andrews, H. B. Jain, and S. Vishwanath, "Millimeter-wave full duplex radios: New challenges and techniques," *IEEE Wireless Commun.*, vol. 28, no. 1, pp. 36–43, Feb. 2021.

[4] B. Yu, C. Qian, J. Lee, *et al.*, "Realizing high power full duplex in millimeter wave system: Design, prototype and results," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 9, pp. 2893–2906, Jun. 2023.

[5] M. Duarte, C. Dick, and A. Sabharwal, "Experiment-driven characterization of full-duplex wireless systems," *IEEE Trans. Wirel. Commun.*, vol. 11, no. 12, pp. 4296–4307, Nov. 2012.

[6] B. P. Day, A. R. Margetts, D. W. Bliss, and P. Schniter, "Full-duplex bidirectional MIMO: Achievable rates under limited dynamic range," *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3702–3713, Apr. 2012.

[7] Z. Zhang, Z. Chen, M. Shen, B. Xia, W. Xie, and Y. Zhao, "Performance analysis for training-based multipair two-way full-duplex relaying with massive antennas," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 6130–6145, Dec. 2017.

[8] X. Xiong, X. Wang, T. Riihonen, and X. You, "Channel estimation for full-duplex relay systems with large-scale antenna arrays," *IEEE Trans. Wirel. Commun.*, vol. 15, no. 10, pp. 6925–6938, Jul. 2016.

[9] A. Masmoudi and T. Le-Ngoc, "Channel estimation and self-interference cancelation in full-duplex communication systems," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 321–334, Mar. 2017.

[10] D. Korpi, T. Riihonen, and M. Valkama, "Achievable rate regions and self-interference channel estimation in hybrid full-duplex/half-duplex radio links," in *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, USA, Apr. 2015, pp. 1–6.

[11] D. Korpi, L. Anttila, and M. Valkama, "Impact of received signal on self-interference channel estimation and achievable rates in in-band full-duplex transceivers," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, Apr. 2014, pp. 975–982.

[12] J. M. B. da Silva, A. Sabharwal, G. Fodor, and C. Fischione, "1-bit phase shifters for large-antenna full-duplex mmwave communications," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 10, pp. 6916–6931, Jul. 2020.

[13] Y.-G. Lim, D. Hong, and C.-B. Chae, "Performance analysis of self-interference cancellation in full-duplex large-scale MIMO systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, USA, Feb. 2016, pp. 1–6.

[14] Q. Ding, Y. Lian, and Y. Jing, "Performance analysis of full-duplex massive MIMO systems with low-resolution ADCs/DACs over rician fading channels," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7389–7403, Apr. 2020.

[15] J. Koh, Y.-G. Lim, C.-B. Chae, and J. Kang, "On the feasibility of full-duplex large-scale MIMO cellular systems," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 9, pp. 6231–6250, Jul. 2018.

[16] I. P. Roberts, J. G. Andrews, and S. Vishwanath, "Hybrid beamforming for millimeter wave full-duplex under limited receive dynamic range," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 12, pp. 7758–7772, Jun. 2021.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, Nov. 2016, ISBN: 0262035618.

[18] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2184–2199, Sep. 2019.

[19] M. B. Mashhadi and D. Gündüz, "Pruning the pilots: Deep learning-based pilot design and channel estimation for MIMO-OFDM systems," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 10, pp. 6315–6328, Apr. 2021.

[20] H. Huang, Y. Song, J. Yang, G. Gui, and F. Adachi, "Deep-learning-based millimeter-wave massive MIMO for hybrid precoding," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 3027–3032, Jan. 2019.

[21] M. Khani, M. Alizadeh, J. Hoydis, and P. Fleming, "Adaptive neural signal detection for massive mimo," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 8, pp. 5635–5648, May 2020.

[22] M. B. Mashhadi, Q. Yang, and D. Gündüz, "Distributed deep convolutional compression for massive mimo csi feedback," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2621–2633, 2021.

[23] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132–143, Dec. 2018.

[24] H. Ye, G. Y. Li, B. F. Juang, and K. Sivanesan, "Channel agnostic end-to-end learning based communication systems with conditional GAN," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Batam, Indonesia, Jan. 2018, pp. 1–5.

[25] P. Dong, H. Zhang, G. Y. Li, I. S. Gaspar, and N. NaderiAlizadeh, "Deep CNN-based channel estimation for mmwave massive MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 5, pp. 989–1000, Jul. 2019.

[26] E. Balevi, A. Doshi, and J. G. Andrews, "Massive MIMO channel estimation with an untrained deep neural network," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 3, pp. 2079–2090, Jan. 2020.

[27] Y. Zhang, M. Alrabeiah, and A. Alkhateeb, "Deep learning for massive MIMO with 1-bit ADCs: When more antennas need fewer pilots," *IEEE Wireless Commun. Lett.*, vol. 9, no. 8, pp. 1273–1277, Aug. 2020.

[28] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "Deep learning-based channel estimation for beamspace mmwave massive MIMO systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 852–855, May 2018.

[29] M. Alrabeiah and A. Alkhateeb, "Deep learning for TDD and FDD massive MIMO: Mapping channels in space and frequency," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, Mar. 2019, pp. 1465–1470.

[30] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989, ISSN: 0893-6080.

[31] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," in *Proc. of Information Theory and Applications Workshop (ITA)*, San Diego, CA, Feb. 2019, pp. 1–8.

[32] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 831–846, 2014.

[33] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive MIMO networks: Spectral, energy, and hardware efficiency," *Foundations and Trends® in Signal Processing*, vol. 11, no. 3-4, pp. 154–655, Dec. 2017, ISSN: 1932-8346.

[34] I. P. Roberts, A. Chopra, T. Novlan, S. Vishwanath, and J. G. Andrews, "Spatial and statistical modeling of multi-panel millimeter wave self-interference," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 9, pp. 2780–2795, Jul. 2023.

[35] I. P. Roberts, A. Chopra, T. Novlan, S. Vishwanath, and J. G. Andrews, "Beamformed self-interference measurements at 28 GHz: Spatial insights and angular spread," *IEEE Trans. Wirel. Commun.*, vol. 21, no. 11, pp. 9744–9760, Jun. 2022.

[36] Z. Xiao, P. Xia, and X.-G. Xia, "Full-duplex millimeter-wave communication," *IEEE Wireless Commun.*, vol. 24, no. 6, pp. 136–143, Dec. 2017.

[37] A. Assalini, E. Dall'Anese, and S. Pupolin, "Linear MMSE MIMO channel estimation with imperfect channel covariance information," in *2009 IEEE International Conference on Communications*, Dresden, Germany, Aug. 2009, pp. 1–5.

[38] L. Sanguinetti, E. Björnson, and J. Hoydis, "Toward massive MIMO 2.0: Understanding spatial correlation, interference suppression, and pilot contamination," *IEEE Trans. Wirel. Commun.*, vol. 68, no. 1, pp. 232–257, Oct. 2020.

[39] S. Sun, T. S. Rappaport, S. Rangan, *et al.*, "Propagation path loss models for 5G urban micro- and macro-cellular scenarios," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, Nanjing, China, Jul. 2016, pp. 1–6.

[40] L. V. Nguyen, D. H. N. Nguyen, and A. L. Swindlehurst, "Deep learning for estimation and pilot signal design in few-bit massive mimo systems," *IEEE Transactions on Wireless Communications*, vol. 22, no. 1, pp. 379–392, 2023.

[41] Remcom, "Wireless InSite," `http://www.remcom.com/wireless-insite`.

PAPER B

Neural CSI Compression Fine-Tuning: Taming the Communication Cost of Model Updates

Mehdi Sattari, Deniz Gündüz, Tommy Svensson

Submitted to *IEEE Transactions on Machine Learning in Communications and Networking*, Feb, 2026

The layout has been revised.

Abstract

Efficient channel state information (CSI) compression is essential in frequency division duplexing (FDD) massive multiple-input multiple-output (MIMO) systems due to the substantial feedback overhead. Recently, deep learning-based compression techniques have demonstrated superior performance for CSI feedback. However, their performance often degrades under distribution shifts across wireless environments, largely due to limited generalization capability. To address this challenge, we consider a full-model fine-tuning scheme, in which both the encoder and decoder are jointly updated using a small number of recent CSI samples from the target environment. A key challenge in this setting is the transmission of updated decoder parameters to the receiver, which introduces additional communication overhead. To mitigate this bottleneck, we explicitly incorporate the bit rate of model updates into the fine-tuning objective and entropy-code the model updates jointly with the compressed CSI. Furthermore, we employ a structured prior that promotes sparse and selective parameter updates, thereby significantly reducing the model-update communication cost. Simulation results across multiple CSI datasets demonstrate that full-model fine-tuning substantially improves the rate–distortion performance of neural CSI compression, despite the additional cost of model updates. We further analyze the impact of the evaluation horizon, the quantization resolution of model updates, and the size of the target-domain dataset on the overall feedback efficiency.

B.1 Introduction

Masssive multiple-input multiple-output (MIMO) is one of the key enabling technologies for 5G and beyond. By equipping the base station (BS) with massive antenna arrays, a large number of users can be served simultaneously through high-resolution beamforming techniques, resulting in remarkable spectral efficiency. To enable massive MIMO technology, channel state

information (CSI) must be available at both the BS and user sides. Various channel estimation techniques can be employed to estimate CSI by observing pilot signals. In time-division duplexing (TDD), only downlink/uplink channels need to be estimated, as the other channel can be derived thanks to reciprocity. However, in frequency-division duplexing (FDD), reciprocity does not hold, so the channels in both directions need to be estimated and fed back[1], [2].

To enable FDD transmission for massive MIMO systems, CSI compression techniques are critical to circumvent excessive feedback overhead. Various CSI compression techniques have been studied in the literature by employing compressed sensing [3], [4], vector quantization [5], [6], and more recently, deep learning tools [7]–[15]. Due to strong spatial correlation, the CSI matrix can exhibit sparsity in certain domains, and compressed sensing methodologies can be applied to efficiently compress the large CSI matrix [3], [4]. However, these algorithms often struggle to find the best basis to project the CSI matrix to lower dimensions. Furthermore, compressed sensing-based algorithms are iterative and time-consuming, making them infeasible for deployment in massive MIMO channels. Similarly, in vector quantization for CSI compression [5], [6], the overhead scales linearly with the channel dimension, rendering it impractical for massive MIMO systems.

Recent advances in data-driven compression approaches, driven by the progress in neural network (NN) architectures have gained substantial interest in all data modalities [16]. Neural data compression is a machine learning technique that performs the compression task using deep neural networks (DNNs). Current research in neural compression is largely driven by the development of deep generative models such as generative adversarial networks (GANs) [17], variational autoencoders (VAEs) [18], normalizing flows [19], and autoregressive models [20] for image and video compression. These models have proven effective in capturing complex data distributions, which is crucial for achieving high compression rates while maintaining data fidelity.

In the wireless communication domain, the first neural CSI compressor was introduced in[7], where the authors used a simple convolutional neural network (CNN) auto-encoder architecture for dimension reduction of a massive MIMO channel matrix. Later, numerous network architectures and machine learning techniques have been proposed to further improve the CSI compression efficiency [8]–[11]. While most works formulated the problem in terms of the

dimensions of the reduced CSI matrix, the feedback channels must ultimately convey the CSI matrix in bits, necessitating additional quantization in the latent space. In [14] and [21], the authors formulated the problem within the rate-distortion (RD) framework and further enhanced the performance of the neural CSI compressor by incorporating learned entropy encoding and decoding blocks. They employed a weighted RD loss function to jointly minimize the mean squared error (MSE) and the average bit-rate overhead, explicitly quantifying the communication cost of compression in bits per CSI dimension under a variable-rate compression scheme.

Machine learning models often perform remarkably well on test sets with a distribution similar to their training data but can fail catastrophically in deployment when the data distribution shifts. Such shifts can arise from various factors, including temporal changes, domain variations, or sampling biases, and pose significant challenges in many real-world applications [22]. Several approaches, such as transfer learning [23], data augmentation [24], and domain adaptation [25], can be employed to enhance the robustness of machine learning models against distribution shift and improve their generalization capabilities. In transfer learning, models trained on generic datasets are fine-tuned for the target domain. Data augmentation improves generalization by artificially generating diverse training samples. Domain adaptation, on the other hand, aligns source and target domain distributions through techniques such as instance re-weighting or feature alignment.

Distribution shift is a common phenomenon in wireless networks. The statistics of the underlying wireless channel change due to various factors such as user mobility, environmental changes, or variations in interference from other devices. When the channel distribution deviates from initial assumptions or models, the efficiency of algorithms for tasks such as channel estimation [26], signal detection [27], etc can degrade. Consequently, maintaining robust communication requires adaptive strategies that can adjust to these shifts. In the context of CSI compression for FDD massive MIMO systems, most models in the literature are trained and tested using data from the same environment, e.g., within a specific macro-cell coverage area. However, this would require users to either store or download [28] new models for every new cell they enter, making these approaches infeasible in practice. Therefore, it is crucial to develop techniques that can overcome the distribution shift problem in CSI compression.

Several works have studied the effect of distribution shift in the CSI compression problem [29]–[35]. In [29], a deep transfer learning method is used to handle the distribution shift and a model-agnostic meta-learning algorithm is proposed to reduce the number of CSI samples. A lightweight translation model and data augmentation method based on domain knowledge are introduced in [30]. Specifically, to adapt to new channel conditions, the authors propose an efficient scenario-adaptive CSI feedback architecture, "CSI-TransNet," and an efficient deep unfolding-based CSI compression network, "SPTM2-ISTANet+." A single-encoder-to-multiple-decoders (S-to-M) approach is presented in [31], where the authors use multi-task learning to integrate multiple independent autoencoders into a unified architecture featuring a shared encoder and several task-specific decoders.

The authors in [32] proposed online learning schemes based on a vanilla autoencoder architecture, exploring various adaptation strategies. In [33], elastic weight consolidation is incorporated to alleviate catastrophic forgetting in NNs. A federated edge learning (FEEL)-based training framework for massive MIMO CSI feedback is introduced in [34] to address privacy concerns arising from transmitting raw CSI datasets during centralized training, and a personalization strategy is further developed to enhance feedback performance. In [35], a gossip learning strategy is adopted, and two data augmentation techniques—random erasing and random phase shift—are employed to improve generalization capability.

However, we highlight that the communication overhead associated with model updates is neglected in all the aforementioned prior works. This represents a significant gap in the literature, as typical DNNs can contain millions of parameters, and transmitting model updates can result in massive feedback overhead. In [34], low-resolution quantization (e.g., 2-bit) is applied to the model updates; however, the resulting update bit-rate is not accounted for in the evaluation. To elaborate, the decoder network in [34] includes approximately 4 million trainable parameters. Even with an aggressive 2-bit quantization, the resulting communication overhead would be around 8 Mbits, substantially higher than that is required to transmit the compressed CSI matrix. More precisely, a compressed CSI matrix of size 64×64, using a modest compression ratio of 16, would only require 4 kbits and 16 kbits for 2-bit and 32-bit quantization, respectively. Comparing the communication cost for compressed CSI and model updates illustrates that transmitting model up-

dates can introduce a prohibitive feedback overhead for FDD massive MIMO systems.

To bridge this gap, in this paper, we explicitly account for the communication overhead of model update transmission. We investigate how to efficiently transmit both the model updates and the latent compressed CSI to obtain the best overall RD trade-off. The key differences and contributions of this work are summarized as follows:

- Prior works addressing distribution shifts in neural CSI compression primarily focus on vanilla autoencoder-based schemes and overlook bit-level CSI compression as well as the communication cost of transmitting model updates to the decoder [29]–[35]. In contrast, by incorporating lossless entropy coding for both the compressed CSI latents and the model updates, we explicitly quantify the RD trade-off in neural CSI compression while accounting for the total communication cost, including both CSI feedback and model-update signaling.

- To enable efficient entropy coding of model updates, we model the update distribution using a spike-and-slab prior, formulated as a mixture of narrow (spike) and wide (slab) Gaussian components. By assigning higher probability mass to the spike component, the proposed prior promotes sparse and selective parameter updates, allowing only the most impactful weights to be modified. In addition, we incorporate the bit rate of model updates as a regularization term in the RD loss. Simulation results demonstrate that the combined use of spike-and-slab modeling and update-rate regularization is essential for significantly reducing model-update communication cost while improving overall RD performance.

- We evaluate the proposed framework across diverse CSI datasets, including QuaDriGa [36], 3GPP clustered delay line (CDL) [37], and DeepMIMO [38], covering both site-specific and stochastic channel models and a wide range of wireless environments. We systematically investigate the impact of the amortized model-update rate over the evaluation horizon, as well as the effects of target-domain dataset size and model-update quantization. Our results reveal a fundamental trade-off between the amortized model-update rate and distortion in dynamic wireless environments, and demonstrate that robust RD performance

can be achieved using a few hundred CSI samples, with finer model-update quantization further improving performance.

The remainder of this paper is organized as follows. Section B.2 introduces the backbone neural CSI compressor for an FDD massive MIMO system and evaluates its performance in the absence of distribution shifts. Section B.3 describes the considered CSI datasets and the proposed neural model fine-tuning framework. Section B.4 presents numerical results in terms of the RD trade-off, along with a complexity analysis. Finally, Section B.5 concludes the paper.

Throughout this paper, the following notation is adopted. Bold uppercase letters denote matrices, and bold lowercase letters denote vectors. The transpose operation is represented by the superscript $(\cdot)^T$. The expectation operator is denoted by $\mathbb{E}$, and $\|\cdot\|$ represents the norm. The symbols $\lfloor\cdot\rceil$, $\lfloor\cdot\rfloor$, and $\lceil\cdot\rceil$ denote the rounding, floor, and ceiling operations, respectively.

B.2 FDD massive MIMO and Backbone Neural CSI Compressor

B.2.1 FDD Massive MIMO

We consider an FDD transmission scenario in which a massive MIMO BS equipped with N_t antennas serves multiple single-antenna users. Downlink transmission is performed using orthogonal frequency division multiplexing (OFDM) over N_c subcarriers. For each subcarrier $m \in \{1, \ldots, N_c\}$, let $\mathbf{h}_m \in \mathbb{C}^{N_t}$ denote the downlink channel vector between the BS and a given user, $\mathbf{w}_m \in \mathbb{C}^{N_t}$ the corresponding transmit precoding vector, $x_m \in \mathbb{C}$ the transmitted data symbol, and $\mathbf{n}_m \in \mathbb{C}$ the additive noise. The received signal at the user on subcarrier m is given by

$$\mathbf{y}_m = \mathbf{h}_m^T \mathbf{w}_m x_m + \mathbf{n}_m. \tag{B.1}$$

The downlink CSI across all subcarriers in the spatial–frequency domain is represented by the matrix

$$\mathbf{H} = \left[\mathbf{h}_1, \ldots, \mathbf{h}_{N_c}\right] \in \mathbb{C}^{N_t \times N_c}. \tag{B.2}$$

In massive MIMO systems, the channel matrix is high-dimensional, making full CSI feedback prohibitively expensive in terms of uplink bandwidth. To alleviate this bottleneck, CSI compression is commonly employed, whereby the user compresses the CSI matrix and the BS subsequently reconstructs the channel matrix. The objective of CSI compression in FDD massive MIMO systems is to minimize the reconstruction distortion while satisfying a prescribed feedback bit-rate constraint from the user to the BS.

