POLITECNICO DI TORINO

Master's Degree in Communications Engineering

Master's Degree Thesis

Interpretation of Learned Solutions for Precoding-Oriented Massive MIMO CSI Feedback Design

Supervisors

Prof. Giorgio TARICCO

Prof. Natasha DEVROYE

Candidate

Roberta BUCCHIGNANI

December 2025

Summary

The development of effective channel feedback strategies is crucial for enabling the widespread adoption of Frequency Division Duplexing in Massive MIMO systems. While conventional methods rely on Compressed Sensing or Codebook-based techniques, Deep Learning has recently emerged as a powerful paradigm for this challenge. End-to-end (E2E) models, such as the framework proposed by Carpi et al. [ICC 2023], have demonstrated superior performance by jointly learning the Pilot Signals, the User's Encoder, and the Base Station's Decoder. Adopting a "task-oriented approach", their objective maximizes the downlink sum-rate while minimizing a differentiable overhead penalty (controlled by a trade-off parameter λ). Crucially, their results demonstrate that this strategy achieves near-optimal performance with significantly compressed feedback, identifying an efficient operating regime well below the requirements of traditional reconstruction-based methods, which necessitate high-fidelity channel estimates. However, despite these performance gains, these models operate as opaque "black boxes," leaving their learned internal strategy unknown.

This thesis provides an original interpretability study to "open" this black box. As the authors' code was not public, we first successfully replicated the E2E pipeline from Carpi et al. [ICC 2023], using this validated model as the faithful baseline for our analysis.

We apply a novel methodological framework, combining Explainable AI techniques, such as Quantitative Density-Based Clustering, Dimensionality Reduction, and Contingency Analysis, to examine the system's internal strategy.

Specifically, our analysis investigates the three fundamental learned stages of the pipeline:

1. The Downlink Pilot Signals
2. The User's latent vector, a compressed representation extracted by the Encoder from the noisy, channel-distorted signals

3. The final precoder vector, generated by the Base Station's network to execute the downlink transmission

We demonstrate that the network's internal strategy is not fixed but emerges as a direct function of the rate-overhead trade-off, effectively switching between two distinct operational modes:

- A Compression-Oriented Regime: Low values of λ (e.g., $\lambda = 2$), where priority is given to compression efficiency (minimizing overhead) over rate maximization.

- A Performance-Oriented Regime: High values of λ (e.g., $\lambda = 100$), where priority is given to the maximization of the sum achievable rate.

In the high-rate regime ($\lambda = 100$), the model aims for a faithful reconstruction of the channel to maximize the sum-rate. Ideally suited for channel estimation, it learns to generate orthogonal pilots at the sensing layer while focusing its energy on the specific channel sector. Here, the latent representations act as continuous, high-fidelity maps of the channel information, a characteristic mirrored in the continuous distribution of the precoders. Both these two continuous representations prove not to be clusterable.

Conversely, in the balanced regime ($\lambda = 2$), the model prioritizes meaning over form. Forced to prioritize efficiency, it abandons orthogonality to evolve towards non-orthogonal, task-oriented pilots: the model learns to retain only task-useful information while discarding the rest. In this state, the Base Station outputs a finite set of clustered precoding vectors. Our clustering analysis quantifies this structure, identifying 11 stable prototypes for the user's latent space and 12-18 prototypes for the precoder. By decoding the mapping from the latent clusters to the precoder clusters, we uncover the Base Station's learned policy: for simple channel states, it adopts an efficient "lookup table" (a deterministic mapping between the input latent and the output precoder). However, for ambiguous, high-interference scenarios, it dynamically switches to a context-aware policy (a “one-to-many" scenario).

We provide twofold proof that this is a learned strategy, not an inherent data property: firstly, the structure completely dissolves as the compression constraint is relaxed (at high λ); secondly, it is entirely absent in the original, continuous, and 'non-clusterable' input channel. This demonstrates that the end-to-end system has learned to “tessellate” the continuous physical space: it partitions the non-clusterable channel manifold into discrete decision regions based on a rule of semantic proximity, mapping physically similar channels to the same prototype.

Ultimately, this work demonstrates that E2E models are not inscrutable black boxes: it is possible to develop interpretations of the underlying learned functions. By unveiling these emergent behaviors, this thesis provides a new XAI-driven pathway for understanding, trusting, and validating learned communication systems.

Table of Contents

List of Tables

List of Figures

Acronyms

BS

Base Station

UE

User Equipment

TDD

Time-Division Duplexing

FDD

Frequency-Division Duplexing

PMI

Precoding Matrix Index

CSI

Channel State Information

CS

Compressive Sensing

DL

Deep Learning

AE

Auto Encoder

AoD

Angle of Departure

AWGN	Additive White Gaussian Noise
i.i.d.	Independent and Identically Distributed
MLP	Multilayer Perceptron
NN	Neural Network
E2E	End-to-End
MSE	Mean Squared Error

Chapter 1

Introduction

Wireless communication has emerged as one of the defining technological successes of recent decades, and its demand continues to accelerate, driven by new applications such as augmented reality and the Internet of Things. In this context, Massive MIMO has become a key enabler for achieving significant increases in spectral efficiency. In its standard form, a base station (BS) equipped with an array of N_t active antennas serves K single-antenna users simultaneously over the same time-frequency resources [1].

Traditionally, Massive MIMO operates in time-division duplexing (TDD) mode. In TDD, the uplink and downlink share the same frequency band but are divided into alternating time slots, allowing for bidirectional communication over a single band. By leveraging the reciprocity of the physical propagation channels under TDD, the BS estimates uplink channels through pilots sent by the user equipment (UE), and then reuses this acquired channel state information (CSI) for both downlink precoding and uplink combining. Since each of the K users transmits a mutually orthogonal uplink pilot within the coherence block, the cost of acquiring CSI is K symbols and does not depend on the number of N_t BS antennas [1]. As a result, the pilot overhead scales with K rather than N_t, preserving scalability as the antenna array grows. Both estimation and payload transmission must fit within a coherence block of $\tau = B_c T_c$ transmission symbols, where B_c denotes the coherence bandwidth (in Hz) and T_c the coherence time (in seconds). Over these τ uses the channel can be regarded as approximately constant, ensuring that the acquired CSI remains valid for data transmission [1].

Currently, many wireless networks utilize frequency-division duplexing (FDD). In FDD, the uplink and downlink operate on different frequency bands, which means that channel reciprocity does not apply. Users estimate their downlink channels through local channel estimators and send the estimated CSI back to the BS, consuming both spectrum and energy. A basic FDD scheme requires approximately N_t downlink pilots per coherence block, K uplink pilots and a CSI feedback of N_t coefficients from each of the K users. Consequently, the non-data overhead scales with the product of the number BS antennas and the number of UEs. Since a coherence block contains only τ symbols, this overhead must fit within the block. Therefore, while TDD can support arbitrarily large values of N_t, FDD imposes a trade-off between N_t and K, making it typically viable only in low-mobility and low-frequency environments [1].