B.2.2 Backbone Neural CSI Compressor

We consider an autoencoder-based NN architecture that explicitly targets the RD trade-off in CSI compression. Quantization, entropy encoding, and entropy decoding are incorporated into the training procedure, enabling optimization of both the feedback rate and the reconstruction distortion of the neural CSI compressor. The proposed compressor is denoted by

$$c = (f_\phi, g_\theta, \gamma_\theta), \tag{B.3}$$

where $f_\phi : \mathcal{C}^{N_t \times N_c} \to \mathcal{Z}$ represents the feature encoder parameterized by ϕ, $g_\theta : \mathcal{Z} \to \mathcal{C}^{N_t \times N_c}$ denotes the feature decoder, and γ_θ corresponds to the entropy coder, jointly parameterized by θ [16]. For notational simplicity, the same symbol θ is used to denote the parameter set of both the feature decoder and the entropy coder.

The *encoder* maps each CSI matrix $\mathbf{H} \in \mathbb{C}^{N_t \times N_c}$ to a latent representation $\mathbf{Z} \in \mathcal{Z}$. The latent representation is then quantized and transmitted losslessly to the receiver using a variable-length entropy code. Let $\bar{\mathbf{Z}} = Q(\mathbf{Z})$ denote the quantized latent representation, where $Q(\cdot)$ represents the quantization operation. The entropy encoder converts the quantized latent values into a bitstream using a lossless coding scheme, given by

$$b_z = \gamma_\theta(\bar{\mathbf{Z}}; p_\theta), \tag{B.4}$$

where p_θ denotes the prior probability model of the latent representation.

At the *decoder*, the entropy decoder is first applied to recover the quantized latent representation from the received bitstream as $\bar{\mathbf{Z}} = \gamma_\theta^{-1}(b_z; p_\theta)$. The de-quantization operation is then applied to obtain the continuous latent representation, denoted by $\hat{\mathbf{Z}} = Q^{-1}(\bar{\mathbf{Z}})$. We emphasize that $Q^{-1}(\cdot)$ does

not represent a true inverse of the quantization function $Q(\cdot)$ and is generally non-bijective in lossy compression. Both the encoder and decoder employ a shared entropy coder γ_θ with prior distribution p_θ to ensure the lossless transmission of the quantized latent representation $\bar{\mathbf{Z}}$. Finally, the feature decoder reconstructs the CSI matrix from the de-quantized latent representation as

$$\hat{\mathbf{H}} = g_\theta\left(\hat{\mathbf{Z}}\right). \tag{B.5}$$

Let the distortion function be denoted by $\rho : \mathbf{H} \times \hat{\mathbf{H}} \to [0, \infty)$, which quantifies the reconstruction error between the ground-truth CSI matrix and its estimate. In practice, the distortion is typically measured using the squared error, i.e., $\rho(\mathbf{H}, \hat{\mathbf{H}}) \propto \|\mathbf{H} - \hat{\mathbf{H}}\|^2$. Under a lossy compression framework, the resulting distortion of a compressor c is defined as

$$D(c) = \mathbb{E}\Big[\rho(\mathbf{H}, \hat{\mathbf{H}})\Big]. \tag{B.6}$$

The expectation is taken over the random realization of the channel matrix $\mathbf{H}$, and without loss of optimality, we restrict our attention to deterministic codes [39]. Let the bit length associated with encoding a CSI matrix $\mathbf{H}$ be defined as

$$l(\mathbf{H}) := \big|\gamma_\theta(Q(f_\phi(\mathbf{H})))\big|, \tag{B.7}$$

and the corresponding average bit rate be defined as

$$R(c) = \mathbb{E}[l(\mathbf{H})], \tag{B.8}$$

which captures the expected number of bits required to encode the CSI matrix $\mathbf{H}$. In practice, this rate can be approximated by the entropy of the quantized latent representation, resulting in

$$R(c) = \mathbb{E}\big[-\log_2 p_\theta(\bar{\mathbf{Z}})\big]. \tag{B.9}$$

The objective is to jointly minimize the distortion and the rate with respect to the encoder, decoder, and latent representations. According to RD theory, the fundamental limits of lossy compression are characterized by a conditional

distribution $p_{\hat{\mathbf{H}}|\mathbf{H}}$. The RD function is given by

$$R(D) = \inf_{p_{\hat{\mathbf{H}}|\mathbf{H}}:\, \mathbb{E}\left[\rho(\mathbf{H},\hat{\mathbf{H}})\right] \le D} I\left(\mathbf{H};\hat{\mathbf{H}}\right), \tag{B.10}$$

where $I(\mathbf{H};\hat{\mathbf{H}})$ denotes the mutual information between the CSI matrix $\mathbf{H}$ and its reconstruction $\hat{\mathbf{H}}$. In theory, this optimal rate can be achieved using vector quantization by jointly compressing many independent realizations of the channel matrix. However, such schemes become intractable for high-dimensional CSI and are incompatible with practical systems, where each CSI matrix must be compressed individually. To address this limitation, we adopt an *operational RD* formulation and denote the set of admissible codecs by $\mathcal{C}$:

$$R_{\mathrm{O}}(D) = \inf_{c\in\mathcal{C}:\, \mathbb{E}\left[\rho(\mathbf{H},\hat{\mathbf{H}})\right] \le D} \mathbb{E}[l(\mathbf{H})], \tag{B.11}$$

which characterizes the *operational RD function* over the class of admissible codecs $\mathcal{C}$. This constrained optimization problem can be equivalently addressed via a Lagrangian formulation,

$$L(\lambda, c) = R(c) + \lambda D(c) = \mathbb{E}[l(\mathbf{H})] + \lambda \mathbb{E}\Big[\rho(\mathbf{H},\hat{\mathbf{H}})\Big]. \tag{B.12}$$

For any $\lambda \ge 0$, minimizing the Lagrangian objective over $\mathcal{C}$ yields an optimal codec c^*. The parameter λ governs the trade-off between rate and distortion. Larger values of λ prioritize distortion minimization, resulting in operation at a higher bit-rate regime, whereas smaller values favor stronger compression at the expense of reconstruction accuracy. By adopting the MSE distortion measure, the resulting RD loss is expressed as

$$L_{\mathrm{RD}}(\phi,\theta) = \mathbb{E}[-\log_2 p_\theta(Q(f_\phi(\mathbf{H})))] + \lambda\, \mathbb{E}\Big[\big\| g_\theta\big(Q^{-1}(Q(f_\phi(\mathbf{H})))\big) - \mathbf{H}\big\|^2\Big]. \tag{B.13}$$

Fig. 1 illustrates the encoder–decoder architecture of the proposed neural CSI compressor, where the input consists of the real and imaginary components of the spatial–frequency channel matrix. The encoder function f_ϕ is composed of a sequence of convolutional layers with nonlinear activation functions that map the high-dimensional massive MIMO CSI matrix $\mathbf{H}$ to a lower-dimensional latent representation. Specifically, the first two layers em-

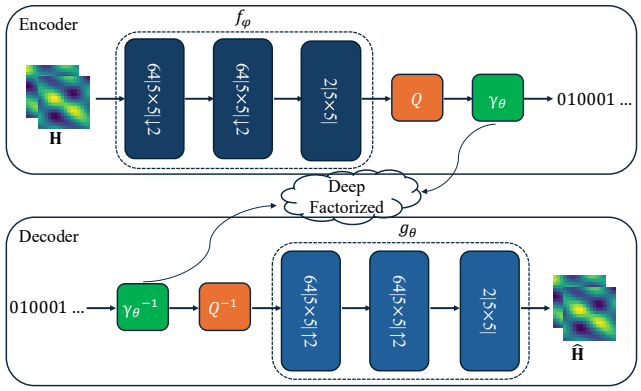


Figure 1: The neural CSI compressor architecture.

ploy convolutional filters with 64 kernels followed by ReLU activation functions, while the final layer uses 2 convolutional kernels with a linear activation. Max-pooling operations are applied to progressively reduce the spatial dimensions of the input CSI tensor.

After obtaining the low-dimensional latent representation, quantization and entropy coding are applied to convert the latent space into a bit sequence. Specifically, a uniform scalar quantizer with unit step size is employed to map continuous latent values to discrete symbols. Leveraging the latent probability distribution learned during training, a lossless entropy coding scheme, such as range coding or context-adaptive binary arithmetic coding (CABAC), is then used to encode the quantized latent representation, resulting in a variable-length bitstream.

Incorporating quantization into the compression framework introduces a challenge for gradient-based optimization, as the quantization operation has zero derivative almost everywhere. To address this issue, a common relaxation is adopted whereby quantization is approximated by adding independent and identically distributed uniform noise, $\Delta\mathbf{Z} \sim \mathcal{U}(0,1)$, to the latent representation, yielding $\tilde{\mathbf{Z}} = \mathbf{Z} + \Delta\mathbf{Z}$. This relaxation provides a differentiable surrogate and yields an upper bound on the expected number of bits required to encode the latent space. During inference, the true quantization operation is applied, and the rate is computed using the actual quantized latent values without additive noise.

On the decoder side, the entropy-coded bitstream is first decoded to recover the quantized latent representation, which is then mapped to a low-dimensional channel representation by the decoding network. The entropy encoder and decoder share the same latent prior distribution. In our architecture, we adopt a fully factorized prior implemented via the `DeepFactorized` model [40], which leverages deep neural networks to learn the latent probability distribution under factorization assumptions.

The decoder network, represented by the function g_θ in Fig. 1, consists of a stack of convolutional layers with a kernel size of 5×5, each followed by a ReLU activation function. To reconstruct the high-dimensional CSI matrix from the low-dimensional entropy-decoded representation, an upsampling operation is applied to progressively restore the spatial resolution.

B.3 Neural CSI Compression Fine-Tuning

A fundamental challenge in learning-based methods is handling distribution shifts. When model capacity is limited or training data is insufficient, such shifts can cause significant performance degradation, thereby limiting real-world effectiveness. This issue is particularly critical in applications such as neural CSI compression for wireless communication systems, where the operating environment is inherently dynamic.

B.3.1 Evaluation of The Backbone Neural CSI Compressor

To assess the performance of the trained neural CSI compressor, we employ the DeepMIMO dataset, which constructs MIMO channel realizations using ray-tracing simulations generated by Remcom Wireless InSite [41]. Specifically, the neural CSI compressor is trained using the static outdoor scenario referred to as O1 at a carrier frequency of 28 GHz. The detailed parameter configuration for this dataset is summarized in Table 1.

A total of 11,920 CSI realizations are generated and partitioned into training, validation, and test sets using a 40%, 40%, and 20% split, respectively. Training is conducted using the RD loss function defined in (B.13), with the objective of jointly optimizing the encoder and decoder parameters as well as the factorized probability model used for entropy coding. The backbone neural CSI compressor is implemented in Python 3.9 using the TensorFlow

Table 1: DeepMIMO dataset parameters

DeepMIMO parameters	**Value**
Scenario	O1
Center frequency	28 GHz
Number of paths	10
Active users	from row 1100 to 2200
Active BS number	BS 5
Bandwidth	50 MHz
Number of OFDM subcarriers	64
BS antennas	$N_x = 1$, $N_y = 64$, $N_z = 1$
UE antennas	$N_x = 1$, $N_y = 1$, $N_z = 1$

framework. Network parameters are optimized using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 32 over 200 training epochs.

We assess the performance of the neural CSI compressor in terms of the RD trade-off achieved, where the distortion is measured as the normalized mean squared error (NMSE), defined as

$$\text{NMSE} \triangleq \mathbb{E}\left[\frac{\|\mathbf{H} - \hat{\mathbf{H}}\|^2}{\|\mathbf{H}\|^2}\right], \tag{B.14}$$

where $\mathbf{H}$ and $\hat{\mathbf{H}}$ denote the true and reconstructed channel matrices, respectively. The bit rate of the entropy-coded latent representation is estimated using (B.9) and normalized by the CSI dimensionality, i.e., 64×64.

The RD curves in Fig. 2 illustrate the performance of the trained neural CSI compressor for different values of λ, where each operating point corresponds to $\lambda \in 5 \times 10^4, 10^5, 5 \times 10^5, 10^6$. The results confirm that the proposed neural CSI compressor achieves strong RD performance when evaluated on previously unseen CSI realizations drawn from the same propagation environment.

B.3.2 Evaluation Under Distribution Shifts

To analyze the impact of distribution shifts, we evaluate the trained neural CSI compressor in propagation environments that differ significantly from the training domain. Specifically, we consider multiple distinct datasets, namely QuaDRiGa [36], 3GPP CDL [37], and DeepMIMO [38] (under a different sce-

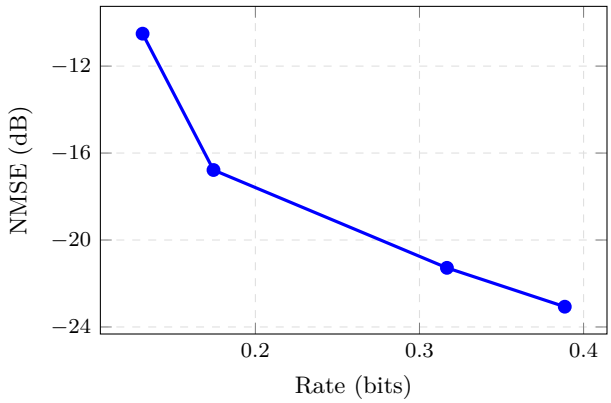


Figure 2: RD performance of the backbone neural CSI compressor.

nario), to assess the resulting RD performance. In the following, we describe the key parameter configurations and channel characteristics of these datasets.

QuaDRiGa Dataset QuaDRiGa is a geometry-based stochastic spatial channel model that captures spatial consistency and realistic multipath evolution in three-dimensional propagation environments. Using the QuaDRiGa channel generator, we simulate an urban massive MIMO deployment with a BS equipped with 64 antenna elements and operating over 64 OFDM subcarriers. The BS is placed on a rooftop, while a mobile user traverses a linear track of length 350 m. The underlying three-dimensional propagation environment is based on the Madrid grid developed within the METIS project, enabling realistic modeling of spatial channel evolution and path dynamics along the user trajectory. The simulation layout is illustrated in Fig. 3, and the corresponding dataset parameters are summarized in Table 2.

3GPP CDL Dataset To further evaluate the performance of the proposed neural CSI compression framework under controlled and standardized channel conditions, we generate a dataset based on the 3GPP CDL channel model defined in TR 38.901. Specifically, we consider the CDL-B delay profile under an urban macrocell (UMa) scenario with a user velocity of 30 km/h. The CDL model characterizes time-varying multipath propagation through a finite set of clusters with fixed angular statistics, while temporal channel evolution is

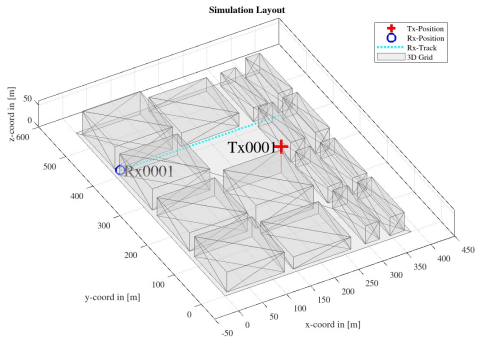


Figure 3: Simulation layout of QuaDRiGa dataset [36].

Table 2: Simulation parameters of the QuaDRiGa dataset.

Parameter	**Value**
Carrier frequency	3.7 GHz
Bandwidth	50 MHz
Number of OFDM subcarriers	64
Number of BS antennas	64
Number of UE antennas	1
Initial user position	(15, 415, 1.2)
BS position	(267, 267, 60)
BS orientation	$\pi/2$ (facing north)
Spatial sampling density	29.6 samples/m

primarily governed by Doppler effects induced by user mobility. In our setup, the BS is equipped with 64 transmit antennas, and the user employs a single receive antenna. The system operates at a carrier frequency of 28 GHz using an OFDM numerology with a subcarrier spacing of 30 kHz. The detailed simulation parameters for this dataset are summarized in Table 3.

DeepMIMO Dataset For the DeepMIMO dataset, we consider another site-specific, ray-tracing-based channel model referred to as O2 Dynamic. In this setting, CSI realizations are generated across a sequence of discrete scenes with a sampling interval of 100 ms. The top view of the scenario, shown in

Table 3: Simulation parameters for the 3GPP CDL dataset.

Parameter	Value
Delay profile	CDL-B
Carrier frequency	28 GHz
User velocity	30 km/h
Delay spread	200 ns
Number of BS antennas	64
Number of UE antennas	1
OFDM subcarrier spacing	30 kHz
Number of resource blocks	25
Number of selected subcarriers	64
CSI sample stride	5 OFDM symbols
Number of CSI samples	5,000

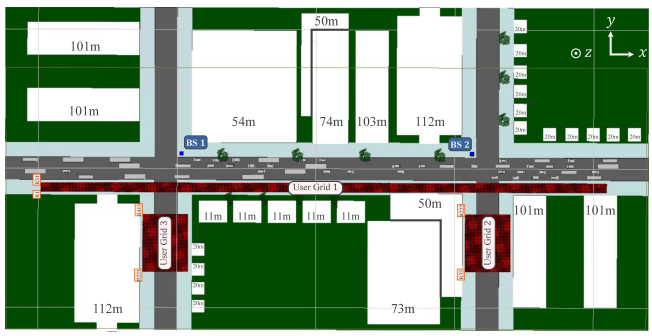


Figure 4: The top view of the O2 Dynamic scenario [38].

Fig. 4, consists of three streets and two intersections in an urban environment. Across the 1,000 captured scenes, vehicular users change positions while the underlying propagation geometry remains fixed, resulting in quasi-stationary channel statistics within each scene. The dataset includes two BSs and approximately 115,000 candidate user locations, enabling large-scale and diverse CSI generation under realistic site-specific conditions. The detailed simulation parameters for this scenario are summarized in Table 4.

To evaluate the generalization capability of the trained neural CSI compressor in previously unseen environments, we assess its RD performance in Fig. 5. The model is tested on the aforementioned datasets without any fine-tuning. As shown in the figure, the trained compressor exhibits a severe performance

Table 4: Simulation parameters of the DeepMIMO dataset (O2 Dynamic scenario).

Parameter	**Value**
Scenario	O2 Dynamic
Carrier frequency	3.5 GHz
Bandwidth	50 MHz
Number of OFDM subcarriers	64
Number of multipath components	10
Active user rows	1–31
Scene index	1
Active BS	BS 1
BS antenna configuration	$N_x = 1,\ N_y = 64,\ N_z = 1$
UE antenna configuration	$N_x = 1,\ N_y = 1,\ N_z = 1$

degradation under distribution shifts, failing to cope with the new channel statistics. These results clearly demonstrate the limitations of direct model reuse and underscore the necessity of an effective model fine-tuning strategy for neural CSI compression in massive MIMO systems.

B.3.3 Neural Model Fine-Tuning

A trained model can be fine-tuned to new channel statistics by exposing it to data samples from the target environment. In the context of neural CSI compression, model fine-tuning corresponds to updating the backbone network parameters, namely ϕ_0 and θ_0 for the encoder and decoder, respectively, which were initially trained on a generic dataset. Fine-tuning the backbone neural CSI compressor using CSI samples from a new domain yields updated encoder and decoder parameters, denoted by ϕ and θ, respectively. Since the encoder is assumed to be fixed at the transmitter, only the updated decoder parameters need to be conveyed in the bitstream.

We consider a full-model fine-tuning, where both the encoder and decoder networks are fine-tuned while explicitly accounting for the additional communication overhead associated with conveying decoder model updates from the transmitter to the receiver. Specifically, the decoder model updates, defined as $\delta \triangleq \theta - \theta_0$, are encoded jointly with the latent representation z. To encode the decoder model update vector, it is first quantized. Unlike the unit-bin quantization applied to the latent space, a higher-resolution quantization is

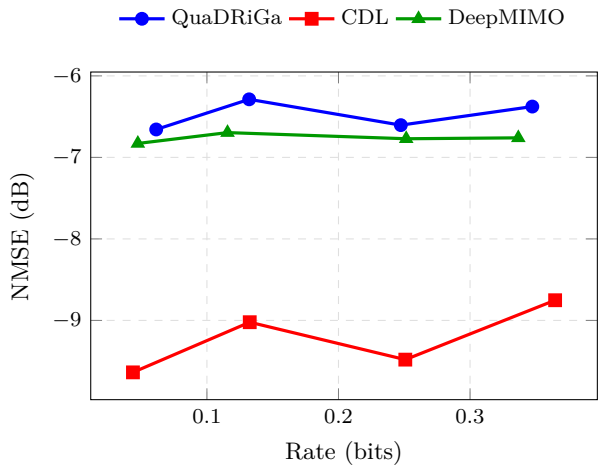


Figure 5: Evaluation of the backbone neural CSI compressor using QuaDRiGa, CDL, and DeepMIMO datasets.

employed for the model updates, since their values typically vary only slightly, depending on the learning rate. In particular, N equispaced bins of width t are used, and the quantization function for the model updates is defined as follows [42]:

$$\bar{\delta} = Q_t(\delta) = \text{clip}\left(\left\lfloor \frac{\delta}{t} \right\rceil t, -\frac{(N-1)t}{2}, \frac{(N-1)t}{2}\right), \tag{B.15}$$

where clipping and rounding operations are defined as:

$$\text{clip}(x, x_{\min}, x_{\max}) = \begin{cases} x_{\min}, & \text{if } x < x_{\min}, \\ x, & \text{if } x_{\min} \leq x \leq x_{\max}, \\ x_{\max}, & \text{if } x > x_{\max}, \end{cases} \tag{B.16}$$

$$\lfloor x \rceil = \begin{cases} \lfloor x \rfloor, & \text{if } x - \lfloor x \rfloor < 0.5, \\ \lceil x \rceil, & \text{if } x - \lfloor x \rfloor \geq 0.5. \end{cases} \tag{B.17}$$

Since both rounding and clipping are non-differentiable operations, they obstruct gradient-based training of the neural CSI compressor. To address this

issue, a commonly used technique known as the straight-through estimator (STE) is employed [43], where the gradient of the quantization function is approximated as $\partial Q_t(\delta)/\partial\delta = 1$. Here, the quantization bin width is controlled by t, and N denotes the number of quantization bins.

Let $\bar{\delta} = Q_t(\delta)$ denote the quantized model update. The discrete model prior $p[\bar{\delta}]$ is obtained by pushing forward the continuous prior $p(\delta)$ through the quantization function Q_t, yielding

$$\begin{aligned} p[\bar{\delta}] &= \int_{Q_t^{-1}(\bar{\delta})} p(\delta)\, d\delta \\ &= \int_{\bar{\delta}-t/2}^{\bar{\delta}+t/2} p(\delta)\, d\delta \\ &= P(\delta < \bar{\delta} + t/2) - P(\delta < \bar{\delta} - t/2). \end{aligned} \tag{B.18}$$

Thus, $p[\bar{\delta}]$ represents the probability mass assigned to the quantization bin centered at $\bar{\delta}$. It is computed as the difference between the cumulative distribution function (CDF) values of $p(\delta)$ evaluated at the bin boundaries.

After quantization, an appropriate model prior $p(\delta)$ must be selected to enable efficient entropy coding of the discrete model updates. Various distributions can be used to model this prior, such as a zero-mean Gaussian, i.e., $p(\delta) = \mathcal{N}(\mathbf{0}, \sigma\mathbf{I})$. However, a key limitation of a Gaussian prior is its relatively high coding cost for zero updates. As a result, even when no effective update is applied, encoding such updates can still incur a non-negligible bit rate.

To address this inefficiency, a spike-and-slab prior is adopted [44], defined as

$$p(\delta) = \frac{p_{\text{slab}}(\delta) + \alpha\, p_{\text{spike}}(\delta)}{1+\alpha}, \tag{B.19}$$

where

$$p_{\text{slab}}(\delta) = \mathcal{N}(\delta \mid 0, \sigma^2 \boldsymbol{I}) \tag{B.20}$$

denotes the slab component, and

$$p_{\text{spike}}(\delta) = \mathcal{N}(\delta \mid 0, (t/6)^2 \boldsymbol{I}) \tag{B.21}$$

denotes the spike component.

The resulting prior $p(\delta)$ is a mixture of Gaussian distributions, where $\alpha \in \mathbb{R}^+$ controls the relative weight of the spike component. The parameters t

Algorithm B.1 Full-Model Fine-Tuning — Encoding

Input: Global model parameters $\{\theta_0, \phi_0\}$ trained on a generic dataset; batch size B; CSI samples from a new environment $\mathcal{H} = \{\mathcal{H}_\mathrm{T}, \mathcal{H}_\mathrm{E}\}$.
Output: Compressed bitstream $\mathbf{b} = (b_{\bar{\delta}}, b_z)$.

Initialize $\phi \leftarrow \phi_0$, $\theta \leftarrow \theta_0$
for epoch = 1 to num_epochs **do**
  for each mini-batch $\mathcal{B} \subset \mathcal{H}_\mathrm{T}$ with $|\mathcal{B}| = B$ **do**
    Load data $\mathbf{H} \in \mathcal{B}$
    Compute $\delta \leftarrow \theta - \theta_0$, quantize $\bar{\delta} \leftarrow Q_t(\delta)$, and set $\bar{\theta} \leftarrow \theta_0 + \bar{\delta}$
    Encode features $\mathbf{Z} \leftarrow f_\phi(\mathbf{H})$ and quantize $\tilde{\mathbf{Z}} \leftarrow \mathbf{Z} + \Delta\mathbf{Z}$
    Decode $\hat{\mathbf{H}} \leftarrow g_\theta(\tilde{\mathbf{Z}})$
    Compute loss $L_\mathrm{RDM}(\phi, \theta)$
    Backpropagate with STE and update ϕ, θ using gradients $\nabla_{\phi,\theta} L_\mathrm{RDM}(\phi, \theta)$
  end for
end for
Fine-tuned parameters: $\phi^* \leftarrow \phi$, $\theta^* \leftarrow \theta$
For evaluation set $\mathcal{H}_\mathrm{E}$: compute $\bar{\mathbf{Z}} \leftarrow Q(f_{\phi^*}(\mathbf{H}))$
Quantize tuned parameters: $\bar{\delta} \leftarrow Q_t(\theta^* - \theta_0)$, $\bar{\theta} \leftarrow \theta_0 + \bar{\delta}$
Entropy encode: $b_{\bar{\delta}} \leftarrow \gamma(\bar{\delta}; p(\bar{\delta}))$, $b_z \leftarrow \gamma_{\bar{\theta}}(\bar{\mathbf{Z}}; p_{\bar{\theta}})$
Return $\mathbf{b} = (b_{\bar{\delta}}, b_z)$

and σ, with $\sigma \gg t/6$, correspond to the standard deviations of the spike and slab distributions, respectively. By setting the standard deviation of the spike component to $t/6$, approximately 99.7% of its probability mass lies within the central quantization bin after quantization. Consequently, selecting a sufficiently large value of α significantly reduces the bit-rate cost associated with zero updates and encourages the model to learn only the most informative parameter updates. This prior effectively promotes sparsity in the model updates while maintaining flexibility for larger, informative deviations.

The bit-rate cost associated with the quantized model updates $\bar{\delta}$, under the discrete prior $p[\bar{\delta}]$, is given by

$$\bar{M} = -\log_2 p[\bar{\delta}]. \tag{B.22}$$

Following common practice, this discrete cost is approximated during training

Algorithm B.2 Full-Model Fine-Tuning — Decoding

Input: Global parameters θ_0 trained on a generic dataset; model prior $p(\bar{\delta})$; bitstream $\mathbf{b} = (b_{\bar{\delta}}, b_z)$.

Output: Decompressed CSI matrix $\hat{\mathbf{H}}$.

1: Entropy decode $\bar{\delta} \leftarrow \gamma^{-1}(b_{\bar{\delta}}; p(\bar{\delta}))$
2: Compute updated decoder parameters $\bar{\theta} \leftarrow \theta_0 + \bar{\delta}$
3: Entropy decode latent $\bar{\mathbf{Z}} \leftarrow \gamma_{\bar{\theta}}^{-1}(b_z; p_{\bar{\theta}})$
4: De-quantize and decode: $\hat{\mathbf{H}} \leftarrow Q^{-1}(g_{\bar{\theta}}(\bar{\mathbf{Z}}))$
5: **Return** $\hat{\mathbf{H}}$

by its continuous counterpart [42],

$$M = -\log_2 p(\delta). \tag{B.23}$$

To regularize the bit-rate cost of model updates during full-model fine-tuning, the update cost is incorporated into the RD objective in (B.13) as

$$L_{\mathrm{RDM}}(\phi, \theta) = L_{\mathrm{RD}}(\phi, \theta) - \log_2 p(\delta), \tag{B.24}$$

By optimizing this objective during full-model fine-tuning, the network learns to balance the achievable RD gain against the bit-rate cost required to transmit the decoder model updates. Model fine-tuning is performed using CSI samples collected in the target environment, denoted by $\mathcal{H}_{\mathrm{T}}$. For evaluation, the fine-tuned model is applied to CSI samples acquired after the fine-tuning phase, denoted by $\mathcal{H}_{\mathrm{E}}$. The encoding and decoding procedures for full-model fine-tuning are summarized in Algorithms **??** and **??**, respectively.

B.4 Performance Evaluation

In this section, we evaluate the effectiveness of different model-fine-tuning schemes applied to the backbone neural CSI compressor in target environments with shifted channel statistics, along with the associated computational cost of model fine-tuning. We consider the following neural CSI compression fine-tuning schemes:

- **Pretrained**: denotes applying the pretrained neural CSI compressor directly in the target environment without any model fine-tuning.

- **EO**: corresponds to the encoder-only fine-tuning scheme. Encoder parameters are updated using CSI samples from the target environment, while the decoder remains fixed. This enables partial specialization of the pretrained model to the target environment without requiring any additional feedback. Results are reported for models trained at different values of λ.

- **FM**: refers to the full-model fine-tuning scheme, where both encoder and decoder parameters are updated. The spike-and-slab prior parameters are set to $t = 0.005$, $N = 50$, and $\sigma = 0.05$. Only the low-bit-rate neural CSI compressor trained at $\lambda = 5 \times 10^4$ is fine-tuned. The reported rate for this scheme represents the total rate, i.e., the sum of the latent bit-rate and the amortized model-update bit-rate. Minor parameter adjustments are made to achieve the best performance.

- **FM1**: Motivated by the observation that earlier layers of DNNs capture more general features, whereas later layers encode finer, task-specific details [45], model fine-tuning is restricted to the final convolutional layer of the decoder network to reduce the bit-rate cost of model updates. Accordingly, FM1 denotes a partial full-model fine-tuning scheme in which only the final convolutional layer of the decoder is updated, while all other layers remain fixed. The same update parameters as in the FM scheme are used.

- **FM1-UP**: is identical to FM1, except that a uniform prior is used for the model updates instead of the spike-and-slab prior, as originally proposed in our initial work [46]. Since only a single decoder layer is updated, resulting in a smaller number of model update parameters compared to FM, the uniform prior imposes weaker sparsity constraints and may therefore better accommodate a larger number of effective updates.

- **FM-2bit**: represents the full-model fine-tuning scheme with coarse quantization using four quantization bins (2-bit quantization). Prior results in [34], based on a federated edge learning framework, suggest that the impact of coarse quantization noise can be mitigated in the downlink stream. Motivated by this observation, the FM scheme is evaluated under this coarse quantization setting.

- **GA**: denotes a genie-aided scheme, in which the decoder is assumed to have perfect knowledge of the model updates without any feedback overhead. This scheme provides a lower bound on the achievable performance of neural CSI compression under distribution shifts. Similar to FM, only the low-bit-rate trained neural CSI compressor is used. The approaches in [29], [32] can be interpreted as instances of this genie-aided setting, as they update a vanilla autoencoder without accounting for the communication overhead of model-update feedback.

- **TM**: corresponds to the translation-module-based scheme proposed in [30]. Inspired by image-to-image translation in computer vision, this approach maps input CSI to the target domain using a convolutional translation module at the encoder and a retranslation module at the decoder. The same network architecture and sparsity-alignment function as in [30] are employed. Similar to the encoder-only fine-tuning, models trained at different values of λ are used. To provide a lower-bound rate, the communication overhead required to convey the retranslation module parameters to the decoder is also ignored.

B.4.1 RD Numerical Results

In this subsection, we present the RD results of the different fine-tuning schemes for neural CSI compression. The schemes are evaluated using the QuaDRiGa, 3GPP CDL, and DeepMIMO datasets introduced in Section B.3. Fine-tuning is carried out using 100 CSI samples from the target domain with a batch size of 50 over 100 epochs.

Fig. 6 compares the RD performance of different fine-tuning schemes across the QuaDRiGa, 3GPP CDL, and DeepMIMO datasets. From the figure, it can be observed that the gains achieved by EO remain limited compared to schemes that permit decoder-side updates. This confirms that encoder-only fine-tuning can only partially compensate for distribution shifts, while the decoder remains mismatched to the target environment, thereby highlighting the importance of full-model fine-tuning. Moreover, the TM scheme, which relies on translation and retranslation modules without fine-tuning the backbone network, does not provide noticeable RD gains, and its performance remains largely comparable to that of the pretrained model. On the other hand, full-model fine-tuning yields substantial RD improvements across all

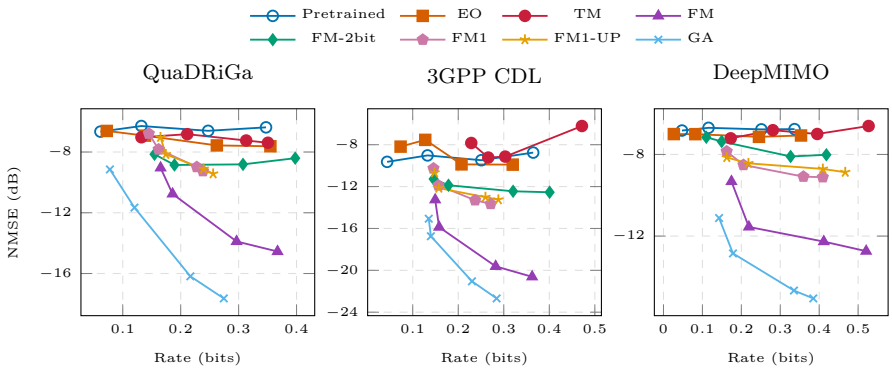


Figure 6: RD results of different fine-tuning schemes evaluated on multiple datasets.

datasets, demonstrating the benefit of jointly updating the encoder and decoder parameters. Notably, the reported rates for FM include both the latent bit-rate and the amortized model-update bit-rate, indicating that these improvements are achieved despite explicitly accounting for the communication overhead associated with decoder updates.

The partial full-model fine-tuning scheme (FM1 and FM1-UP), which restricts updating to the final convolutional layer of the decoder, underperforms full FM but outperforms EO and TM. This confirms that fine-tuning only high-level, task-specific layers is insufficient to fully capture environment-specific channel characteristics compared to full-model fine-tuning; nevertheless, it still provides an RD improvement relative to schemes in which the decoder remains unchanged. Furthermore, a comparison between FM1 and FM1-UP highlights that when only a single decoder layer is updated, weaker sparsity constraints on the model updates can be afforded, and the resulting RD performance is comparable to that achieved by FM1 with the spike-and-slab prior.

A comparison between FM and FM-2bits suggests that employing finer quantization for model updates improves RD performance. This observation contrasts with the results reported in [34] and can primarily be attributed to differences in the system setup. In particular, [34] considers a multi-user scenario with a federated learning framework, where quantization noise introduced by individual users can be partially compensated through aggregation,

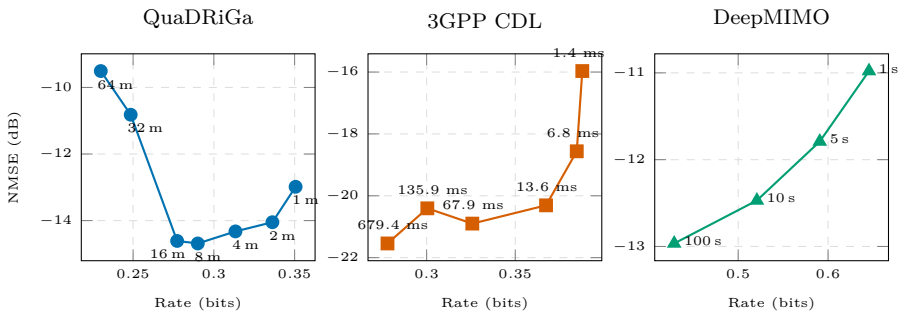


Figure 7: RD trade-off of full-model fine-tuning across different evaluation horizons. Point annotations indicate distance (m) for QuaDRiGa, channel evolution time at 120 km/h for the CDL model, and captured scene duration for DeepMIMO.

whereas such compensation is not present in our considered single-model fine-tuning setting.

Finally, the genie-aided scheme (GA) consistently provides the best RD performance and serves as a lower bound, as it assumes perfect decoder-side knowledge of model updates without any feedback overhead. The performance gap between GA and FM quantifies the cost of explicitly accounting for model-update transmission and underscores the efficiency of the proposed RDM-based full-model fine-tuning framework.

Fig. 7 illustrates the RD performance across evolving CSI samples in the QuaDRiGa, 3GPP CDL, and DeepMIMO datasets. Each operating point corresponds to a different evaluation horizon, defined as the duration over which an updated model is reused before the CSI distribution changes sufficiently to require further updates. The total feedback rate accounts for both the compressed CSI and the amortized communication cost of model updates over the evaluation horizon.