As the number of N_t BS antennas increases, the downlink channel seen by each UE becomes an N_t-dimensional vector (one complex coefficient per antenna), which implies a proportional increase in the amount of CSI that must be acquired and reported. Consequently, it is essential to transmit a compressed representation of the full CSI (or task-sufficient features) over the rate-limited link instead of raw coefficients [2]. This challenge is especially pronounced in FDD, as downlink training and uplink feedback scale with both N_t and K, limiting the feasible operating points within a finite coherence block [1].

Thus, developing effective channel feedback strategies is crucial for enabling the widespread adoption of FDD in Massive MIMO systems.

1.1 Classical CSI compression techniques

In 5G New Radio (5G NR), CSI feedback is set up by the Radio Resource Control (RRC) layer: the BS uses Information Elements (IEs) defined in 3GPP TS 38.331 (most notably CSI-ReportConfig, which references to CSI-ResourceConfig and carries codebookConfig) to instruct the User Equipment (UE) what to measure and how to report it. The physical-layer content of the codebooks and the precoding matrix index (PMI) mapping is specified in TS 38.214, while the CSI-RS signals used for channel measurements are defined in TS 38.211 [3], [4], [5]. Within this framework, NR supports two families of downlink codebooks for CSI feedback: Type-I and Type-II. Type-I is the simpler, single-user oriented option: the UE selects a precoding matrix from a predefined table and reports a small set of indices (PMI fields), keeping feedback short (low overhead) at the expense of angular resolution. Type-II enables multi-user scheduling and higher spatial precision through the use of oversampling: the UE reports multiple beams (PMIs) together with per-beam coefficients (e.g. amplitudes/phases) according to the standard's format. This increases not only spatial precision but also the number of bits to report, leading to an overhead growth with the array size; this issue becomes acute with large antenna arrays in Massive MIMO [4]. From a system-design perspective, the accuracy-overhead trade-off of standardized codebooks motivates compression: as N_t grows, the amount of Type-II CSI needed can become burdensome if fine spatial detail is needed; both academia and industry are thus exploring compressed CSI and learning-based feedback beyond fixed codebooks [2].

As introduced by Sohrabi et al. [6], most classical schemes for multi-user FDD Massive MIMO with limited feedback fall into two families: methods that exploit spatial/temporal correlation and sparsity (compressed sensing, CS) [7], and codebook/vector-quantization-based [8] PMI selection.

1.1.1 Codebook-based methods

A wide range of studies relies on finite codebook-based precoding configured a priori at the BS/UE. As summarized by Love et al. [8], the BS probes candidate precoders and each UE feeds back the set of indices and quality metrics (SNRs) of the top-p strongest received signals. The BS selects the downlink precoder from the codebook based on these reports. A DFT-based codebook is a common choice [9]. In this framework, the operating point depends critically on p: with $p = 1$ the scheme effectively reduces to MRT (there is no mechanism to control inter-user interference) [9][10], while larger p enables interference management but inflates per-UE feedback, since there are more indices and metrics to report [11].

Dietl et al. [12] provide an in-depth comparison between CSI-feedback precoding and codebook-based precoding in rich-scattering scenarios. When the available feedback is very low, CSI feedback is preferred: with only a few bits, picking indices from a finite codebook cannot provide beams with a sufficient angular resolution, whereas feeding back, even coarsely quantized, CSI enables linear precoders (MRT/ZF) that better exploit the channel. This gap becomes even larger in massive MIMO with limited scattering: CSI-feedback pipelines can estimate and quantize a few sparse path parameters and thus use very few pilots and few feedback bits to describe the dominant propagation. Codebook schemes, in contrast, need a codebook size that scales with the number of antennas N_t to keep angular resolution: the overhead becomes unmanageable for large systems. Given these drawbacks, subsequent comparisons in this work focus on CSI-feedback-based precoding methods, which are more efficient under tight feedback budgets and large arrays with limited scattering. This approach is consistent with the analysis in [6], which similarly concludes that codebook overhead scales unmanageably in massive MIMO and, for this reason, also limits its own comparison to CSI-feedback schemes.

1.1.2 Compressed-sensing (CS)-based methods

This second family of methods operates on a different principle. Instead of fixed codebooks, CS-based techniques exploit the underlying physics of the channel: it enables the capturing and reconstruction of a signal using significantly fewer measurements. The operation of CS relies on a key assumption, i.e. that the signal of interest is "sparse". A signal is sparse if the vast majority of its information is contained within a very small number of "non-zero" coefficients. The CS technique acquires only a few measurements and then uses an optimization algorithm to find the sparsest possible signal that matches those measurements. Often, a signal is not sparse in its native domain (e.g., in time domain), but becomes sparse when represented in a different domain (e.g., the frequency domain via Fourier transform).

1.2 Deep Learning-based CSI compression techniques

Machine learning, especially deep neural networks (DNNs) has become a powerful tool for high-dimensional, non-convex design in wireless systems. In FDD Massive MIMO, Deep Learning (DL) frameworks learn compact representations of downlink CSI and direct mapping to precoders, enabling end-to-end optimization across pilot design, UE-side compression and BS-side processing. Three complementary strands have emerged that, together, explain why DL solutions often outperform classical CS methods despite relying on fewer modeling assumptions. These strands are: reconstruction-oriented autoencoders (AEs), rate-distortion models, and task-oriented approaches. Across these settings, learning-based CSI feedback has shown strong potential [13][6].

1.2.1 Reconstruction-oriented Autoencoders

First, reconstruction-oriented AEs cast CSI compression as learned source coding: a UE-side encoder f_θ maps the channel observation into a low-rate latent t_k; a quantizer and an entropy coder map t_k to a bitstream, and a BS-side decoder g_ϕ reconstructs the channel $\hat{\mathbf{H}}$ by minimizing a distortion metric (typically the Normalized Mean Squared Error) at a given feedback budget. This AE paradigm, introduced for CSI by CsiNet [14], is explicitly cited by Carpi et al. [13] as the first demonstration that an AE can compress CSI effectively in massive MIMO; subsequent surveys and follow-ups report improved distortion at increasing compression ratios [15].