In the QuaDRiGa dataset, channel evolution is driven by user mobility along a 350 m linear track in a realistic three-dimensional urban environment, where propagation paths may appear and disappear as the user moves. This leads to pronounced structural changes in the CSI distribution over distance. Consequently, increasing the evaluation horizon results in a noticeable degradation in reconstruction quality unless model updates are transmitted more frequently. This behavior manifests as a clear trade-off between the amortized model-update rate and distortion, highlighting the limited validity period of

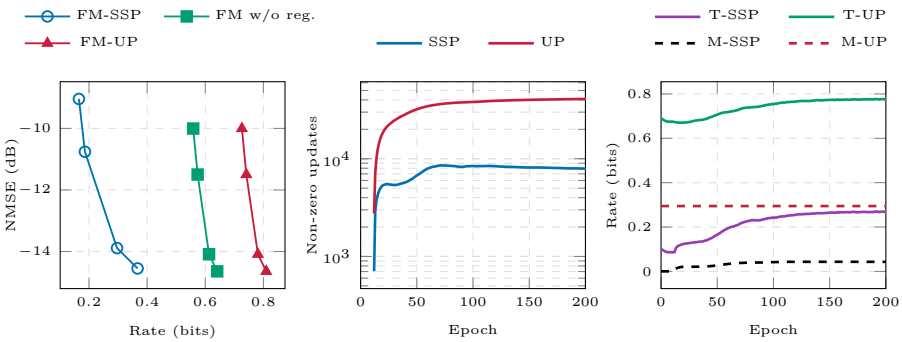


Figure 8: Ablation results for full-model scheme, highlighting the role of the spike-and-slab prior and the model-update regularizer in controlling the model-update rate.

fine-tuned models in highly nonstationary propagation environments.

In contrast, the 3GPP CDL dataset exhibits smooth temporal channel evolution dominated by Doppler effects, with a fixed propagation geometry and no abrupt path changes. As a result, the updated model remains effective over longer time spans, and the RD performance improves as the amortized model-update rate decreases with longer evaluation horizons. It is worth noting that, in this experiment, the user velocity is increased to 120 km/h and a rural macro-cell (RMa) delay profile is employed in order to accelerate CSI variations and create a more challenging evaluation scenario. Similarly, in the DeepMIMO O2 Dynamic scenario, CSI realizations are collected across discrete scenes in a site-specific urban layout with fixed geometry within each scene. Although the user environment varies across scenes, the underlying propagation characteristics remain largely unchanged, leading to quasi-stationary CSI statistics. As a result, the fine-tuned model can be reused over long evaluation horizons with limited degradation in distortion performance.

Fig. 8 presents an ablation study that examines the impact of the model-update prior and the update-rate regularizer on full-model fine-tuning. The scheme denoted as FM-SSP corresponds to full-model fine-tuning with a spike-and-slab prior on the model updates, whereas FM-UP employs a uniform prior. The variant FM w/o reg. removes the model-update regularizer from the RD loss, thereby allowing unrestricted model updates. For clarity, SSP and UP refer to the spike-and-slab and uniform priors, respectively. To analyze

Table 5: Impact of model-update quantization resolution on RD performance.

Quantization	**QuaDRiGa**		**3GPP CDL**	
	Rate	NMSE (dB)	Rate	NMSE (dB)
1 bit	0.25	-5.73	0.232	-7.88
2 bit	0.34	-8.81	0.308	-12.52
3 bit	0.33	-12.05	0.300	-16.59
4 bit	0.32	-13.70	0.287	-17.56
5 bit	0.30	-13.89	0.277	-17.59
6 bit	0.30	-13.89	0.276	-17.59
7 bit	0.30	-13.89	0.276	-17.59

the rate behavior in detail, the total feedback rate is decomposed into its components. The terms T-SSP and T-UP denote the total rate, including both the latent bit-rate and the amortized model-update rate, when using the spike-and-slab and uniform priors, respectively. In contrast, M-SSP and M-UP represent only the model-update rate associated with each prior.

The results in Fig. 8 lead to several conclusions regarding full-model fine-tuning. First, incorporating the model-update regularizer in the RD loss is essential for controlling the bit-rate cost of model updates. When the regularizer is removed (FM w/o reg.), the model-update rate increases rapidly, resulting in a substantially higher total rate without corresponding RD gains. This demonstrates that unconstrained full-model fine-tuning can lead to an excessive increase in model-update bit rate.

Second, the choice of prior has a pronounced impact on the sparsity and rate of model updates. The spike-and-slab prior consistently yields fewer nonzero updates and a lower model-update rate compared to the uniform prior, while achieving strong RD performance. This confirms that explicitly promoting sparsity in the model updates is critical for reducing the feedback overhead associated with model fine-tuning.

Table 5 reports the RD performance of the full-model scheme under different quantization resolutions for model updates, evaluated on the QuaDRiGa and CDL datasets. The results indicate that low-resolution quantization (below 3 bits) leads to substantial RD degradation. In contrast, increasing the quantization resolution beyond 4 bits does not yield any noticeable RD im-

Table 6: RD performance for different dataset size for full-model fine-tuning.

Dataset Size	**QuaDRiGa**		**3GPP CDL**	
	Rate	NMSE (dB)	Rate	NMSE (dB)
1	0.248	-7.03	0.241	-10.48
10	0.286	-10.38	0.267	-15.09
50	0.299	-11.79	0.269	-15.26
100	0.297	-13.89	0.275	-17.64
200	0.305	-14.63	0.288	-18.86
300	0.316	**-13.96**	0.302	-19.98
500	0.325	**-10.39**	0.330	-20.75

provement, suggesting diminishing returns from high-resolution quantization. Overall, moderate-resolution quantization is sufficient to achieve good compression performance.

The results in Table 6 illustrate the impact of the dataset size on the RD performance of the full-model fine-tuning scheme. The results indicate that increasing the number of CSI samples used for fine-tuning initially leads to clear improvements in RD performance. However, beyond a certain dataset size, further increasing the number of training samples yields only marginal performance gains, suggesting that a few hundred CSI samples are sufficient for effective full-model fine-tuning. In the QuaDRiGa dataset, using a larger number of CSI samples (e.g., 300–500 samples) leads to a degradation in NMSE performance. This behavior is attributed to the increased heterogeneity of the fine-tuning data, as larger training sets include more diverse channel geometries and spatial conditions. As a result, the fine-tuned model becomes less locally specialized, which is suboptimal for the fixed evaluation horizon considered in this study.

Table 7 evaluates the sensitivity of the spike-and-slab prior hyperparameters (α, σ, t) on RD performance for QuaDRiGa dataset. We vary one hyperparameter at a time while keeping the others fixed, as indicated in each block. From the t-sweep, we observe that extremely small values of t lead to overly strong shrinkage toward zero, which destabilizes the update distribution and results in increased rate and degraded NMSE. In this regime, moderate-magnitude updates are excessively penalized and are inefficiently

Table 7: Sensitivity analysis of the spike-and-slab prior hyperparameters.

α	σ	t	Rate	NMSE (dB)
t sweep (α=100, σ=0.05)				
100	0.05	0.0005	0.577	-9.60
100	0.05	0.001	0.346	-12.90
100	0.05	0.005	0.314	-13.59
100	0.05	0.01	0.275	-13.17
100	0.05	0.05	0.225	-6.25
σ sweep (α=100, t=0.005)				
100	0.005	0.005	NaN	NaN
100	0.01	0.005	inf	-13.71
100	0.05	0.005	0.314	-13.59
100	0.1	0.005	0.317	-13.68
100	0.5	0.005	0.320	-13.53
100	1	0.005	0.319	-13.58
α sweep (σ=0.005, t=0.05)				
0.1	0.005	0.05	0.506	-13.50
1	0.005	0.05	0.363	-13.79
10	0.005	0.05	0.315	-13.73
100	0.005	0.05	0.314	-13.59
1000	0.005	0.05	0.320	-13.55
10000	0.005	0.05	0.322	-13.54

encoded through the slab component. On the other hand, excessively large t weakens the sparsity-inducing effect of the spike component, reducing the separation between near-zero and significant updates and degrading the RD performance. A broad stable region is observed for $t \in [10^{-3}, 10^{-2}]$, where the RD trade-off is well balanced.

For the σ-sweep, very small slab variances lead to numerical instability, as the slab component becomes excessively narrow and cannot accommodate moderate update magnitudes. This results in unstable entropy estimates and unreliable coding behavior. In contrast, for $\sigma \geq 0.05$, the RD performance remains stable, with only minor variations across more than one order of mag-

nitude, indicating that the method is not sensitive to the precise choice of slab scale within this range. Finally, the α-sweep shows that α primarily governs the balance between the spike and slab components, thereby controlling the effective sparsity level of the updates. Across a wide range ($\alpha \in [1, 10^4]$), the RD trade-off exhibits only modest variation, with diminishing changes for large α.

B.4.2 Computational Complexity

We analyze the computational cost of the proposed full-model fine-tuning by explicitly accounting for the number of floating-point operations (FLOPs) required by the neural CSI compressor considered in this work. The training complexity of model fine-tuning naturally depends on the chosen backbone architecture, with simpler or more complex networks incurring lower or higher computational costs, respectively. In this study, the backbone neural CSI compressor consists of stacked 5×5 convolutional and transposed convolutional layers operating on CSI tensors of spatial resolution 64×64.

For a 2D convolution producing an output tensor of size $H \times W \times C_{\text{out}}$ with kernel size $k \times k$ and C_{in} input channels, the number of multiply–accumulate operations (MACs) is given by $HWC_{\text{out}}(k^2 C_{\text{in}})$, where one MAC is counted as two FLOPs. Summing the contributions of all convolutional layers, a single forward pass through the encoder–decoder architecture requires approximately 0.48 GFLOPs per CSI sample. To account for backpropagation and gradient computation during fine-tuning, we adopt a standard approximation whereby the total training cost is three times the forward-pass complexity, resulting in approximately 1.43 GFLOPs per CSI sample per fine-tuning iteration.

The on-device update is performed using $N_T = 100$ CSI samples, a batch size of $B = 50$, and $E = 100$ epochs, corresponding to two optimization steps per epoch and 200 steps in total. Under these settings, the total fine-tuning cost is given by

$$\text{FLOPs}_{\text{total}} \approx 1.43 \times N_T \times E \approx 14.25 \text{ TFLOPs}. \tag{B.25}$$

The expected fine-tuning latency can be approximated as $T \approx \text{FLOPs}_{\text{total}}/\eta$, where η denotes the effective sustained throughput of the mobile processor in TFLOPs/s [47]. For typical mobile devices, η may range from 0.1 to

1 TFLOPs/s depending on the available hardware acceleration and numerical precision, resulting in fine-tuning latencies on the order of several tens of seconds.

The corresponding energy consumption scales linearly with the total number of operations and can be expressed as $E \approx \text{FLOPs}_{\text{total}} \epsilon_{\text{FLOP}}$, where ϵ_{FLOP} denotes the energy consumed per floating-point operation. For representative values of ϵ_{FLOP} between 10 and 100 pJ/FLOP [47], the total energy cost of fine-tuning lies in the range of 0.04–0.4 Wh, corresponding to a small fraction of a typical smartphone battery capacity. In practical deployments, the fine-tuning will be executed as a background process and invoked only occasionally, e.g., when significant distribution shifts are detected. As a result, the fine-tuning cost can be amortized over time and is unlikely to interfere with latency-critical operations.

B.5 Conclusion

A major limitation of neural compression approaches is their significant performance degradation when channel statistics change, which is a natural phenomenon in wireless environments. To address this distribution shift problem, we proposed a fine-tuning scheme for neural CSI compression that explicitly accounts for the communication overhead associated with transmitting model updates. Lossless entropy coding is applied to both latent representations and model updates, and a spike-and-slab prior is adopted to promote sparse and efficient parameter updates. Furthermore, the bit-rate of model updates is incorporated into the fine-tuning process through a regularized RD loss function. We conducted simulations across a wide range of wireless CSI datasets, covering both site-specific and stochastic channel models. The results demonstrate that full-model fine-tuning is essential for effectively adapting the neural CSI compressor to varying environments.

We further evaluated full-model fine-tuning under different evaluation horizons, quantization resolutions, and fine-tuning sample sizes. The results show that in environments with rapidly varying CSI statistics, applying fine-tuned models over excessively long horizons can lead to noticeable RD degradation. In addition, using highly diverse CSI samples for fine-tuning may reduce model specialization within a localized operating region. We also observe that moderate quantization resolutions, on the order of 4–5 bits, are sufficient to

achieve strong RD performance. As a future direction, this work can be extended toward fully online learning of neural CSI compression. In this setting, the encoder would dynamically adapt its parameters in response to evolving CSI statistics.

References

[1] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE Trans. Wirel. Commun.*, vol. 9, no. 11, pp. 3590–3600, Oct. 2010.

[2] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 186–195, 2014.

[3] X. Rao and V. K. N. Lau, "Distributed compressive CSIT estimation and feedback for FDD multi-user massive MIMO systems," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3261–3271, 2014.

[4] X.-L. Huang, J. Wu, Y. Wen, F. Hu, Y. Wang, and T. Jiang, "Rate-adaptive feedback with bayesian compressive sensing in multiuser MIMO beamforming systems," *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 4839–4851, 2016.

[5] V. Rizzello, M. Nerini, M. Joham, B. Clerckx, and W. Utschick, "User-driven adaptive CSI feedback with ordered vector quantization," *IEEE Wireless Commun. Lett.*, vol. 12, no. 11, pp. 1956–1960, 2023.

[6] R. Bhagavatula and R. W. Heath, "Predictive vector quantization for multicell cooperation with delayed limited feedback," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2588–2597, 2013.

[7] C. Wen, W. Shih, and S. Jin, "Deep learning for massive MIMO CSI feedback," *IEEE Wireless Commun. Lett.*, vol. 7, no. 5, pp. 748–751, Mar. 2018.

[8] J. Guo, C.-K. Wen, S. Jin, and G. Y. Li, "Convolutional neural network-based multiple-rate compressive sensing for massive MIMO CSI feedback: Design, simulation, and analysis," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2827–2840, 2020.

[9] Z. Lu, J. Wang, and J. Song, "Multi-resolution CSI feedback with deep learning in massive MIMO system," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[10] S. Tang, J. Xia, L. Fan, X. Lei, W. Xu, and A. Nallanathan, "Dilated convolution based CSI feedback compression for massive MIMO systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 10, pp. 11 216–11 221, 2022.

[11] Y. Cui, A. Guo, and C. Song, "Transnet: Full attention network for CSI feedback in FDD massive MIMO system," *IEEE Wireless Commun. Lett.*, vol. 11, no. 5, pp. 903–907, 2022.

[12] Z. Hu, G. Liu, Q. Xie, J. Xue, D. Meng, and D. Gündüz, "A learnable optimization and regularization approach to massive MIMO CSI feedback," *IEEE Trans. Wireless Commun.*, vol. 23, no. 1, pp. 104–116, 2024.

[13] H. Wu, M. Zhang, Y. Shao, K. Mikolajczyk, and D. Gündüz, *MIMO channel as a neural function: Implicit neural representations for extreme CSI compression in massive MIMO systems*, 2024.

[14] M. B. Mashhadi, Q. Yang, and D. Gündüz, "Distributed deep convolutional compression for massive mimo csi feedback," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2621–2633, 2021.

[15] W. Chen, W. Wan, S. Wang, P. Sun, G. Y. Li, and B. Ai, *CSI-PPPNet: A one-sided one-for-all deep learning framework for massive MIMO CSI feedback*, 2023.

[16] Y. Yang, S. Mandt, and L. Theis, "An introduction to neural data compression," *Found. Trends. Comput. Graph. Vis.*, vol. 15, no. 2, pp. 113–200, Apr. 2023, ISSN: 1572-2740.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," in *Adv. Neural Inf. Process. Syst.*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014.

[18] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd Int. Conf. Learn. Represent. (ICLR) 2014, Banff, AB, Canada, Apr. 14-16, 2014, Conf. Track Proc.*, 2014.

[19] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proc. 14th Int. Conf. Artif. Intell. Stat. (AISTATS)*, G. Gordon, D. Dunson, and M. Dudík, Eds., ser. Proc. Mach. Learn. Res. Vol. 15, Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 29–37.

[20] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proc. Mach. Learn. Res. Vol. 48, New York, NY, USA: PMLR, 20–22 Jun 2016, pp. 1747–1756.

[21] Q. Yang, M. B. Mashhadi, and D. Gündüz, "Deep convolutional compression for massive MIMO CSI feedback," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, 2019, pp. 1–6.

[22] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. MIT Press, 2022.

[23] F. Zhuang, Z. Qi, K. Duan, *et al.*, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[24] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.

[25] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Int. Conf. Mach. Learn. (ICML)*, PMLR, 2015, pp. 1180–1189.

[26] M. Sattari, H. Guo, D. Gündüz, A. Panahi, and T. Svensson, "Full-duplex millimeter wave MIMO channel estimation: A neural network approach," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 1093–1108, 2024.

[27] M. Khani, M. Alizadeh, J. Hoydis, and P. Fleming, "Adaptive neural signal detection for massive mimo," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 8, pp. 5635–5648, May 2020.

[28] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Airnet: Neural network transmission over the air," in *2022 IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 2451–2456.

[29] J. Zeng, J. Sun, G. Gui, *et al.*, "Downlink CSI feedback algorithm with deep transfer learning for FDD massive MIMO systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1253–1265, 2021.

[30] Z. Liu, L. Wang, L. Xu, and Z. Ding, "Deep learning for efficient CSI feedback in massive MIMO: Adapting to new environments and small datasets," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2024.

[31] X. Li, J. Guo, C.-K. Wen, S. Jin, S. Han, and X. Wang, "Multi-task learning-based CSI feedback design in multiple scenarios," *IEEE Trans. Commun.*, vol. 71, no. 12, pp. 7039–7055, 2023.

[32] B. Zhang, H. Li, X. Liang, X. Gu, and L. Zhang, "Model transmission-based online updating approach for massive MIMO CSI feedback," *IEEE Communications Letters*, vol. 27, no. 6, pp. 1609–1613, 2023.

[33] X. Zhang, J. Wang, Z. Lu, and H. Zhang, "Continuous online learning-based CSI feedback in massive MIMO systems," *IEEE Communications Letters*, vol. 28, no. 3, pp. 557–561, 2024.

[34] Y. Cui, J. Guo, C.-K. Wen, and S. Jin, "Communication-efficient personalized federated edge learning for massive MIMO CSI feedback," *IEEE Transactions on Wireless Communications*, vol. 23, no. 7, pp. 7362–7375, 2024.

[35] J. Guo, Y. Zuo, C.-K. Wen, and S. Jin, "User-centric online gossip training for autoencoder-based CSI feedback," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 559–572, 2022.

[36] S. Jaeckel, L. Raschkowski, K. Börner, and L. Thiele, "Quadriga: A 3-d multi-cell channel model with time evolution for enabling virtual field trials," *IEEE Trans. Antennas Propag.*, vol. 62, no. 6, pp. 3242–3256, 2014.

[37] "Study on channel model for frequencies from 0.5 to 100 GHz," 3GPP, Tech. Rep. TR 38.901, version 17.0.0, Mar. 2022, Available: `https://www.3gpp.org/ftp/Specs/archive/38_series/38.901/`.

[38] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," in *Proc. of Information Theory and Applications Workshop (ITA)*, San Diego, CA, Feb. 2019, pp. 1–8.

[39] L. Theis and E. Agustsson, "On the advantages of stochastic encoders," *CoRR*, vol. abs/2102.09270, 2021.

[40] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Int. Conf. Learn. Represent. (ICLR)*, 2018.

[41] Remcom, "Wireless InSite," `http://www.remcom.com/wireless-insite`.

[42] T. van Rozendaal, I. A. Huijben, and T. Cohen, "Overfitting for fun and profit: Instance-adaptive data compression," in *Int. Conf. Learn. Represent. (ICLR)*, 2021.

[43] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, 2013, Accessed: 2025-01-27.

[44] V. Ročková and E. I. G. and, "The spike-and-slab lasso," *Journal of the American Statistical Association*, vol. 113, no. 521, pp. 431–444, 2018.

[45] O. Jourairi, M. Balcilar, A. Lambert, and F. Schnitzler, "Improving The Reconstruction Quality by Overfitted Decoder Bias in Neural Image Compression," in *2022 Picture Coding Symposium (PCS)*, 2022, pp. 61–65.

[46] M. Sattari, D. Gündüz, and T. Svensson, "Dynamically Fine-Tuned Neural Compressor for FDD Massive MIMO CSI Feedback," in *2025 IEEE 26th International Workshop on Signal Processing and Artificial Intelligence for Wireless Communications (SPAWC)*, 2025, pp. 1–5.

[47] P. Hübner, A. Hu, I. Peng, and S. Markidis, "Apple vs. Oranges: Evaluating the Apple Silicon M-Series SoCs for HPC Performance and Efficiency," in *2025 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2025, pp. 45–54.

PAPER C

CSI Prediction Using Diffusion Models

Mehdi Sattari, Javad Aliakbari, Alexandre Graell i Amat,
Tommy Svensson

IEEE Transactions on Wireless Communications, major revision, Feb, 2026.

The layout has been revised.

Abstract

Acquiring accurate channel state information (CSI) is critical for reliable and efficient wireless communication, but challenges such as high pilot overhead and channel aging hinder timely and accurate CSI acquisition. CSI prediction, which forecasts future CSI from historical observations, offers a promising solution. Recent deep learning approaches, including recurrent neural networks and Transformers, have achieved notable success but typically learn deterministic mappings, limiting their ability to capture the stochastic and multimodal nature of wireless channels. In this paper, we introduce a novel probabilistic framework for CSI prediction based on diffusion models, offering a flexible design that supports integration of diverse prediction schemes. We decompose the CSI prediction task into two components: a temporal encoder, which extracts channel dynamics, and a diffusion-based generator, which produces future CSI samples. We investigate two inference schemes—autoregressive and sequence-to-sequence—and explore multiple diffusion backbones, including U-Net and Transformer-based architectures. Furthermore, we examine a diffusion-based approach without an explicit temporal encoder and utilize the DDIM scheduling to reduce model complexity. Extensive simulations demonstrate that our diffusion-based models significantly outperform state-of-the-art baselines.

C.1 Introduction

Multiple-input multiple-output (MIMO) technology plays a pivotal role in advanced wireless communications. To fully exploit the benefits of MIMO systems, accurate acquisition of channel state information (CSI) is essential for designing efficient precoding and combining algorithms. However, CSI acquisition is challenged by high overhead, channel aging, and significant computational complexity. To address these issues, a variety of algorithms have been proposed to obtain CSI with reduced overhead and complexity [1]–[5].

CSI prediction refers to the task of forecasting future CSI based on historical observations. Various classical approaches have been proposed, including linear extrapolation [6], Kalman filtering [7], sum-of-sinusoids models [8], Gaussian processes [9], predictor antenna [10], and autoregressive (AR) models [11]. However, the underlying statistics of wireless channels are inherently complex and difficult to model accurately, making reliable CSI prediction particularly challenging. As a result, traditional methods often fail to deliver satisfactory performance, especially under highly dynamic channel conditions.

Deep neural networks (DNNs) have been increasingly adopted in wireless communication tasks. Motivated by their significant success in time-series forecasting problems [12]–[16], the use of DNNs for CSI prediction has attracted growing research interest [17]–[31]. Various neural network (NN) architectures have been explored to capture the temporal dynamics of CSI, including recurrent neural networks (RNNs), such as long short-term memory (LSTM) and gated recurrent units (GRUs), as well as Transformer-based models. RNN-based architectures have been successfully employed for CSI prediction in [20], [21], [24], while Transformer-based approaches have been investigated in [25], [26], showing superior performance in modeling long-range temporal dependencies.

Accurately predicting CSI is a fundamentally challenging problem due to the highly stochastic nature of wireless channels and their complex underlying distributions. Future channel states are influenced by a wide range of factors, including multipath fading, user mobility, and environmental dynamics, which together lead to significant uncertainty in CSI evolution. Deep learning models such as RNNs and Transformers have shown promising results for time-series forecasting, but they primarily learn a deterministic mapping from historical CSI to future CSI. This approach often fails to capture the inherent randomness and multimodality of wireless channel evolution. In contrast, recent advances in generative artificial intelligence (AI) have enabled the development of models that learn the full underlying probability distribution of data. Among these, diffusion models [32] have achieved state-of-the-art performance in high-dimensional data generation tasks across various modalities. By modeling CSI prediction as a probabilistic generative process, diffusion models can naturally account for uncertainty in channel evolution while learning the joint spatiotemporal structure of CSI sequences.

Motivated by the success of recent advances of diffusion models in modeling

complex data distributions and the limitations of RNN- and Transformer-based approaches in CSI prediction, this paper explores the application of diffusion models to the CSI prediction problem. The main contributions of this work are summarized as follows:

- We propose, for the first time, a diffusion-based framework for CSI prediction. The framework offers a flexible design that enables seamless integration of diverse CSI prediction models. We formulate the CSI prediction task as two components: temporal encoding and generative sampling. The temporal encoder extracts latent representations of the channel dynamics, while the diffusion generator produces the next CSI frame conditioned on these latent features.

- We explore two inference schemes: AR and sequence-to-sequence (seq2seq). The AR strategy recursively predicts future CSI frames, enabling a single trained model to flexibly adapt to arbitrary context lengths and prediction horizons. In contrast, the seq2seq scheme enables parallel prediction of multiple future CSI frames in a single pass, albeit with reduced flexibility in handling variable context lengths and prediction horizons.

- To reduce complexity for real-time applications, we investigate a simplified design in which the diffusion model directly learns temporal and spatial dependencies without an explicit temporal encoder. Furthermore, to enhance sampling efficiency and stability, we adopt the denoising diffusion implicit model (DDIM) [33] scheduling technique. Our experiments show that as few as three sampling steps are sufficient to achieve competitive performance.

- We evaluate multiple generator backbones, including U-Net [34], 3D U-Net [35], and diffusion Transformer (DiT) [36], in combination with temporal encoders such as ConvLSTM [37] and LinFormer [26]. Extensive simulation results show that our diffusion-based models significantly outperform state-of-the-art baselines and exhibit strong generalization to unseen wireless environments.

The rest of this paper is organized as follows: Section C.2 introduces the channel model and CSI prediction problem, as well as a preliminary for diffusion models. In Section C.3, we present our diffusion-based CSI prediction

framework and training and inference procedures. Section C.4 provides numerical simulations and compares different CSI prediction schemes in terms of the normalized mean squared error (NMSE). Finally, Section C.5 concludes the paper.

C.2 Preliminaries and Problem Formulation

C.2.1 Channel Model

We focus on the clustered delay line (CDL) channel model based on the 3GPP specification [38]. The CDL channel model is a spatial channel model for wireless channel simulation. This model comprises different scenarios, denoted as `CDL-A` to `CDL-E`, where each scenario models different channel statistics with different delay profiles, angular spreads, etc. For example, `CDL-B` generates an urban macrocell scenario with a wider angular spread. The time-varying MIMO channel impulse response in the CDL model is given by

$$\mathbf{H}(t) = \sum_{l=1}^{L} \sum_{r=1}^{R} \alpha_{l,r}(t)\, \mathbf{a}_{\mathrm{Rx}}\left(\phi_{l,r}^{\mathrm{Rx}}\right)\, \mathbf{a}_{\mathrm{Tx}}^{H}\left(\phi_{l,r}^{\mathrm{Tx}}\right)\, e^{-j2\pi f \tau_l}, \tag{C.1}$$

or its discrete counterpart,

$$\mathbf{H}_n = \sum_{l=1}^{L} \sum_{r=1}^{R} \alpha_{l,r}[n]\, \mathbf{a}_{\mathrm{Rx}}\left(\phi_{l,r}^{\mathrm{Rx}}\right)\, \mathbf{a}_{\mathrm{Tx}}^{H}\left(\phi_{l,r}^{\mathrm{Tx}}\right)\, e^{-j2\pi \frac{k f_s}{N_c} \tau_l}, \tag{C.2}$$

where, L is the number of clusters, R the number of rays per cluster, $\alpha_{l,r}(t)$ time-varying complex path gain for ray r in cluster l, $\mathbf{a}_{\mathrm{Rx}}(\cdot), \mathbf{a}_{\mathrm{Tx}}(\cdot)$ denote receive/transmit array response vectors, $(\phi^{\mathrm{Rx}}, \phi^{\mathrm{Tx}})$ represent angles of arrival/departure, f_s the sampling frequency, N_c the number of subcarriers, and τ_l is the delay of the l-th cluster. The time-varying complex path gain is modeled as

$$\alpha_{l,r}[n] = \sqrt{P_{l,r}}\, e^{j(2\pi f_{l,r} n + \phi_{l,r})}, \tag{C.3}$$

where $P_{l,r}$ is the power of the r-th ray in cluster l, $f_{l,r}$ the Doppler shift, and $\phi_{l,r}$ the random initial phase. For a uniform linear array (ULA), the array

response vector is

$$\mathbf{a}(\phi) = \frac{1}{\sqrt{N_\mathrm{t}}} \left[1,\, e^{j2\pi \frac{d}{\lambda} \sin(\phi)},\, \ldots,\, e^{j2\pi \frac{d}{\lambda} (N_\mathrm{t}-1) \sin(\phi)}\right]^T, \tag{C.4}$$

where, N_t denotes the number of antenna elements and d the antenna spacing.

We consider a MIMO system in which a base station (BS) serves multiple single-antenna users. The BS is equipped with a ULA of N_t antennas. Orthogonal frequency division multiplexing (OFDM) with N_c subcarriers is employed for downlink transmission. For each subcarrier $m \in \{1, \ldots, N_\mathrm{c}\}$ and time index n, $\mathbf{h}_{m,n} \in \mathbb{C}^{N_\mathrm{t}}$ denotes the channel vector from the BS with N_t antennas to the user. By stacking all subcarriers, the downlink CSI matrix in the spatial–frequency domain at time n is given by

$$\mathbf{H}_n = \left[\mathbf{h}_{1,n}\, \mathbf{h}_{2,n}\, \cdots\, \mathbf{h}_{N_\mathrm{c},n}\right] \in \mathbb{C}^{N_\mathrm{t} \times N_\mathrm{c}}. \tag{C.5}$$

C.2.2 Diffusion Models

Denoising diffusion probabilistic models (DDPMs): Diffusion models are a class of generative models that learn to approximate complex data distributions by progressively corrupting the data with noise and then reversing this process. The foundational work on DDPM was introduced in [32]. In DDPMs, the forward process gradually adds Gaussian noise to the data, while the reverse process aims to reconstruct the original data using a NN. This is modeled as a Markov chain with a predefined, time-dependent noise schedule.

The forward process defines a Markov chain that incrementally perturbs the data over T time steps. Starting from a data sample $\mathbf{H}^0 \sim p(\mathbf{H}^0)$, the forward diffusion is expressed as

$$q(\mathbf{H}^{1:T} \mid \mathbf{H}^0) = \prod_{t=1}^{T} q(\mathbf{H}^t \mid \mathbf{H}^{t-1}), \tag{C.6}$$

where each conditional distribution is Gaussian

$$q(\mathbf{H}^t \mid \mathbf{H}^{t-1}) = \mathcal{N}\left(\mathbf{H}^t; \sqrt{1-\beta_t}\, \mathbf{H}^{t-1}, \beta_t\, \mathbf{I}\right), \tag{C.7}$$

with $\beta_t \in (0, 1)$ denoting the noise variance at step t, and $\mathbf{H}^t$ representing the noisy latent at time t.

The marginal distribution at an arbitrary time step t, conditioned directly on the original sample $\mathbf{H}^0$, is

$$q(\mathbf{H}^t \mid \mathbf{H}^0) = \mathcal{N}\left(\mathbf{H}^t; \sqrt{\bar{\alpha}_t}\,\mathbf{H}^0, (1-\bar{\alpha}_t)\,\mathbf{I}\right), \tag{C.8}$$

which leads to the reparameterization

$$\mathbf{H}^t = \sqrt{\bar{\alpha}_t}\,\mathbf{H}^0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{C.9}$$

where $\boldsymbol{\epsilon}$ denotes standard Gaussian noise, and

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i, \quad \text{with} \quad \alpha_t \triangleq 1 - \beta_t. \tag{C.10}$$

The noise schedule is crafted such that $\bar{\alpha}_T \approx 0$, ensuring that the final latent $\mathbf{H}^T$ is nearly standard Gaussian.

The reverse process is a learned Markov chain that denoises $\mathbf{H}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ back to the data domain, and each reverse transition is approximated as

$$p_{\boldsymbol{\theta}}(\mathbf{H}^{t-1} \mid \mathbf{H}^t) = \mathcal{N}\left(\mathbf{H}^{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{H}^t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(t)\right), \tag{C.11}$$

with the prior given by

$$p(\mathbf{H}^T) = \mathcal{N}(\mathbf{H}^T; \mathbf{0}, \mathbf{I}). \tag{C.12}$$

In DDPM, the mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{H}^t, t)$ is expressed using a predicted noise function $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{H}^t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{H}^t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\, \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t) \right), \tag{C.13}$$

where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t)$ is an NN trained to predict the forward noise at time t. The variance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(t)$ is usually fixed and time-dependent

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(t) = \tilde{\beta}_t\,\mathbf{I}, \quad \text{with} \quad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t. \tag{C.14}$$

The model is trained by minimizing a simplified variational lower bound, which reduces to a weighted mean squared error (MSE) between the true noise

$\boldsymbol{\epsilon}$ and the predicted noise $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$

$$\mathcal{L}_{\text{noise}} = \mathbb{E}_{\mathbf{H}^0,t,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t)\right\|^2\right]. \tag{C.15}$$

An alternative yet equivalent training objective is to have the model directly predict the clean data $\mathbf{H}^0$ from the noisy observation $\mathbf{H}^t$. In this case, the training loss becomes the MSE between the true clean sample and the predicted sample $\mathbf{H}^0_{\boldsymbol{\theta}}$

$$\mathcal{L}_{\text{data}} = \mathbb{E}_{\mathbf{H}^0,t,\boldsymbol{\epsilon}}\left[\left\|\mathbf{H}^0 - \mathbf{H}^0_{\boldsymbol{\theta}}(\mathbf{H}^t, t)\right\|^2\right]. \tag{C.16}$$

This formulation is particularly useful in inverse problems, such as denoising or super-resolution, where the objective is to recover a clean signal rather than generate diverse samples. Both loss functions are mathematically connected through the forward diffusion equation, and the predicted clean sample can be converted to noise (or vice versa) using closed-form expressions.

DDIM: The DDPM framework involves a stochastic reverse process and often requires a large number of diffusion steps (e.g., $T = 1000$) to achieve high-quality generation. To reduce the number of steps without significant degradation in quality, Song *et al.* [33] proposed DDIM, which reinterpret the reverse process as a non-Markovian mapping that can be made fully deterministic. Starting from the DDPM estimate of the clean sample $\hat{\mathbf{H}}^0$ at time step t

$$\hat{\mathbf{H}}^0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{H}^t - \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t)\right), \tag{C.17}$$

and the DDIM update is given by

$$\mathbf{H}^{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{\mathbf{H}}^0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\,\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t) + \sigma_t\,\boldsymbol{\epsilon}, \tag{C.18}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ and

$$\sigma_t = \zeta \cdot \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}} \cdot \sqrt{1-\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}}, \tag{C.19}$$

where $\zeta \in [0, 1]$ controls the level of stochasticity, $\zeta = 0$ yields a fully deterministic process, while $\zeta = 1$ recovers the stochastic DDPM update. This

formulation eliminates the need to sample fresh Gaussian noise at each step in the deterministic setting ($\zeta = 0$), enabling fast sampling and allowing for arbitrary step schedules. By tuning ζ, DDIM provides a trade-off between sample diversity and generation speed.

C.2.3 Problem Formulation

Time-varying wireless channels can be naturally modeled as time-series data, where CSI prediction entails capturing both the spatial distributions and temporal dynamics of the channel. Under the MSE criterion, the optimal predictor corresponds to the minimum mean squared error (MMSE) estimate, which can be formulated as

$$f^*(\mathbf{H}_\mathrm{p}) = \arg\min_{f} \ \mathbb{E}\big[\|\mathbf{H}_\mathrm{f} - f(\mathbf{H}_\mathrm{p})\|^2\big] \,, \tag{C.20}$$

where $\mathbf{H}_\mathrm{f} \triangleq \{\mathbf{H}_{n+1}, \mathbf{H}_{n+2}, \ldots, \mathbf{H}_{n+N_\mathrm{f}}\}$ denotes the set of future CSI over the next N_f time steps, and $\mathbf{H}_\mathrm{p} \triangleq \big\{\mathbf{H}_{n-N_\mathrm{p}+1}, \mathbf{H}_{n-N_\mathrm{p}+2}, \ldots, \mathbf{H}_n\big\}$ represents the historical CSI observations over the previous N_p time steps. The optimal predictor $f^*(\mathbf{H}_\mathrm{p})$ is the conditional mean estimator (CME) [39], given by

$$f^*(\mathbf{H}_\mathrm{p}) = \mathbb{E}[\mathbf{H}_\mathrm{f} \mid \mathbf{H}_\mathrm{p}] = \int \mathbf{H}_\mathrm{f}\, p(\mathbf{H}_\mathrm{f} \mid \mathbf{H}_\mathrm{p})\, d\mathbf{H}_\mathrm{f}. \tag{C.21}$$

Deriving the optimal MMSE predictor requires knowledge of both the conditional distribution of the future CSI given the past (capturing the channel dynamics) and the prior distribution of the wireless channel (capturing the spatial statistics). Deriving this predictor in closed form is intractable due to high-dimensional integrals, nonlinear time-varying dynamics, and the lack of a known prior for practical channels.

A practical approach to approximate the optimal predictor is to employ discriminative DNNs, such as RNNs or Transformers. A discriminative DNN with learnable parameters $\boldsymbol{\theta}$ aims to learn the nonlinear mapping from historical CSI observations, $\mathbf{H}_\mathrm{p}$, to the future CSI, $\mathbf{H}_\mathrm{f}$. With sufficient data and model capacity, a DNN trained using the MSE loss provides a data-driven approximation of the MMSE predictor, i.e.,

$$f_{\boldsymbol{\theta}}(\mathbf{H}_\mathrm{p}) \approx f^*(\mathbf{H}_\mathrm{p}). \tag{C.22}$$

However, such approximations face several limitations. First, jointly capturing the temporal dynamics of the wireless channel together with its spatial distribution is highly challenging. Second, discriminative models provide a deterministic mapping that fails to capture the uncertainty of channel evolution. Third, learning such complex dynamics from finite training data makes these models prone to overfitting, thereby limiting their robustness in practical deployments.