1.2.2 Rate-Distortion methods

A second line of work concerns rate-distortion-optimized DL with learned entropy models. Borrowing the toolchain from neural image compression, a UE-side encoder produces a latent vector that is quantized (with a differentiable surrogate during training). To optimize the overhead, this vector is entropy-coded: this requires a learned entropy model, which is typically a neural sub-network trained jointly with the AE to estimate the probability distribution of the latent vector ($p(\mathbf{t}_k)$). In this framework, both components are learned: the AE learns the compression (mapping the channel to an efficient latent vector), while the entropy model learns its probability distribution. This learned distribution is then used by a standard entropy coder (e.g., arithmetic coding) to generate the bit stream. During training, the entropy calculated from this model ($\mathbb{E}[-\log_2 p(\mathbf{t}_k)]$) serves as a differentiable loss term, acting as a proxy for the final overhead. The encoder, decoder, and entropy model are trained end-to-end with a rate-distortion objective that trades the reconstruction error for the feedback bits. Carpi [13] explicitly adopts this framework. The overhead term in their loss is calculated by applying the learned entropy model to the "pseudo-quantized latents". This term simply refers to the output of the differentiable quantization surrogate (e.g., additive uniform noise) used during training. The canonical tools for this approach were introduced by Balle' et al. [16], known as the "Entropy Bottleneck". [1]

Along the same line, DeepCMC by Mashhadi et al. [17] proposes a fully-convolutional AE for CSI feedback that integrates quantization and entropy coding in the architecture and is trained with a weighted rate-distortion loss, thereby enabling an explicit trade-off between CSI quality and feedback overhead. The decoder uses residual layers to improve reconstruction, and a distributed variant jointly decodes compressed CSI from multiple UEs to exploit inter-user correlation. Compared with earlier AE baselines, DeepCMC achieves lower NMSE between the original and reconstructed channel matrices at the same bit-rate, but, as with most rate-distortion designs, the objective remains reconstruction quality, not the downlink precoding task itself.

1.2.3 Task-oriented methods

A third distinct task-oriented branch optimizes the BS task itself (linear precoding) rather than channel NMSE, optimizing sum-rate performance directly and thus outperforming reconstruction-oriented pipelines at the same feedback budget. In the multi-user setting of Sohrabi et al. [6], DL is used end-to-end across the whole chain: the BS learns the downlink pilots; each UE runs a neural encoder that maps its pilot observations to a fixed-length binary vector ("binary taps"); BS-side network directly outputs the linear precoders. Channel estimation is not handled as a stand-alone module; training maximizes the downlink sum-rate, thereby avoiding

[1]The Entropy Bottleneck is often implemented with a "factorized learned prior", which assumes that all elements of the latent vector $\mathbf{t}_k$ are statistically independent (i.e., $p(\mathbf{t}_k) = \prod_i p(t_{k,i})$). The model thus learns a separate probability distribution for each latent element, and the total entropy (the overhead term in the loss) becomes the sum of the individual element-wise entropies ($\log(\prod p_i) = \sum \log(p_i)$).

explicit channel reconstruction, and the feedback cost (overhead) is fixed by the chosen feedback dimension (and it cannot be further compressed via entropy coding). As noted in [6], many prior DL works either study single-user scenarios with no inter-user interference or focus on CSI reconstruction at the BS assuming perfect CSI at the UE. In contrast, they handle the multi-user case where each UE only senses and feeds back its own channel, while precoding is a function of all users' channels; secondly, they train end-to-end (including pilot sequences) to directly enhance downlink spectral efficiency. In addition, they discuss generalizability with respect to feedback rate and number of users, proposing two-step procedures to operate over varying bit budgets B and user counts K without retraining the UE encoders from scratch. In single-user massive MIMO, Chen et al. [18] propose implicit CSI feedback: DL is used on both sides to learn what to send and how to interpret it for beamforming. A UE-side encoder network compresses the pilot observations into a compact code containing precoding-sufficient features (rather than a full channel vector). A BS-side decoder network maps that code to the precoder. During training, the objective is task-level (rate), and the code is quantized to meet the feedback budget. Compared with standardized codebooks, this reduces overhead while optimizing the end task directly.

Overall, DL-based feedback beats classical CS at the same overhead. The different strands of research offer distinct advantages: rate-distortion methods (1.2.2) control the feedback ovearhead by learning the probability distribution (the entropy model) used by a standard, fixed entropy coder, while task-oriented methods (1.2.3) align the entire training pipeline with the end-goal (precoding) rather than a proxy distortion metric.

1.3 The Approach Proposed by Carpi et al.

Carpi et al. [13] focus on a multi-user FDD downlink Massive MIMO system, adopting a sparse limited-scattering geometric multipath channel model.[2] Rather than treating estimation as a stand-alone module, they adopt an end-to-end viewpoint that jointly designs the downlink pilots, the channel estimation and quantization strategy under limited feedback, and the downlink linear precoder at the BS. UEs compress CSI over a rate-limited link, and the BS processes that feedback to determine directly the precoding vectors, rather than a channel estimate. They keep an autoencoder-shaped codec: the UE encoder is followed by a quantization and learned entropy model, with the feedback overhead included as a differentiable component of the optimization objective (the loss function),

[2]The technical details of this channel model, adopted from [6] and used by [13], are discussed in Section 2.3.

rather than being a fixed constraint. The goal is to understand and optimize the trade-off between CSI feedback overhead and system performance, measured by the sum achievable rate across users. Training is end-to-end in the multi-user loop with a task-oriented objective: to maximize the downlink sum-rate while simultaneously minimizing a differentiable overhead penalty, with the balance between the two controlled by a trade-off parameter λ. Their results reveal a saturation effect in the rate-overhead trade-off: beyond a certain feedback capacity, further increases in overhead yield diminishing returns in sum-rate, eventually plateauing near the theoretical limit achievable with perfect CSI (CSIT). This shows that the precoding-oriented strategy can achieve near-optimal performance with significantly compressed feedback, identifying an efficient operating regime well below the requirements of traditional reconstruction-based methods (which necessitate high-fidelity channel estimates to perform effective precoding).

1.4 Goal of dissertation

This thesis' goal is to interpret, in the sense of explainable AI, the learned precoding-oriented CSI-feedback framework of Carpi et al. [13] for downlink FDD massive MIMO, where reliable downlink precoders rely on user-supplied compressed CSI. While their deep learning framework demonstrates high performance, it operates as a "black box", leaving its learned internal strategy unknown. This thesis aims to "open" that black box using methodologies from Explainable AI (XAI), which provide *post-hoc* analysis tools to understand the inner workings of complex models like deep neural networks. Our original contribution is to adapt these XAI tools to reveal and visualize the strategy the network has learned to solve the end-to-end communication problem.
Towards this, our contributions may be summarized as follows:

- **Reproducibility and Baseline Validation**: We first successfully replicated the end-to-end learning pipeline from Carpi et al. [13]. This involved re-implementing the learned pilots, UE-side encoder, and BS-side precoder decoder, as the authors' code was not public. Our implementation validates their findings, confirming that the precoding-oriented approach achieves a higher sum-rate in the low-feedback regime. This validated model serves as the faithful baseline for our interpretability analysis.
  We conducted extensive end-to-end simulations, systematically sweeping key system parameters (number of antennas N_t, pilot length L, feedback dimensionality N_b, and related SNR settings) to understand the overhead-sum-rate performance trade off, and how different factors affect this.

- **Understanding the learned spaces**: We move beyond performance metrics to analyze the internal strategy of the learned model. We provide quantitative and visual proof (via XAI tools as quantitative clustering and non-linear dimensionality reduction) that the network's internal strategy is not fixed, but emerges as a direct function of the rate-overhead trade-off (λ). We demonstrate that the model develops a structured, semantic compression mechanism (i.e., a finite set of prototype vectors) only when forced to solve this trade-off, and that this structured behavior vanishes when the compression constraint is removed.