C.3 Diffusion CSI Prediction

Inspired by the formulation of the optimal CME in (C.21), we decompose the prediction task into two components: a temporal encoder and a generator. The temporal encoder extracts latent temporal representations from the time-varying channel, while the generator learns the underlying channel distribution. To capture the probabilistic nature of CSI prediction, we employ diffusion models as the generator. In contrast to discriminative DNNs that produce deterministic point estimates, diffusion models yield probabilistic predictions, thereby accounting for the inherent uncertainty of wireless channels. Furthermore, since diffusion models are trained across the entire signal-to-noise ratio (SNR) range, they are expected to generalize more robustly to diverse channel conditions compared to task-specific discriminative models.

We apply two different inference schemes: the first follows an AR approach, while the second adopts a seq2seq prediction framework. In the following sections, we elaborate on these prediction schemes and summarize the overall methodology in algorithmic formulations. The overall training procedure of our diffusion-based CSI predictor, which leverages latent representations from the temporal encoder, is outlined in Algorithm C.1. Note that the training process is identical for both AR and seq2seq schemes, except for the output target: in the AR scenario, the model is trained to generate the next CSI frame, whereas in the seq2seq case, it is trained to produce a sequence of CSI samples over a fixed prediction horizon.

Algorithm C.1 Training

Input: Dataset $\mathcal{D} = \{(\mathbf{X}, \mathbf{Y})\}$; model parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_{\mathrm{TE}}, \boldsymbol{\theta}_{\mathrm{G}}]$; diffusion scheduler (noise schedule $\{\beta_t\}_{t=1}^{T}$); diffusion steps T; SNR range $[\rho_{\min}, \rho_{\max}]$.
Output: Trained model parameters $\boldsymbol{\theta}^*$.

for epoch $= 1, \ldots, N_{\text{epochs}}$ **do**
  for each batch $(\mathbf{X}, \mathbf{Y}) \in \mathcal{D}$ **do**
    $\rho \sim \mathcal{U}[\rho_{\min}, \rho_{\max}]$
    $\tilde{\mathbf{X}} \leftarrow \sqrt{\rho}\mathbf{X} + \mathbf{N}, \quad \mathbf{N} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{Z} \leftarrow f_{\boldsymbol{\theta}_{\mathrm{TE}}}(\tilde{\mathbf{X}})$
    $t \sim \mathcal{U}[0, T-1]$
    $\mathbf{X}^t \leftarrow \sqrt{\bar{\alpha}_t}\,\mathbf{Y} + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{Y}_{\text{diff}} \leftarrow \mathrm{Concat}(\mathbf{X}^t, \mathbf{Z})$
    $\hat{\mathbf{Y}} \leftarrow f_{\boldsymbol{\theta}_{\mathrm{G}}}(\mathbf{Y}_{\text{diff}}, t)$
    $\mathcal{L}(\theta) \leftarrow \mathrm{Loss}(\hat{\mathbf{Y}}, \mathbf{Y})$
    Update $\boldsymbol{\theta}$ using gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$
  end for
end for
Return $\boldsymbol{\theta}^*$

C.3.1 AR

AR inference is a sequential prediction strategy in which each future time step is generated conditioned on the previously observed or predicted steps. In this scheme, the model takes a historical sequence, predicts the next step, and then recursively feeds this prediction back to forecast subsequent steps until the desired horizon is reached. The AR inference scheme provides a flexible prediction framework, where a single trained model can operate with arbitrary context lengths and prediction horizons. This flexibility makes it well-suited for sequential modeling tasks, as the model dynamically adapts its predictions based on evolving inputs. The overall scheme of the AR CSI prediction framework is depicted in Fig. 1.

Temporal Encoding: Learning the dynamics of CSI data is a crucial component of the CSI prediction problem. To capture the complex nature of a time-varying wireless channel, we employ a learning model to extract latent features along the temporal dimension. Let the CSI sequence input to the

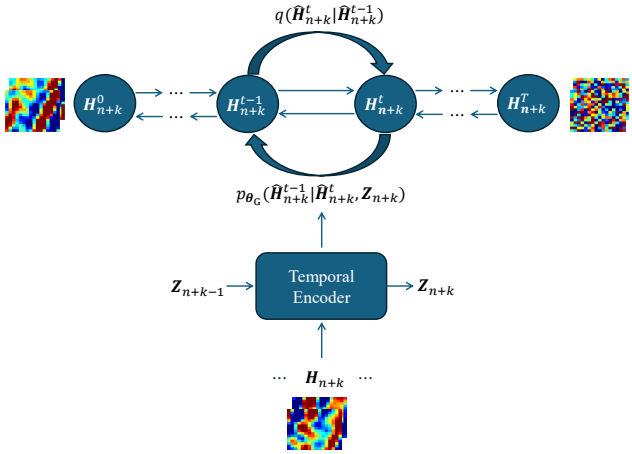


Figure 1: CSI prediction with a diffusion model conditioned on the temporal encoder's latent representation, using an AR inference strategy.

temporal encoder at the k^{th} time step be

$$\mathbf{H}_{n+k} = \{\mathbf{H}_{n-N_{\text{p}}+1:n}, \hat{\mathbf{H}}_{n+1:n+k-1}\}, \tag{C.23}$$

where $\hat{\mathbf{H}}_{n+1:n+k-1}$ denotes the predicted CSI sequence up to step k. The temporal encoder processes this input at the k^{th} step and extracts the corresponding latent temporal representation,

$$\mathbf{Z}_{n+k} = f_{\boldsymbol{\theta}_{\mathbf{TE}}}(\mathbf{H}_{n+k}). \tag{C.24}$$

The resulting latent representation is then used as a conditioning input to the diffusion model for CSI prediction.

Generator: To generate future CSI samples from the latent features extracted by the temporal encoder, we employ a diffusion model. Conditioning on the latent features in the reverse diffusion process steers the generation toward plausible future CSI realizations. The forward process follows the Markov transition in (C.9), where Gaussian noise is progressively added according to a predefined schedule. To approximate the reverse process, an NN

Algorithm C.2 AR Inference

Input: Historical CSI sequence $\mathbf{H}_\mathrm{p}$; trained parameters $\boldsymbol{\theta}^* = [\boldsymbol{\theta}^*_\mathrm{TE}, \boldsymbol{\theta}^*_\mathrm{G}]$; diffusion scheduler with noise schedule $\{\beta_t\}_{t=1}^T$; diffusion steps T; prediction horizon N_f.
Output: Predicted CSI sequence $\{\hat{\mathbf{H}}_n\}_{n=1}^{N_\mathrm{f}}$.

Initialize $\mathbf{H}_c \leftarrow \mathbf{H}_\mathrm{p}$
for $n = 1$ **to** N_f **do**
  $\mathbf{Z} \leftarrow f_{\boldsymbol{\theta}^*_\mathrm{TE}}(\mathbf{H}_\mathrm{c})$
  Initialize $\mathbf{H}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  for $t = T$ **to** 1 **do**
    $\hat{\mathbf{H}}^0 \leftarrow f_{\boldsymbol{\theta}^*_\mathrm{G}}(\mathrm{Concat}(\mathbf{H}^t, \mathbf{Z}), t)$
    $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t) = \frac{\mathbf{H}^t - \sqrt{\bar{\alpha}_t}\,\hat{\mathbf{H}}^0_{\boldsymbol{\theta}}(\mathbf{H}^t, t)}{\sqrt{1-\bar{\alpha}_t}}$
    $\mathbf{H}^{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\,\hat{\mathbf{H}}^0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t)$
  end for
  $\hat{\mathbf{H}}_n \leftarrow \hat{\mathbf{H}}^0$
  $\tilde{\mathbf{H}}_n \leftarrow \hat{\mathbf{H}}_n + \mathbf{N}, \quad \mathbf{N} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$
  $\mathbf{H}_\mathrm{c} \leftarrow \mathrm{Concat}(\mathbf{H}_\mathrm{c}, \tilde{\mathbf{H}}_n)$
end for
Return $\{\hat{\mathbf{H}}_n\}_{n=1}^{N_\mathrm{f}}$

is trained to approximate the conditional distribution

$$f_{\boldsymbol{\theta}_\mathrm{G}}(\hat{\mathbf{H}}^t_{n+k}, \mathbf{Z}_{n+k}) \approx p_{\boldsymbol{\theta}_\mathrm{G}}(\hat{\mathbf{H}}^{t-1}_{n+k} \mid \hat{\mathbf{H}}^t_{n+k}, \mathbf{Z}_{n+k}), \tag{C.25}$$

and after T sampling steps, the predicted CSI sample is

$$\hat{\mathbf{H}}_{n+k} = \hat{\mathbf{H}}^0_{n+k}. \tag{C.26}$$

Inference: In the inference step, after predicting the next CSI frame, the generated sample is appended to the input sequence and fed back into the model to predict the subsequent time step. This procedure is repeated until the desired prediction horizon is reached, as summarized in Algorithm C.2.

C.3.2 Seq2seq

Seq2seq inference is a parallel prediction strategy in which the entire sequence of future steps is generated simultaneously rather than recursively. Instead of

predicting one step at a time and feeding it back into the model, a seq2seq approach maps the input sequence directly to multiple outputs in a single forward pass.

Temporal Encoding: In the seq2seq prediction framework, the temporal encoder processes a fixed-length CSI sequence and extracts a latent representation,

$$\mathbf{Z} = f_{\boldsymbol{\theta}_{\mathrm{TE}}}(\mathbf{H}_{\mathrm{p}}). \tag{C.27}$$

Generator: Conditioned on this latent feature, the generator produces the entire sequence of future CSI samples over a fixed prediction horizon in parallel using a diffusion process. The generator then learns to approximate the reverse process by modeling the conditional distribution

$$f_{\boldsymbol{\theta}_{\mathrm{G}}}(\hat{\mathbf{H}}^t, \mathbf{Z}) \approx p_{\boldsymbol{\theta}_{\mathrm{G}}}(\hat{\mathbf{H}}^{t-1} \mid \hat{\mathbf{H}}^t, \mathbf{Z}), \tag{C.28}$$

and after T sampling steps, the predicted CSI sample is

$$\hat{\mathbf{H}}_{\mathrm{f}} = \hat{\mathbf{H}}^0. \tag{C.29}$$

This design eliminates the sequential dependency of AR inference, reducing inference time and computational overhead, especially for long prediction horizons. By predicting all future steps in parallel, seq2seq inference also avoids the problem of error accumulation, since early mistakes do not propagate through the sequence. However, this efficiency comes at the cost of reduced flexibility as a seq2seq model can be trained for a fixed context length and prediction horizon, and may struggle to capture fine-grained temporal dependencies compared to AR methods.

Inference: During inference, the trained temporal encoder processes the historical CSI samples and encodes them into a temporal latent representation. This representation is concatenated with standard Gaussian noise and passed to the diffusion generator, where the reverse diffusion process is carried out using the DDIM scheduler as outlined in Algorithm C.3.

It is worth noting that various temporal encoders (e.g., RNNs, Transformers) and backbone models for diffusion (e.g., U-Net, DiT) can be employed. In Section C.4, we examine our diffusion-based CSI prediction framework under

Algorithm C.3 Seq2seq Inference

Input: Historical CSI sequence $\mathbf{H}_{\mathrm{p}}$ of length N_{p}; trained parameters $\boldsymbol{\theta}^* = [\boldsymbol{\theta}^*_{\mathrm{TE}}, \boldsymbol{\theta}^*_{\mathrm{G}}]$; diffusion scheduler with noise schedule $\{\beta_t\}_{t=1}^T$; diffusion steps T.
Output: Predicted CSI sequence $\{\hat{\mathbf{H}}_{\mathrm{f}}\}$.

$\mathbf{Z} \leftarrow f_{\boldsymbol{\theta}^*_{\mathrm{TE}}}(\mathbf{H}_{\mathrm{p}})$
Initialize $\mathbf{H}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T$ **to** 1 **do**
  $\hat{\mathbf{H}}^0 \leftarrow f_{\boldsymbol{\theta}^*_{\mathrm{G}}}(\mathrm{Concat}(\mathbf{H}^t, \mathbf{Z}), t)$
  $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t) = \frac{\mathbf{H}^t - \sqrt{\bar{\alpha}_t}\,\hat{\mathbf{H}}^0_{\boldsymbol{\theta}}(\mathbf{H}^t, t)}{\sqrt{1-\bar{\alpha}_t}}$
  $\mathbf{H}^{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\,\hat{\mathbf{H}}^0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{H}^t, t)$
end for
$\hat{\mathbf{H}}_{\mathrm{f}} \leftarrow \hat{\mathbf{H}}^0$
Return $\hat{\mathbf{H}}_{\mathrm{f}}$

different architectural choices.

C.3.3 Unified Seq2seq

Employing separate models for temporal encoding and CSI generation provides a robust framework for learning the temporal and spatial characteristics of the dynamic CSI distribution. Training separate models for temporal and spatial dependencies complicates optimization and requires substantial training data. It also increases computational cost, as both networks must be trained and executed for CSI prediction. To address these limitations and enhance practicality in real-world applications, we integrate temporal encoding and generation into a single diffusion model that jointly learns temporal and spatial dependencies from CSI samples. In this formulation, the diffusion model directly approximates the conditional distribution of the CSI dynamics given historical CSI sequences,

$$p_{\boldsymbol{\theta}_{\mathrm{G}}}\left(\hat{\mathbf{H}}^{t-1} \mid \hat{\mathbf{H}}^t, \mathbf{H}_{\mathrm{p}}\right). \tag{C.30}$$

This unified framework reduces model complexity and improves efficiency, though it may be less effective in capturing fine-grained temporal and spatial structures compared to architectures with two dedicated modules.

C.4 Experimental Setup

In this section, we present a detailed description of the experimental setup used to evaluate the performance of the proposed diffusion-based CSI prediction models. We first introduce the CDL simulation parameters employed to generate the CSI dataset. We then describe the proposed diffusion models with different backbone architectures, including U-Net, DiT, and 3D U-Net, as well as several state-of-the-art CSI prediction baselines for comparison. The training configuration and the techniques adopted during model training are outlined next. Finally, we provide a comprehensive set of numerical results to assess and compare the performance of these models in terms of NMSE across varying prediction horizons and SNR regimes.

C.4.1 Dataset Setup

We generate CSI data according to the CDL channel model introduced in Section C.2.1. A total of 100,000 independent CSI samples are created. Each sample is stored as a tensor of shape $[100 \times 2 \times N_\mathrm{t} \times N_\mathrm{c}]$, representing 100 time steps, 2 channels for the real and imaginary parts, N_t transmit antennas, and N_c subcarriers. Each sample is then split into past CSI data, $\mathbf{H}_\mathrm{p}$ of length N_p, and future ground-truth CSI data, $\mathbf{H}_\mathrm{f}$ of length N_f, to form training pairs $(\mathbf{X}, \mathbf{Y})$ with $\mathbf{X} = \mathbf{H}_\mathrm{p}$ and $\mathbf{Y} = \mathbf{H}_\mathrm{f}$. Min-max scaling is used to scale the dataset in the range $(0, 1)$. Unless otherwise specified, we set $N_\mathrm{p} = 30$ and $N_\mathrm{f} = 10$ throughout the experiments presented in the subsequent sections.

To generate the CSI dataset, we simulate time-varying channels using 3GPP-compliant CDL channel models, following the parameters summarized below:

- Carrier frequency: $f_\mathrm{c} = 28\,\mathrm{GHz}$.
- Channel model: 3GPP CDL channel model with profiles randomly selected from the set {`CDL-A`, `CDL-B`, `CDL-C`, `CDL-D`, `CDL-E`}.
- Antenna configuration: 16×1 MIMO system, with 16 transmit antennas at the base station and 1 receive antenna at the user terminal.
- Bandwidth: 25 resource blocks (RBs) with a subcarrier spacing of 30 kHz, resulting in a total of 300 subcarriers. CSI is extracted from 16 evenly spaced subcarriers across the band.

- OFDM frame: Each CSI sample consists of 100 OFDM symbols (time steps), with each OFDM symbol having a duration of approximately $T_{\text{sym}} \approx 33.3\,\mu\text{s}$.
- Velocity: User velocities are sampled uniformly from the range $[30, 120]$ km/h to cover a variety of mobility scenarios.
- Delay spread: The channel delay spread is sampled uniformly from the range $[50, 400]$ ns.

C.4.2 Neural Prediction Models

We evaluate the performance of several baseline models and our proposed diffusion-based scheme for CSI prediction. These models are selected to cover a diverse range of architectural paradigms, including RNNs such as GRU and ConvLSTM, efficient Transformer variants such as LinFormer [26], and proposed generative diffusion-based predictors. A brief description of each model is provided below:

- **GRU:** We use a GRU model with two hidden layers, each comprising 128 channels and employing the tanh activation function. This serves as a standard sequential baseline for capturing temporal dependencies in CSI prediction.
- **ConvLSTM:** We adopt a single-layer ConvLSTM with a hidden state of 128 channels and a spatial kernel size of 3×3, producing latent feature maps in $\mathbb{R}^{128\times N_{\text{t}}\times N_{\text{c}}}$. The output is followed by Group Normalization (single group), Dropout with a rate of 0.2, a 3×3 convolutional projection layer with 4 output channels, and a final tanh activation. A brief review of ConvLSTM fundamentals is given in Appendix C.A.
- **LinFormer:** LinFormer, introduced in [26], is a Transformer variant tailored for CSI prediction. It replaces the computationally expensive self-attention mechanism with a time-aware multi-layer perceptron (TMLP), significantly reducing complexity. The model comprises six encoder blocks, each with a model dimension of 512 and a feed-forward hidden size of 512.
- **DiU:** DiU represents a diffusion-based CSI prediction framework employing a U-Net backbone with AR inference. A ConvLSTM acts as

the temporal encoder to capture channel dynamics. The U-Net design is detailed in Appendix C.B.

- **DiU-seq2seq:** This is a seq2seq variant of the diffusion framework using the same U-Net backbone as DiU but predicts multiple future CSI frames in a single forward pass, without an explicit temporal encoder.

- **LinFusion:** LinFusion combines a LinFormer-based temporal encoder with a U-Net diffusion generator under a seq2seq inference scheme. The LinFormer and U-Net components follow the same architecture configurations as in the LinFormer and DiU models, respectively.

- **DiT:** In this variant, we employ a DiT as the denoising backbone, while a ConvLSTM serves as the temporal encoder. Architectural details and parameter settings for DiT are provided in Appendix C.C.

- **DiU3:** We employ a 3D U-Net as the diffusion backbone with seq2seq inference. The 3D U-Net extends the standard U-Net into three dimensions, jointly modeling spatio-temporal features. The details for the 3D U-Net design used in our simulation are provided in Appendix C.D.

C.4.3 Training Parameters

For training the diffusion model, we employ the DDIM scheduling technique to accelerate the sampling process. Diffusion timesteps t are drawn uniformly from $\{0, \ldots, T-1\}$. The input to the diffusion generator is formed by concatenating the noisy CSI at timestep t with the latent features extracted from the temporal encoder. To enhance robustness against varying SNR conditions and to account for channel estimation errors during pilot transmission, we corrupt the training data with additive noise.

We adopt the Huber loss [40] to train both the diffusion model and the temporal encoder, and use the diffusion loss formulated in (C.16) to predict clean samples. The Huber loss is a robust regression loss that behaves quadratically for small errors and linearly for large errors, combining the benefits of MSE and mean absolute error (MAE). With a threshold parameter $\delta > 0$, it is defined as

$$\mathcal{L}_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & \text{if } |y - \hat{y}| \leq \delta, \\ \delta\left(|y - \hat{y}| - \frac{\delta}{2}\right), & \text{otherwise.} \end{cases} \tag{C.31}$$

We employ the Adam optimizer to jointly update the parameters of the temporal encoder and the diffusion generator, using learning rates of 1×10^{-3} and 1×10^{-3}, respectively. To enhance training stability, we apply an exponential moving average (EMA) with a decay factor of 0.995 and an update interval of 10 steps, together with gradient clipping at a maximum norm of 1.0. The model is trained for 500 epochs with a batch size of 512.

In the implementation, the Huber loss threshold is set to $\delta = 0.016$, while the SNR values are uniformly sampled from the range $[-20, 20]$ dB. For the diffusion noise schedule, we adopt the scaled and clipped squared–cosine design proposed in [41], defined as

$$\bar{\alpha}_t = \cos^2\left(\frac{\frac{t}{T} + 0.008}{1.008} \cdot \frac{\pi}{2}\right), \tag{C.32}$$

which leads to the variance schedule

$$\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, \quad t \geq 1. \tag{C.33}$$

To prevent degenerate values, the coefficients are further clipped as

$$\beta_t = \min(\beta_{\max},\ \max(\beta_{\min},\ \beta_t)), \tag{C.34}$$

with $\beta_{\min} = 10^{-4}$, $\beta_{\max} = 2 \times 10^{-2}$, and $T = 2000$ in our experiments. Compared to a linear schedule, the squared–cosine design produces smaller β_t values in the early steps, resulting in a slower corruption rate at the start of the forward process, and larger β_t near the end, leading to stronger noise injection in later steps.

C.4.4 Numerical Results

We evaluate the performance of the proposed diffusion-based CSI prediction models and baseline approaches introduced in Section C.4.2 under various experimental settings, including different SNR levels, prediction horizons, and user mobility scenarios. The performance of all methods is quantified using the NMSE, defined as

$$\text{NMSE} \triangleq \mathbb{E}\left[\frac{\|\mathbf{H} - \hat{\mathbf{H}}\|^2}{\|\mathbf{H}\|^2}\right], \tag{C.35}$$

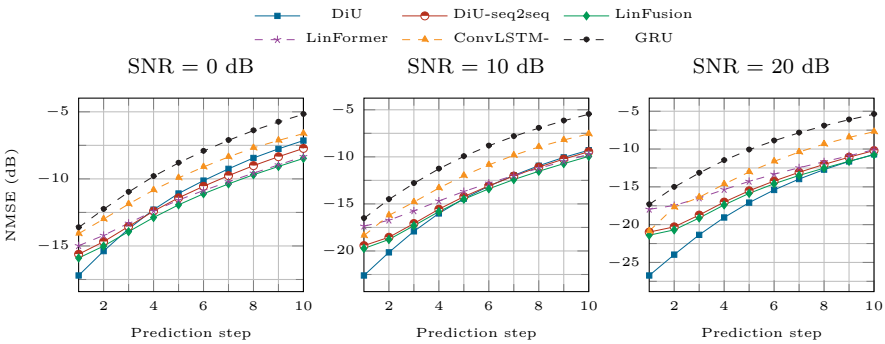


Figure 2: NMSE of different CSI prediction models over varying prediction steps at inference SNRs of 0, 10, and 20 dB.

where $\mathbf{H}$ and $\hat{\mathbf{H}}$ denote the ground-truth and predicted channel matrices, respectively.

Fig. 2 illustrates the prediction accuracy of different models across varying prediction steps under three inference SNR regimes: 0, 10, and 20 dB. The results show that the proposed diffusion models consistently outperform state-of-the-art benchmarks such as GRU, ConvLSTM, and LinFormer. The performance gain is most pronounced at higher SNRs and shorter prediction horizons. For example, at an inference SNR of 20 dB, the diffusion model with AR inference achieves improvements of more than 5 dB and 8 dB in NMSE compared to ConvLSTM and LinFormer, respectively. When comparing the three diffusion-based predictors, the AR variant noticeably outperforms the seq2seq variants. In contrast, the seq2seq diffusion models provide slightly better NMSE performance at lower SNRs and for longer prediction horizons, primarily because of error propagation over time in AR inference. Furthermore, incorporating diffusion into the Linformer architecture yields an NMSE improvement of approximately 4 dB compared to the Linformer baseline. While LinFusion outperforms Linformer alone, the diffusion model using a ConvLSTM temporal encoder and a U-Net backbone consistently achieves the lowest NMSE, particularly for short prediction horizons.

Fig. 3 presents the NMSE performance of different CSI prediction models as a function of inference SNR, evaluated across three prediction horizons:

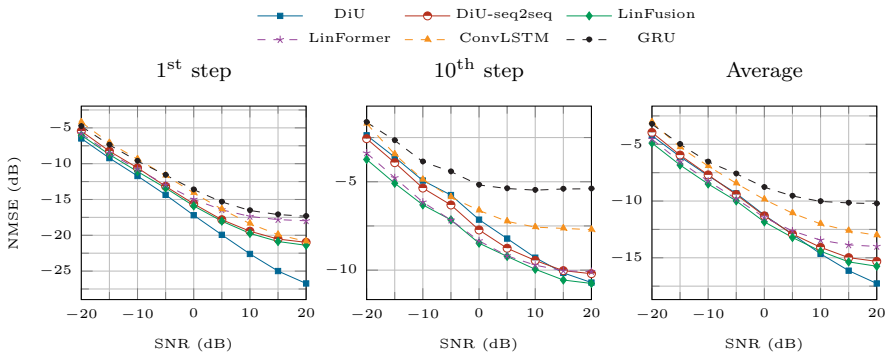


Figure 3: NMSE performance of CSI prediction models versus inference SNR. Results are reported for the first prediction step, the last prediction step, and the average NMSE across all prediction steps.

1^{st}-step prediction, 10^{th}-step prediction, and the average performance over 10 prediction steps. As anticipated, higher SNR levels enhance the performance of all models, with our diffusion-based schemes consistently surpassing the baselines. Specifically, for the average NMSE across all steps, the diffusion model with a U-Net backbone and AR inference achieves the best overall performance. For first-step prediction, DiU with AR inference yields up to 5–8 dB NMSE gains compared to GRU, ConvLSTM, and LinFormer, particularly at high SNRs. At longer prediction horizons, the seq2seq diffusion models surpass AR inference, though at the expense of reduced accuracy for short horizons and lower flexibility in handling variable context lengths and prediction horizons.

We evaluate the prediction accuracy of different models under fixed user mobility levels of 30 km/h, 60 km/h, and 120 km/h, as shown in Fig. 4. As expected, the prediction performance degrades with increasing mobility due to faster channel dynamics and weaker temporal correlations. Nevertheless, the proposed diffusion-based predictors consistently outperform all baseline models across all velocity regimes. The NMSE gains of AR-based diffusion are particularly pronounced at lower user speeds (e.g., 30 km/h), where stronger temporal correlations can be exploited. In this regime and for short prediction horizons, DiU with AR inference achieves up to a 5 dB improvement over

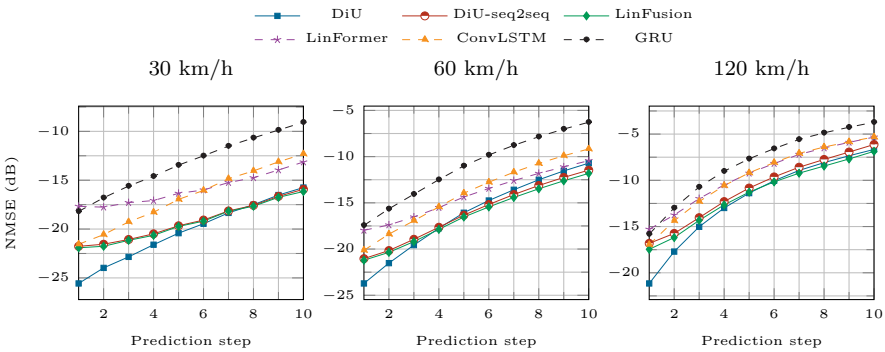


Figure 4: NMSE performance of CSI prediction models as a function of prediction step, evaluated on CSI samples with user mobilities of 30 km/h, 60 km/h, and 120 km/h.

both DiU-seq2seq and LinFusion. However, this performance gap gradually diminishes as the prediction horizon extends and the user velocity increases.

Fig. 5 compares the CSI prediction performance of different backbone architectures and context lengths across varying prediction horizons. The left plot evaluates our proposed diffusion-based predictors with U-Net, DiT, and 3D U-Net backbones. These results demonstrate the flexibility of the diffusion framework, as different architectures can be seamlessly integrated into the generator. While all models achieve comparable performance, the U-Net backbone consistently achieves the lowest NMSE. Moreover, although the 3D U-Net variant applies convolutional operations over the 3D volume of CSI data and has significantly higher model capacity, it does not outperform the 2D U-Net variant combined with an explicit temporal encoder and AR inference. This underscores the effectiveness of our proposed diffusion-based CSI prediction framework with an explicit temporal encoder.

The middle plot compares different variants of the seq2seq inference strategy for diffusion-based CSI prediction for DiU. We evaluate multiple configurations, including 1:1, 2:1, 5:1, 5:2, 10:5, and 30:10, where an N:M setup denotes predicting M future CSI frames from N historical observations. Configurations with fewer than 10 prediction outputs still require recurrence to reach the full prediction horizon, indicating that autoregression is unavoid-

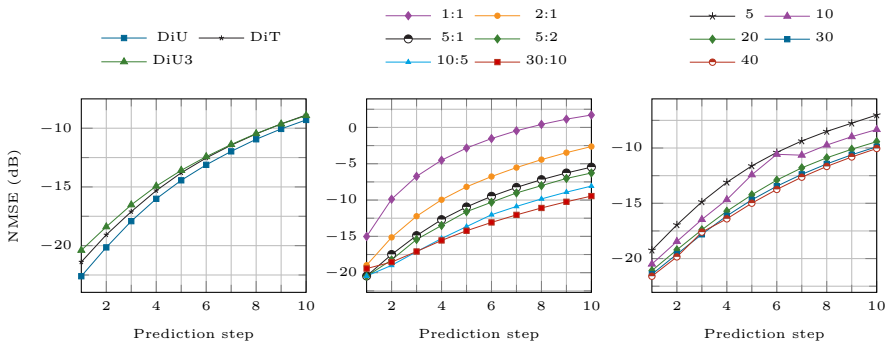


Figure 5: NMSE performance versus prediction step at SNR = 10 dB. The left panel compares DiU, DiT, and DiU3; the middle panel shows different seq2seq variants; and the right panel presents DiU with varying context lengths.

able for many trained seq2seq models. In this setup, 30:10 corresponds to the DiU-seq2seq model. The results show that increasing the context length in seq2seq prediction provides more temporal information and consistently improves NMSE performance.

Finally, the right plot highlights the impact of varying the input context length N_{p} on prediction accuracy of DiU with AR inference. Increasing the context length from $N_{\mathrm{p}} = 5$ to $N_{\mathrm{p}} = 40$ steadily improves NMSE performance, demonstrating that the diffusion model benefits from larger temporal contexts. However, the performance gain begins to saturate beyond $N_{\mathrm{p}} = 30$, indicating diminishing returns when excessively long histories are used.

Fig. 6 illustrates the impact of different sampling steps on the performance of the proposed diffusion-based CSI prediction model at an inference SNR = 10 dB. The left plot shows the NMSE as a function of prediction step for different sampling schedules, while the right panel provides a qualitative visualization of reconstructed CSI frames for one representative sequence. Rows correspond to different sampling step settings, and columns represent future prediction steps.

The results demonstrate that the proposed model achieves competitive performance even with a small number of sampling steps. In particular, using as few as 3 sampling steps achieves nearly the same NMSE as the 100-step schedule, while significantly reducing inference time. Beyond 5 steps, the

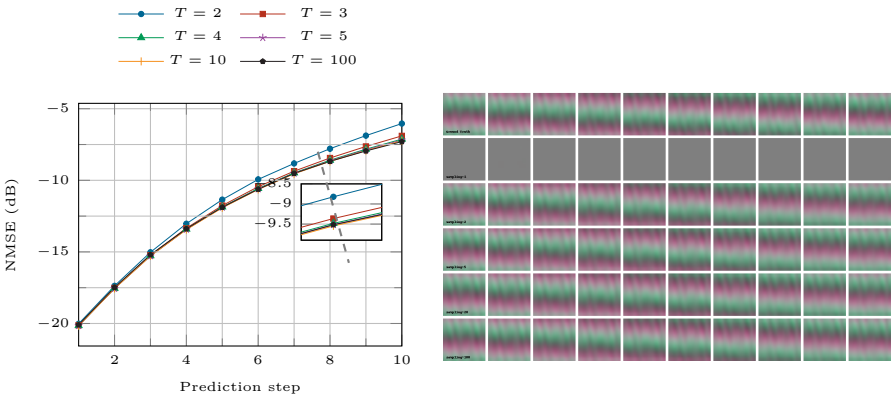


Figure 6: Left: NMSE performance versus prediction step at 10 dB SNR for different sampling steps. Right: qualitative channel visualization for a single sequence, where rows correspond to the ground truth and predictions for each sampling step, and columns indicate the prediction steps.

performance improvements become marginal, indicating that the proposed method can operate efficiently with very few diffusion iterations. The qualitative reconstructions on the right further confirm that high-fidelity predictions can be achieved even under low sampling budgets.

Fig. 7 illustrates the generalization performance of different CSI prediction models when evaluated on a test dataset generated at a carrier frequency of $f_c = 3$ GHz, while all models were trained exclusively on data generated at $f_c = 28$ GHz. This setup induces a noticeable distribution shift in the channel statistics, since lower carrier frequencies exhibit different propagation characteristics and scattering behaviors compared to mmWave frequencies. The results show that the proposed diffusion-based models achieve the lowest generalization error and maintain robust performance under unseen channel statistics. This advantage stems from modeling the full conditional distribution $p(\mathbf{H}_f \mid \mathbf{H}_p)$ rather than learning a single deterministic mapping from past to future CSI. By capturing the underlying spatiotemporal structure of the channel—rather than memorizing training-specific patterns—diffusion-based predictors remain resilient under substantial domain shifts.

The baseline discriminative models suffer from noticeable performance degra-

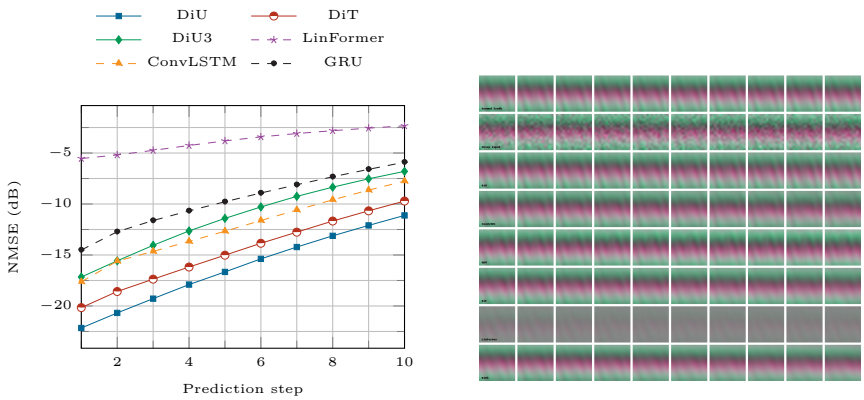


Figure 7: Left: NMSE performance versus prediction step, evaluated at $f_c = 3$ GHz. Right: qualitative channel visualization for a single sequence, where rows correspond to the ground truth and different prediction schemes, and columns represent the prediction steps.

dation under domain shift. LinFormer in particular exhibits a sharp NMSE increase, indicating strong overfitting to the training distribution at $f_c = 28$ GHz when evaluated at $f_c = 3$ GHz. The 3D U-Net diffusion model also exhibits some performance loss due to its relatively high parameter count, which makes it more susceptible to overfitting. By comparison, the ConvLSTM model generalizes better than LinFormer, largely owing to its simpler architecture and lower capacity, which reduce its sensitivity to domain shifts. Overall, these results highlight that diffusion-based CSI predictors—particularly those with lightweight backbone architectures—are substantially more robust to domain shifts, making them especially promising for deployment in real-world wireless systems.

The right plot presents a qualitative comparison of predicted CSI sequences at SNR = 10 dB across six forecasting models: DiU, ConvLSTM, GRU, DiT, LinFormer, and DiU3. The first row shows the ground-truth CSI frames, followed by the noisy input frames in the second row. The subsequent rows present the reconstructed CSI sequences for each prediction model, where each column corresponds to a future prediction step. As observed, LinFormer struggles to recover the spatial structure of the wireless channel due to its poor

Table 1: Comparison of model complexity measured by FLOPs and the number of trainable parameters. Abbreviations: G = billions, M = millions.

Model	FLOPs (G)	Parameters (M)
DiU	10.54	1.60
DiU-seq2seq	0.49	1.03
DiT	6.11	0.70
DiU3	48.32	10.19
LinFusion	1.01	4.60
LinFormer	0.11	3.70
ConvLSTM	1.52	0.15
GRU	0.31	2.10

generalization capability, while the other models are able to reconstruct CSI samples more faithfully across prediction steps. Furthermore, DiU3 produces slightly blurred reconstructions, reflecting its performance loss when evaluated under different channel statistics.

Table 1 reports the computational complexity and parameter counts of the evaluated CSI prediction models. The number of floating-point operations (FLOPs) per forward pass and the total number of learnable parameters are provided for a typical configuration with $N_{\mathrm{p}} = 30$ input frames and $N_{\mathrm{f}} = 10$ predicted CSIs. Among the baseline methods, GRU and LinFormer exhibit the lowest computational cost; however, they require a relatively large number of parameters to maintain representational capacity, making them more susceptible to overfitting. Models based on AR prediction generally incur higher computational costs due to temporal recurrence but offer greater flexibility across different context lengths and prediction horizons. Seq2seq diffusion models—and DiU-seq2seq in particular—enable faster inference and lower per-pass computation, making them well suited for latency-critical applications. In contrast, the diffusion-based predictor with a 3D U-Net backbone incurs substantially higher FLOPs owing to the joint spatio-temporal processing performed over the full 3D CSI volume.

C.5 Conclusion

In this paper, we presented a class of diffusion-based models for CSI prediction, designed to flexibly integrate different time-series forecasting architectures. By progressively learning the underlying distribution of spatial CSI patterns and their temporal evolution, diffusion models capture the inherent uncertainty of wireless channels rather than memorizing a fixed input–output mapping. We considered both designs with and without an explicit temporal encoder: in the former, the temporal encoder extracts latent temporal dynamics from historical CSI, while in the latter, the diffusion generator directly leverages historical CSI samples to predict future frames. Furthermore, we investigated AR and seq2seq inference schemes, explored multiple generator backbones including U-Net, DiT, and 3D U-Net, and incorporated advanced techniques such as DDIM for efficient sampling. Extensive simulations with the CDL channel model in a mmWave MIMO setup demonstrate that the proposed diffusion-based models consistently outperform state-of-the-art benchmarks such as GRU, ConvLSTM, and LinFormer across diverse SNR levels, prediction horizons, and user mobility scenarios. In particular, the diffusion model with a U-Net backbone delivers the best overall performance, achieving up to 5–8 dB NMSE gains for short-term prediction horizons at high SNRs. Moreover, we show that as few as three diffusion sampling steps are sufficient to attain competitive prediction accuracy, and that generative diffusion models exhibit stronger generalization capabilities than discriminative baselines.