- **A Novel Framework for XAI in Wireless Systems:** We introduce and apply a new methodological framework for interpreting learned end-to-end communication systems. To our knowledge, this is the first investigation to apply a combined suite of XAI tools (specifically, density-based clustering, non-linear manifold visualization, and contingency matrix analysis) to explain the internal policy of a learned CSI-feedback and precoding pipeline. This work is similar in spirit to the line of research on interpreting learned error-correcting codes [19, 20, 21, 22, 23, 24, 25, 26] and provides a new template for "opening the black box" in learned wireless design.

Chapter 2

System Model

This chapter lays out Carpi et al.'s [13] system model in full detail. We retain their system setup and notation throughout.

2.1 Approach

We pose the design as an end-to-end problem: maximize the sum of achievable rates in (2.6) as subject to a budget on the feedback bits required to convey $(\mathbf{b}_1, ..., \mathbf{b}_K)$ over the uplink.

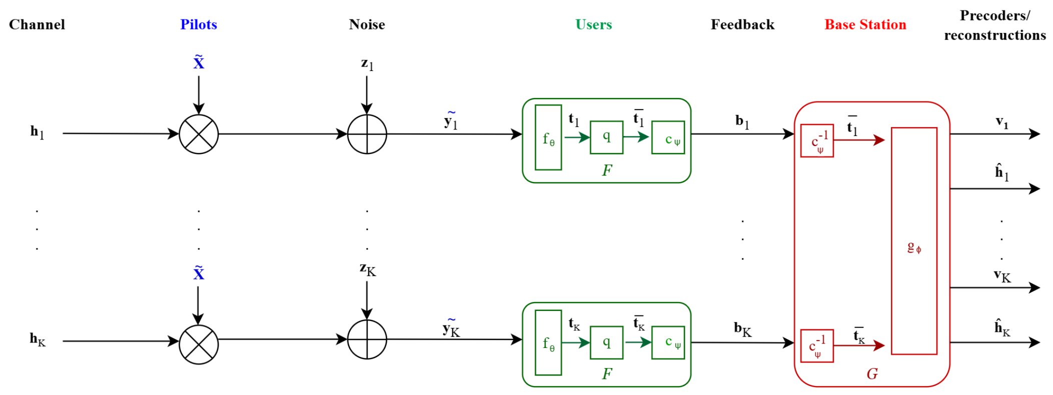


Figure 2.1: System Block Diagram.

We consider a multi-user precoding-oriented end-to-end architecture (2.1) similar to that of Sohrabi et al. [6]. While this setup is reminiscent of the CSI-feedback frameworks in [17], those methods target accurate channel reconstruction, whereas Carpi et al [13]'s objective is to learn precoders directly: a precoding-oriented loss embeds the rate-overhead tradeoff into the training objective. Under this loss,

the mapping in the UE block, denoted with $\mathcal{F}$, learns a channel representation expressly tailored for precoding efficiency; the mapping in the BS block, $\mathcal{G}$, maps that representation directly to the Precoding Matrix $\mathbf{V}$.
The pilots, feedback schemes $\mathcal{F}$, and BS processing $\mathcal{G}$ are modeled with neural networks, and incorporate the learned compression-quantization mechanism of [16] to respect the feedback-bit constraint. Unlike [6], we include a feedback overhead optimization mechanism that estimates the feedback distribution during training and uses entropy coding to generate the bit streams at test time.

2.2 Transmitted Signal

We consider a massive MIMO system where a BS with N_t transmit antennas serves K single-antenna users. Assuming linear precoding at the BS, the downlink transmitted vector is

$$\mathbf{x} = \sum_{k=1}^{K} \mathbf{v}_k s_k = \mathbf{V}\mathbf{s}$$

with $\mathbf{v}_k \in \mathbb{C}^{N_t}$ representing the precoding vector and $s_k \in \mathbb{C}$ the symbol to be sent for the k-th user.

2.3 Channel Model

The system assumes a BS equipped with a Uniform Linear Array (ULA) with N_t antennas.
The vector of channel gains for the k-th user, $\mathbf{h}_k \in \mathbb{C}^{N_t}$, is modeled as the superposition of L_p propagation paths:

$$\mathbf{h}_k = \frac{1}{\sqrt{L_p}} \sum_{\ell=1}^{L_p} \alpha_{\ell,k}\, \mathbf{a}_t(\beta_{\ell,k}), \tag{2.1}$$

Each path ℓ is defined by two components:

- The standard ULA steering vector (transmit array response), which is a function of the path's random Angle of Departure (AoD), $\beta_{\ell,k}$

$$\mathbf{a}_t(\beta) = \left[1,\ e^{j\,\psi(\beta)},\ e^{j\,2\,\psi(\beta)},\ \ldots,\ e^{j\,(N_t-1)\,\psi(\beta)}\right]^{\mathsf{T}}, \tag{2.2}$$

  with phase increment

$$\psi(\beta) = 2\pi \frac{f_c d}{c} \sin\beta, \tag{2.3}$$

where N_t is the number of BS antennas, d is the antenna spacing, f_c is the carrier frequency, and c is the speed of light.
Considering L_p paths and K users, each path ℓ towards a specific user k has its own AoD $\beta_{\ell,k}$, that is shared across all antennas towards the same user, and that is drawn uniformly within the user's angular sector ($\beta_{\ell,k} \sim \mathcal{U}(\theta_k - \Delta_k, \theta_k + \Delta_k)$). These draws are independent across both users and paths. Each antenna experiences an additional phase shift that grows linearly with the antenna index n.[1]

- The complex path gain $\alpha_{\ell,k}$ towards the k-th user, modeled as an independent, zero-mean circularly symmetric complex Gaussian random variable ($\alpha_{\ell,k} \sim \mathcal{C},\mathcal{N}(0,1)$). These gains are independent both across the K users and across the L_p paths, and identically distributed.

Because the same $\beta_{\ell,k}$ and $\alpha_{\ell,k}$ appear in every antenna entry, the N_t entries of the vector $\mathbf{h}_k$ are not independent: they are tied together by the underlying physics of the propagation paths. Since each user k has its own set of AoDs, and in our scenario both users share the same distribution parameters, they statistically look the same.
In our simulation setup, the random variables for all paths and users (both the AoDs $\beta_{\ell,k}$ and the gains $\alpha_{\ell,k}$) are drawn independently from the same statistical distributions. As a result, the channel vectors $\mathbf{h}_k$ for all users are independent and identically distributed (i.i.d.) draws from the same underlying continuous distribution.

2.4 Downlink Pilots Model

The BS probes the channel by transmitting a sequence of pilot symbols of length L. These transmissions are represented by the pilot matrix $\tilde{\mathbf{X}} \in \mathbb{C}^{N_t \times L}$. Each transmitted pilot symbol (i.e., each column $\tilde{\mathbf{x}}_\ell$ of $\tilde{\mathbf{X}}$) must satisfy an instantaneous power constraint $\|\tilde{\mathbf{x}}_\ell\|_2^2 \leq P$. The k-th user receives the pilot signal $\tilde{\mathbf{y}}_k \in \mathbb{C}^{1 \times L}$, which is modeled as:

$$\tilde{\mathbf{y}}_k = \mathbf{h}_k^H \tilde{\mathbf{X}} + \mathbf{z}_k \tag{2.4}$$

where $\mathbf{z}_k \sim \mathcal{CN}(\mathbf{0}, \sigma^2 \mathbf{I})$ is the Additive White Gaussian Noise (AWGN). The pilot matrix $\tilde{\mathbf{X}}$ is not fixed or pre-defined matrix (e.g., based on DFT). Instead, $\tilde{\mathbf{X}}$ is

[1]In our specific simulation setting, all K users are assigned the same angular support, defined by a common central angle θ and half-width Δ. While the general model could support heterogeneous users (i.e., different θ_k and Δ_k which would lead to non-i.i.d. channels), our homogeneous setup ensures that all path angles $\beta_{\ell,k}$ are drawn independently from the same uniform law, $\mathcal{U}[\theta - \Delta, \theta + \Delta]$.

treated as a set of free parameters that are jointly optimized with the encoder and decoder networks to maximize the final system objective.

2.5 User Model

After receiving the noisy pilot observation $\tilde{\mathbf{y}}_k$ (2.4), each user k processes this signal locally to create a compressed feedback message $\mathbf{b}_k$.

2.6 Feedback Model

Each UE employs the same feedback mechanism $\mathcal{F}$. Although the K UE experience distinct channel realizations $[\mathbf{h}_1, ..., \mathbf{h}_K]$, prior work [6, 17] indicates that a single, user-agnostic $\mathcal{F}$ remains effective in multiuser settings when the realizations follow the same channel statistics.
The scheme comprises three components:

1. An Encoder Network (f_θ): A deep neural network (DNN), defined as a function f_θ with a set of trainable parameters θ. Its role in the model is to extract features from the received pilots $\tilde{\mathbf{y}}_k$ and map them to a low-dimensional latent representation $\mathbf{t}_k \in \mathbb{R}^{N_b \times 1}$. The specific architecture of this network (e.g. the number of layers and the total parameter count) is an implementation choice detailed in Section 3.2.

2. A Quantizer (q): A quantizer q, which applies uniform scalar quantization to the nearest integer, yielding the quantized vector for the k-th user $\bar{\mathbf{t}}_k$. During training, this non-differentiable operation is replaced by additive i.i.d. uniform noise u_k whose support matches the quantization step $[u_k^1, ..., u_k^{N_b}] \sim \mathcal{U}[-0.5, +0.5]$ [11]. The resulting pseudo-quantized vector $\tilde{\mathbf{t}}_k = \mathbf{t}_k + \mathbf{u}_k$ is used in place of $\bar{\mathbf{t}}_k = \mathrm{q}(\mathbf{t}_k)$ to enable gradient backpropagation.

3. An Entropy Coder c_ψ: An entropy coder c_ψ with a set of trainable parameters ψ, which maps the quantized representation to a lossless bitstream. Following [16], $\tilde{\mathbf{t}}_k$ is modeled by a parameteric, fully factorized prior in which each component is Gaussian with zero mean and a standard deviation learned during training. At test time, these learned parameters are used by c_ψ to encode $\bar{\mathbf{t}}_k$.

2.7 Base Station Model

The Base Station (BS) employs a centralized processing operator $\mathcal{G}$ that aggregates the feedback from all K users to determine the downlink transmission strategy.

Unlike the user-side processing, which occurs independently per device, the BS processing is joint and multi-user. The scheme comprises three functional components:

1. An Entropy Decoder (c_ψ^{-1}): This component inverts the entropy coding applied at the UE. Its behavior differs strictly between training and testing phases to maintain end-to-end differentiability:At test time: It losslessly reconstructs the quantized latent vectors from the received bitstreams, i.e., $\bar{\mathbf{t}}_k = c_\psi^{-1}(b_k)$ for each user k.During training: The entropy decoding is bypassed. The pseudo-quantized vectors with additive noise $\tilde{\mathbf{t}}_k$ (generated by the UE) are fed directly to the next stage. This allows gradients to flow back from the BS to the UE, enabling the joint optimization of the encoder-decoder pair.

2. A Decoder Network (g_ϕ): A deep neural network parameterized by ϕ, which serves as the core processing unit. It takes as input the concatenation of the latent representations from all K users, $\mathbf{z} = [\mathbf{t}_1, \ldots, \mathbf{t}_K]$, effectively fusing the distributed CSI into a global state. Its role is to map this compressed multi-user state to the Precoding Matrix $\mathbf{V} \in \mathbb{C}^{N_t \times K}$. While the architecture could technically output a channel reconstruction $\hat{\mathbf{H}}$, in our precoding-oriented design, g_ϕ is optimized solely to generate $\mathbf{V}$.

3. Power Normalization: A deterministic layer that ensures the generated precoding matrix satisfies the physical power constraints of the massive MIMO array. The raw output of g_ϕ is normalized such that the total transmit power does not exceed the budget P:

$$\mathrm{Tr}(\mathbf{V}\mathbf{V}^H) \leq P$$

2.8 Metrics

We quantify the CSI feedback task along three axes: feedback overhead, system performance, and channel distortion.

2.8.1 Overhead

The feedback overhead captures the number of bits needed to convey the uplink messages $\mathbf{b}_k$. Following prior work [16, 17] we proxy this by the empirical entropy of the pseudo-quantized latent vector $\tilde{\mathbf{t}}_k$. This vector $\tilde{\mathbf{t}}_k$ is the output of the UE's encoder, and it is a random variable that depends on three underlying random sources: the channel realization $\mathbf{h}_k$, the additive receiver noise $\mathbf{z}_k$, the additive quantization noise $\mathbf{u}_k$, used during training. Therefore, the expected code length must be averaged over the distributions of these three source variables. Specifically, $\mathbf{t}_k = F(\mathbf{h}_k, \mathbf{z}_k, \mathbf{u}_k; \theta, \tilde{\mathbf{X}})$, where F represents the entire user-side network. The per-user overhead O_k, measured in bits per channel use [bits/ch. use], is the differential entropy of $\mathbf{t}_k$ under a learned density model $p_{\tilde{t}}(\cdot, \psi)$:

$$O_k(\boldsymbol{\theta}, \boldsymbol{\psi}) = \mathbb{E}_{\mathbf{h}_k, \mathbf{u}_k, \mathbf{z}_k}\left[-\log_2 p_{\tilde{t}}\left(\tilde{\mathbf{t}}_k; \boldsymbol{\psi}\right)\right], \tag{2.5}$$

which estimates the code length for each feedback vector $\mathbf{b}_k$.
The total overhead is the sum across the K users

$$O(\boldsymbol{\theta}, \boldsymbol{\psi}) = \sum_{k=1}^{K} O_k(\boldsymbol{\theta}, \boldsymbol{\psi}).$$