C.A ConvLSTM

ConvLSTM extends the standard LSTM architecture by replacing fully connected operations with 2D convolutional operations, enabling it to better preserve local spatial correlations in structured data such as images, videos, or CSI. By sliding convolutional filters across spatial dimensions, ConvLSTM is able to jointly capture spatial features and temporal dependencies more effectively than traditional LSTMs.

Similar to the standard LSTM, ConvLSTM consists of three gates: the input gate, forget gate, and output gate. However, unlike fully connected LSTMs, the computations are performed using convolutions instead of matrix

multiplications. The cell state and hidden state at time step n are updated as follows:

$$\mathbf{Y}_n = [\mathbf{X}_n, \mathbf{Z}_{n-1}], \tag{C.36}$$

$$[\mathbf{i}_n, \mathbf{f}_n, \mathbf{o}_n, \mathbf{g}_n] = \text{Conv2D}(\mathbf{Y}_n), \tag{C.37}$$

$$\mathbf{i}_n = \sigma(\mathbf{i}_n), \quad \mathbf{f}_n = \sigma(\mathbf{f}_n), \quad \mathbf{o}_n = \sigma(\mathbf{o}_n), \tag{C.38}$$

$$\mathbf{g}_n = \tanh(\mathbf{g}_n), \tag{C.39}$$

$$\mathbf{S}_n = \mathbf{f}_n \odot \mathbf{S}_{n-1} + \mathbf{i}_n \odot \mathbf{g}_n, \tag{C.40}$$

$$\mathbf{Z}_n = \mathbf{o}_n \odot \tanh(\mathbf{S}_n), \tag{C.41}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function, $\tanh(\cdot)$ is the hyperbolic tangent function, and $\odot$ represents element-wise multiplication. The operator $\text{Conv2D}(\cdot)$ applies a 2D convolution over the spatial dimensions. Here, $\mathbf{X}_n$ is the input at time step n, $\mathbf{Z}_{n-1}$ is the previous hidden state, and $\mathbf{S}_n$ denotes the current cell state.

C.B U-Net

U-Nets were originally introduced for biomedical image segmentation in [34] and have since become a standard denoising backbone in diffusion-based generative models due to their strong ability to capture hierarchical spatial features. The U-Net architecture follows an encoder–decoder design, where the encoder progressively downsamples the input to extract latent representations, while the decoder upsamples and reconstructs the output using both learned features and skip connections from the encoder.

The network begins with an initial 3×3 convolution that projects the two input channels (real and imaginary CSI components) into 32 feature channels. These features then enter the encoder path. To incorporate temporal conditioning on the diffusion step t, we embed the scalar timestep using sinusoidal positional encodings, followed by a two-layer multilayer perceptron (MLP) that maps the embedding to a 256-dimensional vector. This timestep embedding is injected into every residual block via adaptive conditioning, allowing the denoising process to depend on the current stage of the diffusion process.

The encoder consists of two resolution stages. The first down block is a standard convolutional block operating on 32 channels. It applies two residual

blocks, each conditioned on the timestep embedding, followed by a 2×2 spatial downsampling that reduces the resolution from $N_\mathrm{t} \times N_\mathrm{c}$ to $N_\mathrm{t}/2 \times N_\mathrm{c}/2$. The second down block is an attention-augmented convolutional block that increases the channel width to 64. It applies two residual blocks, each followed by a self-attention layer with a single-head attention mechanism. The outputs of these down blocks are stored and later used as skip connections to improve reconstruction fidelity.

At the bottleneck, the network operates on 64 channels at a spatial resolution of $N_\mathrm{t}/2 \times N_\mathrm{c}/2$. This mid block consists of a residual block, followed by a self-attention layer, and another residual block. These layers are also conditioned on the timestep embedding. Including attention at the bottleneck enables the model to capture global dependencies between antennas and subcarriers.

The first up block is an attention-augmented up block that merges bottleneck features with the skip connection from the second encoder stage, applying three sequences of residual and attention layers. The second up block is a standard up block that fuses features with the first encoder skip connection, applies three residual blocks without attention, and upsamples back to the original $N_\mathrm{t} \times N_\mathrm{c}$ spatial resolution.

Throughout the network, the residual blocks use the Sigmoid Linear Unit (SiLU) activation function. Specifically, every residual block inside the encoder, bottleneck, and decoder applies the sequence: Group Normalization $\rightarrow$ SiLU activation $\rightarrow$ Convolution. Additionally, after the final decoder stage, the features pass through a group normalization layer and another SiLU activation before the concluding 3×3 convolution, where no activation function is applied after the final convolution.

C.C DiT

DiT adopts a Vision Transformer (ViT)-style architecture and incorporates timestep conditioning via adaptive LayerNorm-Zero (adaLN-Zero). The overall architecture includes patch embedding, positional embedding, timestep embedding, and a Transformer block with specialized adaptive layer norm modulation. In particular DiT is similar to a ViT design that processes 2D images as a sequence of patch embedding with learnable positional encoding and stacking Transformer encoder blocks. DiT further applies embedding to

diffusion timesteps using sinusoidal functions and an MLP. It also considers the conditioning input to every Transformer block using adaLN-Zero.

Let the noisy input at diffusion step t be denoted as $\tilde{\mathbf{H}}_t \in \mathbb{R}^{2\times N_\mathrm{t}\times N_\mathrm{c}}$. The model operates on non-overlapping patches of size $P\times P$ with $P = 4$. A strided convolution with kernel and stride equal to P performs the patchification and per-patch projection, yielding a sequence of tokens $\mathbf{Z}_0 \in \mathbb{R}^{T\times D}$, where $T = (N_\mathrm{t}/P)\cdot(N_\mathrm{c}/P)$ is the number of patches and $D = 128$ is the hidden size. To preserve spatial arrangement after flattening, a learnable positional embedding $\mathbf{P} \in \mathbb{R}^{1\times T\times D}$ is added to the patch embeddings, giving $\mathbf{X}_0 = \mathbf{Z}_0 + \mathbf{P}$.

The scalar diffusion index t is mapped into a high-dimensional representation using sinusoidal functions $\boldsymbol{\phi}(t) \in \mathbb{R}^F$ with $F = 256$. This is subsequently projected through a two-layer MLP with SiLU activation to the model width, producing the timestep embedding $\mathbf{c} \in \mathbb{R}^{B\times D}$. This embedding conditions every block through adaLN-Zero.

The Transformer stack is composed of $L = 8$ DiT blocks. Each block first processes the conditioning vector $\mathbf{c}$ using a modulation with a single-layer MLP and SiLU activation to produce six sets of vectors: scale, shift, and gate parameters for both the attention and MLP sublayers. Specifically, we obtain $\big(\boldsymbol{\delta}^{\mathrm{msa}}, \boldsymbol{\gamma}^{\mathrm{msa}}, \mathbf{g}^{\mathrm{msa}}, \boldsymbol{\delta}^{\mathrm{mlp}}, \boldsymbol{\gamma}^{\mathrm{mlp}}, \mathbf{g}^{\mathrm{mlp}}\big)$, all in $\mathbb{R}^{B\times D}$. Given input tokens $\mathbf{X}$, the modulation is applied to LayerNorm outputs as

$$\mathrm{AdaLN}(\mathbf{X}; \boldsymbol{\delta}, \boldsymbol{\gamma}) = \mathrm{LN}(\mathbf{X}) \odot (1 + \boldsymbol{\gamma}[:, \mathrm{None}, :]) + \boldsymbol{\delta}[:, \mathrm{None}, :],$$

where broadcasting is over the token dimension. The block then computes

$$\mathbf{U} = \mathrm{MHSA}(\mathrm{AdaLN}(\mathbf{X}; \boldsymbol{\delta}^{\mathrm{msa}}, \boldsymbol{\gamma}^{\mathrm{msa}}))\,, \tag{C.42}$$

$$\mathbf{X} \leftarrow \mathbf{X} + \mathbf{g}^{\mathrm{msa}}[:, \mathrm{None}, :] \odot \mathbf{U}, \tag{C.43}$$

$$\mathbf{V} = \mathrm{MLP}\big(\mathrm{AdaLN}(\mathbf{X}; \boldsymbol{\delta}^{\mathrm{mlp}}, \boldsymbol{\gamma}^{\mathrm{mlp}})\big)\,, \tag{C.44}$$

$$\mathbf{X} \leftarrow \mathbf{X} + \mathbf{g}^{\mathrm{mlp}}[:, \mathrm{None}, :] \odot \mathbf{V}. \tag{C.45}$$

The multi-head self-attention employs $H_a = 8$ heads, so each head has width $D/H_a = 16$. The feed-forward MLP expands the hidden dimension to $\rho D = 256$ with Gaussian Error Linear Units (GELU) nonlinearity before projecting back to D.

After the final block, another adaLN modulation and a linear projection with a single-layer MLP and SiLU activation map each token to $2P^2$ output

values, corresponding to a patch of size $P \times P$ with C channels. An inverse patching step reconstructs the spatial tensor $\hat{\mathbf{H}} \in \mathbb{R}^{2\times N_t \times N_c}$.

To ensure stable training under high-noise conditions, the weights producing the modulation vectors and residual gates, as well as the final linear projection, are initialized to zero. Consequently, the model behaves as an approximate identity function at initialization. The adopted configuration uses $P = 4$, $D = 128$, $L = 8$, $H_a = 8$, and an MLP expansion ratio of $\rho = 2.0$, which balances global modeling capacity with computational efficiency for CSI prediction tasks.

C.D 3D U-Net

We extend the U-Net architecture to a 3D variant in order to model the spatio-temporal structure of CSI volumes. The network takes as input a stack of frames consisting of both the observed past and the future frames concatenated along the temporal dimension, and augments the channel dimension with a binary mask channel (1 for past frames and 0 for future frames) to distinguish between conditioning and prediction targets.

The diffusion timestep t is embedded using sinusoidal position encodings followed by a two-layer MLP that projects the embedding to a 256-dimensional vector. This embedding is injected into every residual block to enable adaptive conditioning on the current stage of the diffusion process.

The encoder begins with an initial 3D convolutional block that projects the (2+1) input channels (real, imaginary, and mask) into 32 feature channels. It then applies three downsampling stages with output widths of 64, 128, and 256, respectively. Each 3D convolutional block contains two 3×3×3 convolutions, each followed by Group Normalization and SiLU activations, with the timestep embedding added between the two convolutions. After each stage, the spatial resolution is reduced by half via maxpooling, while preserving the temporal dimension. Skip connections are extracted from the outputs of each 3D convolutional block before pooling for use in the decoder.

At the bottleneck, the network applies another 3D convolutional block with 256 channels, followed by a self-attention layer that performs multi-head self-attention only along the temporal axis while treating each spatial location independently. This allows the model to explicitly capture temporal dependencies between CSI frames while keeping the computational cost independent

of the spatial size.

The decoder mirrors the encoder and consists of three upsampling stages. Each block begins with a 3D transposed convolutional block layer that upsamples only the spatial dimensions, concatenates the result with the corresponding skip connection, and processes the combined features to fuse spatial and temporal information. The channel dimensions are reduced progressively from $256 \rightarrow 128 \rightarrow 64 \rightarrow 32$, while the temporal resolution remains constant.

Finally, a $1\times1\times1$ convolution maps the 32 feature channels to the two output channels representing the real and imaginary parts of the predicted future CSI. Since the input volume includes both past and future frames, only the last frames of the output are retained to produce predictions.

Within every 3D convolutional block, the operations are applied in the following order: Convolution $\rightarrow$ Group Normalization $\rightarrow$ time-embedding addition $\rightarrow$ SiLU $\rightarrow$ Convolution $\rightarrow$ Group Normalization $\rightarrow$ SiLU. Thus, the SiLU activation is consistently used in all residual blocks across the encoder, bottleneck, and decoder. The multi-head self-attention layer does not use any activation internally and relies on the surrounding residual blocks for nonlinearity. No activation is applied after the final convolution to allow the network to produce unconstrained outputs.

References

[1] E. Biglieri, R. Calderbank, A. Constantinides, A. Goldsmith, A. Paulraj, and H. V. Poor, *MIMO Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[2] B. Hassibi and B. M. Hochwald, "How much training is needed in multiple-antenna wireless links?" *IEEE Trans. Inf. Theory*, vol. 49, no. 4, pp. 951–963, 2003.

[3] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[4] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 831–846, 2014.

[5] M. Sternad, T. Svensson, T. Ottosson, A. Ahlén, A. Svensson, and A. Brunstrom, "Towards systems beyond 3G based on adaptive OFDMA transmission," *Proc. IEEE*, vol. 95, no. 12, pp. 2432–2455, 2007.

[6] H. Yin, H. Wang, Y. Liu, and D. Gesbert, "Addressing the curse of mobility in massive MIMO with prony-based angular-delay domain channel predictions," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2903–2917, 2020.

[7] H. Kim, S. Kim, H. Lee, C. Jang, Y. Choi, and J. Choi, "Massive MIMO channel prediction: Kalman filtering vs. machine learning," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 518–528, 2021.

[8] I. C. Wong and B. L. Evans, "Joint channel estimation and prediction for OFDM systems," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, 2005, 5 pp.–2259.

[9] L. S. Muppirisetty, T. Svensson, and H. Wymeersch, "Spatial wireless channel prediction under location uncertainty," *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 1031–1044, 2016.

[10] H. Guo, B. Makki, D.-T. Phan-Huy, E. Dahlman, M.-S. Alouini, and T. Svensson, "Predictor antenna: A technique to boost the performance of moving relays," *IEEE Commun. Mag.*, vol. 59, no. 7, pp. 80–86, 2021.

[11] K. E. Baddour and N. C. Beaulieu, "Autoregressive modeling for fading channel simulation," *IEEE Trans. Wireless Commun.*, vol. 4, no. 4, pp. 1650–1662, 2005.

[12] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018, ISSN: 0377-2217.

[13] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.

[14] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2018, pp. 3634–3640.

[15] C. Meijer and L. Y. Chen, *The rise of diffusion models in time-series forecasting*, arXiv:2401.03006, 2024.

[16] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, *Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting*, arXiv:2101.12072, 2021.

[17] C. Wu, X. Yi, Y. Zhu, W. Wang, L. You, and X. Gao, "Channel Prediction in High-Mobility Massive MIMO: From Spatio-Temporal Autoregression to Deep Learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 1915–1930, 2021.

[18] O. Stenhammar, G. Fodor, and C. Fischione, "A Comparison of Neural Networks for Wireless Channel Prediction," *IEEE Wireless Communications*, vol. 31, no. 3, pp. 235–241, 2024.

[19] G. Liu, Z. Hu, L. Wang, J. Xue, H. Yin, and D. Gesbert, "Spatio-Temporal Neural Network for Channel Prediction in Massive MIMO-OFDM Systems," *IEEE Transactions on Communications*, vol. 70, no. 12, pp. 8003–8016, 2022.

[20] W. Jiang and H. D. Schotten, "Recurrent Neural Network-Based Frequency-Domain Channel Prediction for Wideband Communications," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–6.

[21] W. Jiang and H. D. Schotten, "Deep Learning for Fading Channel Prediction," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 320–332, 2020.

[22] S. Fan, Z. Liu, X. Gu, and H. Li, "Csi-LLM: A Novel Downlink Channel Prediction Method Aligned with LLM Pre-Training," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 1–6.

[23] H. Kim, J. Choi, and D. J. Love, *Machine Learning for Future Wireless Communications: Channel Prediction Perspectives*, 2025.

[24] W. Xu, J. An, Y. Xu, C. Huang, L. Gan, and C. Yuen, "Time-Varying Channel Prediction for RIS-Assisted MU-MISO Networks via Deep Learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1802–1815, 2022.

[25] H. Jiang, M. Cui, D. W. K. Ng, and L. Dai, "Accurate Channel Prediction Based on Transformer: Making Mobility Negligible," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 9, pp. 2717–2732, 2022.

[26] Y. Jin, Y. Wu, Y. Gao, S. Zhang, S. Xu, and C.-X. Wang, "LinFormer: A Linear-based Lightweight Transformer Architecture For Time-Aware MIMO Channel Prediction," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2025.

[27] C. Jiang, J. Guo, C.-K. Wen, and S. Jin, "Enhancing Reliability in AI-Based CSI Prediction: A Proxy-Based Performance Monitoring Approach," *IEEE Transactions on Communications*, vol. 73, no. 4, pp. 2602–2615, 2025.

[28] M. Akrout, F. Bellili, A. Mezghani, and R. W. Heath, *Next-slot OFDM-CSI prediction: Multi-head self-attention or state space model?* 2024.

[29] V. Rizzello, B. Böck, M. Joham, and W. Utschick, "Reverse ordering techniques for attention-based channel prediction," *IEEE Open J. Signal Process.*, vol. 5, pp. 248–256, 2024.

[30] Z. Zhao, F. Meng, Z. Lyu, H. Li, X. Li, and G. Zhu, *CSI-BERT2: A BERT-inspired framework for efficient CSI prediction and classification in wireless communication and sensing*, arXiv:2412.06861, 2025.

[31] S. Mourya, P. Reddy, S. Amuru, and K. K. Kuchi, "Spectral temporal graph neural network for massive MIMO CSI prediction," *IEEE Wireless Commun. Lett.*, vol. 13, no. 5, pp. 1399–1403, 2024.

[32] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 6840–6851.

[33] J. Song, C. Meng, and S. Ermon, *Denoising diffusion implicit models*, arXiv:2010.02502, 2022.

[34] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional networks for biomedical image segmentation*, arXiv:1505.04597, 2015.

[35] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning dense volumetric segmentation from sparse annotation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv. (MICCAI)*, 2016, pp. 424–432.

[36] W. Peebles and S. Xie, *Scalable diffusion models with transformers*, arXiv:2212.09748, 2023.

[37] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo, *Convolutional LSTM network: A machine learning approach for precipitation nowcasting*, arXiv:1506.04214, 2015.

[38] "Study on channel model for frequencies from 0.5 to 100 GHz," 3GPP, Tech. Rep. TR 38.901, version 17.0.0, Mar. 2022, Available: `https://www.3gpp.org/ftp/Specs/archive/38_series/38.901/`.

[39] B. Fesl, B. Böck, F. Strasser, M. Baur, M. Joham, and W. Utschick, *On the asymptotic mean square error optimality of diffusion models*, arXiv:2403.02957, 2025.

[40] P. J. Huber, "Robust estimation of a location parameter," *Ann. Math. Statist.*, vol. 35, no. 1, pp. 73–101, 1964.

[41] A. Nichol and P. Dhariwal, *Improved denoising diffusion probabilistic models*, arXiv:2102.09672, 2021.