2.8.2 Performance

System performance is measured by the achievable sum-rate, measured in bits per channel use [bits/ch. use]:

$$R(\boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\phi}) = \sum_{k=1}^{K} R_k(\boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\phi}),$$

with

$$R_k(\boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{h}_k, \mathbf{U}, \mathbf{Z}}\left[\log_2\left(1 + \frac{|\mathbf{h}_k^{\mathrm{H}}\mathbf{v}_k|^2}{\sum_{j\neq k}|\mathbf{h}_k^{\mathrm{H}}\mathbf{v}_j|^2 + \sigma^2}\right)\right] \tag{2.6}$$

being the achievable rate for the k-th user.
The instantaneous rate for user k is a random variable that depends on two main components: the user's channel $\mathbf{h}_k$, and the precoding matrix $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_K]$.
The overhead metric O_k is a per-user metric, depending only on that user's local, random information (the vectors $\mathbf{h}_k$, $\mathbf{u}_k$, $\mathbf{z}_k$).
What changes in the per-user rate R_k is that the precoding matrix $\mathbf{V}$ is not fixed; it is the output of the BS's network g_ϕ, which processes the joint information from all K users:

$$\mathbf{V} = g_{\boldsymbol{\phi}}(\tilde{\mathbf{t}}_1, \dots, \tilde{\mathbf{t}}_K)$$

As established in the previous section, each latent vector $\tilde{\mathbf{t}}_k$ is itself a random variable, being a function of the original channel $\mathbf{h}_k$, the receiver noise $\mathbf{z}_k$, and the quantization noise $\mathbf{u}_k$. Therefore, the final precoding vector $\mathbf{v}_k$ (and the entire matrix $\mathbf{V}$) is a complex, deterministic function of all underlying random sources $\mathbf{H}, \mathbf{U}, \mathbf{Z}$, where $\mathbf{U}$ and $\mathbf{Z}$ represent the collection of all users' quantization and receiver noise, respectively, and $\mathbf{H}$ is the full channel matrix. To find the expected rate we must average the instantaneous rate over the distributions of these sources.

2.8.3 Distortion

To align with conventional reconstruction-based methods, we also report a distortion metric, though this is not the focus of our task-oriented analysis. The distortion is measured by the mean-squared error (MSE) between the true channel $\mathbf{H}$ and the BS-side channel reconstruction $\hat{\mathbf{H}}$.
The reconstruction $\hat{\mathbf{H}}$ is an alternative output of the same BS network g_ϕ, derived from the same latent vectors:

$$\hat{\mathbf{H}} = [\hat{\mathbf{h}}_1, \dots, \hat{\mathbf{h}}_K] = g_{\boldsymbol{\phi}}(\tilde{\mathbf{t}}_1, \dots, \tilde{\mathbf{t}}_K)$$

Crucially, just like the precoding matrix $\mathbf{V}$ analyzed in the previous section, the reconstruction $\hat{\mathbf{H}}$ is a deterministic function of the latent vectors $\tilde{\mathbf{t}}_k$. These, in turn, depend on the underlying random sources $\mathbf{H}, \mathbf{U}$, and $\mathbf{Z}$. Therefore, for the same reason as the Rate metric (2.6), the expected distortion must be averaged over the distributions of all these source variables:

$$D(\boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\phi}) = \mathbb{E}_{\mathbf{H},\mathbf{U},\mathbf{Z}} \left\| \mathbf{H} - \hat{\mathbf{H}} \right\|_F^2. \tag{2.7}$$

Chapter 3

System Implementation

This section documents, in full detail, how we implemented the precoding-oriented CSI feedback pipeline described by Carpi et al. [13] and used it to produce the final overhead–performance curve, as the authors' original code was not publicly available. The implementation follows their system model and learning architecture closely, with careful attention to reproducibility, numerical stability, and evaluation against the same baselines.

3.1 Learning Objective

We train the system end-to-end, jointly optimizing the parameters of the blocks $\mathrm{f}_\theta, \mathrm{c}_\psi$ and g_ϕ under a differentiable objective that balances the three design criteria. Specifically, during training we minimize the function

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}) = O - \lambda R + \gamma D, \tag{3.1}$$

where O quantifies the feedback overhead (2.5), R measures system performance (2.6), and D penalizes channel reconstruction error (2.7). The non-negative weights $\lambda, \gamma \geq 0$ control the trade-off among these terms. Classical overhead–distortion (or rate–distortion) formulations correspond to $\lambda = 0$, whereas precoding-oriented designs set $\gamma = 0$. In this work we adopt the latter regime ($\gamma = 0$) and sweep λ to explore different feedback budgets. Joint designs that output both precoding vectors and channel reconstructions can be accommodated by choosing $\lambda > 0$ and $\gamma > 0$, but are outside our scope.

3.2 Pipeline Implementation

We implemented the end-to-end learning pipeline as a set of classes with explicit interfaces.

3.2.1 Downlink Pilots Model

As defined in the System Model (Subsection 2.4), the pilot matrix $\tilde{\mathbf{X}} \in \mathbb{C}^{N_t \times L}$ is optimized jointly with the network. To implement this, $\tilde{\mathbf{X}}$ is instantiated in our pipeline as a trainable parameter.
The matrix multiplication $\mathbf{h}_k^H \tilde{\mathbf{X}}$ from the model is realized using a 'bias-free' linear layer, i.e. a standard fully-connected layer with its bias term set to zero and where $\tilde{\mathbf{X}}$ serves as the layer's trainable weight matrix.
To enforce the power constraint $\|\tilde{\mathbf{x}}_\ell\|_2^2 \leq P$, the columns of the $\tilde{\mathbf{X}}$ weight matrix are explicitly renormalized (L2-normalized and scaled by $\sqrt{P}$) at every forward pass of the training.
Finally, the AWGN $\mathbf{z}_k$ is simulated by adding the appropriate amount of circular complex Gaussian noise to the layer's output.

3.2.2 User Module

The `User` module implements the per-user feedback f_θ as a 4-layer fully-connected network.
The network processes the real-imaginary stacked pilot vector $\tilde{\mathbf{y}}_k \in \mathbb{R}^{1 \times 2L}$.
The forward pass is defined as

$$
\begin{aligned}
\mathbf{z}_1 &= \phi_1\left(\mathrm{BN}_1\left(\mathbf{W}_1\mathbf{y} + \mathbf{c}_1\right)\right) & \mathbf{W}_1 &\in \mathbb{R}^{1024\times(2L)} & \mathbf{c}_1 &\in \mathbb{R}^{1024\times 1}\\
\mathbf{z}_2 &= \phi_2\left(\mathrm{BN}_2\left(\mathbf{W}_2\mathbf{z}_1 + \mathbf{c}_2\right)\right) & \mathbf{W}_2 &\in \mathbb{R}^{2048\times 1024} & \mathbf{c}_2 &\in \mathbb{R}^{2048\times 1}\\
\mathbf{z}_3 &= \phi_3\left(\mathrm{BN}_3\left(\mathbf{W}_3\mathbf{z}_2 + \mathbf{c}_3\right)\right) & \mathbf{W}_3 &\in \mathbb{R}^{256\times 2048} & \mathbf{c}_3 &\in \mathbb{R}^{256\times 1}\\
\mathbf{t} &= \mathbf{W}_4\mathbf{z}_3 + \mathbf{c}_4 & \mathbf{W}_4 &\in \mathbb{R}^{N_b\times 256} & \mathbf{c}_4 &\in \mathbb{R}^{N_b\times 1}
\end{aligned}
$$

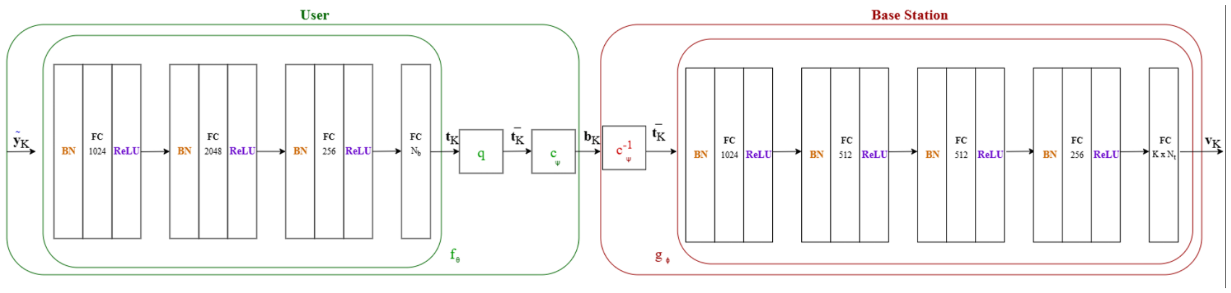


Figure 3.1: Detailed UE and BS models.

Each of the first three layers consists of a linear transformation (weights $\mathbf{W}_i$ and bias vector $\mathbf{c}_i$), followed by two standard components essential for training deep networks. First, 1-D Batch Normalization (BN_i) is applied. This technique stabilizes the training process by standardizing the activations from the previous layer to have approximately zero mean and unit variance [1]. Second, the ReLU non-linearity $\phi_i(\cdot) = \max\{0, \cdot\}$ is applied, which introduces non-linearity in the network by simply zeroing-out any negative inputs. The final layer is linear (weights $\mathbf{W}_4$, bias $\mathbf{c}_4$) and produces the latent vector $\mathbf{t} \in \mathbb{R}^{N_b \times 1}$, fed to the entropy model [2]. The same set of trainable parameters $\theta = \{(\mathbf{W}_i, \mathbf{c}_i)\}_{i=1}^4$ is used for all the users.

The final layer is strictly linear, with no activation. This is intentional: the next component in the pipeline is an entropy bottleneck that learns a parametric prior over $\mathbf{t}$ and converts it into bits. Imposing a nonlinearity at the output, as the ReLU, would distort the marginal shapes, making them harder to code: a ReLU would truncate the negative half-line, forcing $t_i \geq 0$. The entropy coder c_ψ itself is a network whose parameters ψ are trained to model the probability distribution of $\mathbf{t}$. This block is trained jointly with the encoder f_θ that produces $\mathbf{t}$. This addresses the core training dynamic: f_θ learns to produce $\mathbf{t}$, while c_ψ concurrently learns to estimate the distribution $p_{\tilde{\mathbf{t}}}(\cdot; \psi)$ through the trained parameters ψ. The loss function forces f_θ to produce latent vectors $\mathbf{t}$ that c_ψ can model efficiently (i.e., with low entropy). This learned probabilistic prior is then used both during training, assigning a likelihood to each latent sample, and during testing, driving the arithmetic coder that turns $\mathbf{t}$ into an actual bit-stream. Formally, the expected code length (in bits) under the bottleneck's prior satisfies

$$\mathrm{E}[-\log_2 q_\psi(\mathbf{t})] = H(P) + D_{KL}(P||Q_\psi)$$

where P is the empirical distribution of the latent produced by the encoder and Q_ψ is the probabilistic model (prior) that the bottleneck is trained to learn. To be precise, the bottleneck's architecture can represent a set of possible distributions. In this specific implementation, Q_ψ is the family of factorized, zero-mean Gaussian distributions. The trainable parameters ψ that define the specific distribution are the standard deviations for each independent component of the latent vector. The entropy term $H(P)$ is fixed by the data; what we can control is the mismatch

[1]At inference time, BN does not recompute batch statistics: it uses the fixed running mean and variance accumulated during training. As a result, normalization no longer depends on the current batch composition or size, and the mapping becomes deterministic: the same input, with fixed weights, always produces the same output.

[2]3.1 details the architectures of the UE module f_θ and the BS module g_ϕ, connected via the quantizer q and the learned compression blocks c_ψ. As previously said and as illustrated in the diagram, each hidden layer in both networks follows a standard, repeating motif: a BN layer, a FC layer and a ReLU activation. The last layer is linear. The number for each FC layer specifies the output dimension (i.e., the number of neurons) for that stage.

$D_{KL}(P||Q_\psi)$. Introducing a ReLU at the output distorts P, increasing this mismatch and thus the number of bits we pay for the same task performance. Keeping the final layer linear avoids that distortion: the bottleneck can drive D_{KL} down, yielding lower expected code-length.

The encoder's architecture is deliberately designed with an expansion-contraction

Table 3.1: User Module f_θ architecture.

Layer	**Input Dim.** (d_{in})	**Output Dim.** (d_{out})	**Parameters**	**Activation**	**Notes**
Layer 1	$2L = 16$	1024	W_1, c_1	BN + ReLU	$L = 8$
Layer 2	1024	2048	W_2, c_2	BN + ReLU	
Layer 3	2048	256	W_3, c_3	BN + ReLU	
Layer 4	256	$N_b = 16$	W_4, c_4	Linear	$N_b = 16$

structure (a 'narrow-wide-narrow' profile). Looking at 3.1, it begins with a small input ($2L = 16$), expands into two very wide hidden layers (1024 and 2048 neurons), and then contracts to a final latent vector ($N_b = 16$).
This design choice serves a specific purpose. The initial expansion into wide layers, which hold most of the parameters, allows for rich, non-linear mixing of the $2L$ pilot features. This enables the model to separate the task-relevant signal (precoding-oriented structure) from nuisance variations (noise and incidental correlations). The subsequent contraction to 256 concentrates what the model has distilled into a compact, information-dense representation before the final linear projection to N_b. Operationally, the same encoder f_θ is applied to every user independently, and the resulting latents are then concatenated. The order in which users are processed has no effect: permuting the users simply permutes the corresponding latents, with no change in their values. This guarantees that any differences in rate or bit-rate arise from the channel realizations themselves, not from arbitrary user indices.

Once the per-user feature vector $\mathbf{t}_k \in \mathbb{R}^{\text{batch}\times N_b}$ comes out of the MLP, the encoder immediately hands it to the entropy bottleneck, which is the component that turns the continuous latent features into an entropy-coded bitstream. We rely on the `EntropyBottleneck` module provided by `CompressAI` [27]
The behavior of this module depends on the mode:

- In training mode, the goal is to make the non-differentiable quantization step callable. For each user k, the call `entropy_bottleneck(t_k)` returns two objects: a quantized surrogate $\tilde{t}_k \in \mathbb{R}^{\text{batch}\times \text{N}_b}$ (created by adding differentiable uniform noise), and a tensor of per-symbol likelihoods $\ell_k \in \mathbb{R}_{+}^{\text{batch}\times \text{N}_b}$, i.e. the model's probability for each latent coefficient under a learned, factorized prior $(p_{\tilde{\mathbf{t}}_k}(\cdot;\psi))$. We aggregate $-\log_2 \ell_k$ across users and latent dimensions to form the overhead term in the loss. This allows gradients to flow, training f_θ to produce latents that the bottleneck c_ψ can model efficiently (i.e., with low entropy). Stacking across users yields $\tilde{\mathbf{T}} \in \mathbb{R}^{\text{batch}\times N_b\times K}$, consumed by the BS to predict the precoder, and the likelihoods $\mathbf{\Lambda} \in \mathbb{R}^{\text{batch}\times N_b\times K}$, which feed the overhead term in the loss.

- In evaluation mode, real bitstreams are produced. For each user k, the MLP output t_k is first quantized into an integer symbol sequence $s_k = \mathrm{q}(t_k) = (s_{k,1}, ..., s_{k,N_b})$, which is an inherently lossy step. Compression operates on discrete symbols, not on floats. The entropy bottleneck relies on Asymmetric Numeral System (specifically range ANS, rANS[3]) to losslessly encode the discrete sequence s_k using the learned probability model. The coder returns a list of byte strings (via `compress`, [27]), which constitutes the exact payload that would be fed back from the UE; their measured lengths (in bits) coincide to what is reported as the measured overhead (the actual payload size in bits). Because the feedback coding stage is lossless with respect to the quantized representation, and the order of users is preserved, the end-to-end mapping is deterministic: the precoder depends only on the bitstreams and the learned parameters, not on incidental runtime details.

3.2.3 Base-Station Module

The BS module receives the compressed feedback from all K users; the g_ϕ processes it jointly to generate the $N_t \times K$ precoding matrix.
The input to this module, $\bar{\mathbf{t}}$, depends on the operational mode:

- In training mode, the network receives the stack of pseudo-quantized latents $\tilde{\mathbf{T}} \in \mathbb{R}^{B\times N_b\times K}$ produced at the UE's encoder. These are concatenated to form the input tensors $\mathbf{z} = [\tilde{t}_1||\tilde{t}_2||...||\tilde{t}_K] \in \mathbb{R}^{B\times(KN_b)}$.

- In evaluation mode, the BS receives a list of byte strings (the payloads from each user). Using the same entropy model as the UE's encoder, c_ψ^{-1}, it invokes `decompress` ([27]) to losslessly recover the quantized integer symbols s_k, which are bit-for-bit identical to the encoder's quantized latents $q(t_k)$. These are

[3]rANS encodes a sequence of discrete symbols by updating a single integer state according to each symbol's probability; see [27] and [28] for details.

concatenated across users, $\mathbf{z} = [\tilde{s}_1||\tilde{s}_2||...||\tilde{t}_K] \in \mathbb{R}^{B\times(KN_b)}$, and fed to the network.

The **z** vector is fed to the module g_ϕ, which consists of a deep MLP; it acts as a "mirror image" of the UE's f_θ around the feedback bottleneck. Where the UE module compresses a single user's high-dimensional observation into a low-dimensional latent, the BS module expands the concatenated low-dimensional latents back into a high-dimensional, task-specific object (the joint precoder).
As detailed in [13], g_ϕ is a 5-layer fully-connected network, summarized in 3.2.

Table 3.2: Base-Station Module g_ϕ architecture.

Layer	**Input Dim.** (d_{in})	**Output Dim.** (d_{out})	**Parameters**	**Activation**	**Notes**
Layer 1	$K \times N_b = 32$	1024	W_1, c_1	BN + ReLU	$K = 2, N$
Layer 2	1024	512	W_2, c_2	BN + ReLU	
Layer 3	512	512	W_3, c_3	BN + ReLU	
Layer 4	512	256	W_4, c_4	BN + ReLU	
Layer 5	256	$K \times N_t = 128$	W_5, c_5	Linear	$N_t = 64$

The first four hidden layers follow the same Linear $\rightarrow$ BatchNorm1D $\rightarrow$ ReLU motif as the encoder, forming a shared "backbone". This backbone processes the concatenated input **z** and produces a final, high-level latent embedding of dimension 256. This shared embedding is then fed in parallel to two separate, linear output heads, that act as a fifth layer. One head is trained to predict the real part of the precoder ($\mathbf{V}_R \in \mathbb{R}^{64}$), and the second head is trained to predict the imaginary part ($\mathbf{V}_I \in \mathbb{R}^{64}$). These two real-valued outputs are then re-shaped, combined to form the complex final precoder ($\mathbf{V} \in \mathbb{C}^{64}$). A final normalization step enforces the power constraint $\text{Tr}(\mathbf{V}\mathbf{V}^H) = P$.
The result is an encoder-decoder symmetry around the feedback bottleneck: compression at the UE, followed by concatenation, joint decoding, and physical constraints enforcement at the BS. The symmetry is conceptual rather than parametric, since weights are not shared. Moreover, the UE encoder is applied identically to each user (preserving permutation symmetry), whereas the BS network operates jointly across all users, to capture inter-user coupling essential for multi-user precoding (such as managing inter-user interference).

3.3 Simulation Parameters and Results

The simulation parameters are summarized in 3.3. We focus on a multi-user massive MIMO system where the BS with $N_t = 64$ antennas serves $K = 2$ UEs simultaneously. The channel is modeled with $L_p = 2$ propagation paths, allowing us to study the CSI feedback task in a scenario involving both multi-user interference and multipath fading.
To reproduce the fundamental rate-overhead trade-off analysis originally presented in [13], we trained the E2E system by sweeping the trade-off parameter λ across a wide dynamic range, effectively forcing the model to operate in diverse regimes, from high-compression (low λ) to high-performance (high λ). 3.2 illustrates the resulting performance curve obtained from our simulations. The plot confirms the expected behavior described in Section 1.3: a steep initial increase in sum-rate at low feedback values is followed by a saturation effect where the performance plateau approaches the perfect CSIT upper bound. The successful replication of this performance curve serves as a validation of our codebase, confirming that the learned encoder and precoder are functioning correctly.

Table 3.3: Simulation Parameters

Parameter	**Value**
Number of Antennas N_t	64
Number of Users K	2
Number of Paths L_p	2
Length of the Latent Vector ($\mathbf{t}$) N_b	16
Length of the Pilots L	8
Number of training batches	10^4
Batch size	1024
Learning Rate	10^{-3}
Power constraint P	1
Signal-to-Noise ratio	10 dB
Sector centers β_1, β_2	$0°$
Half-widths Δ_1, Δ_2	$30°$

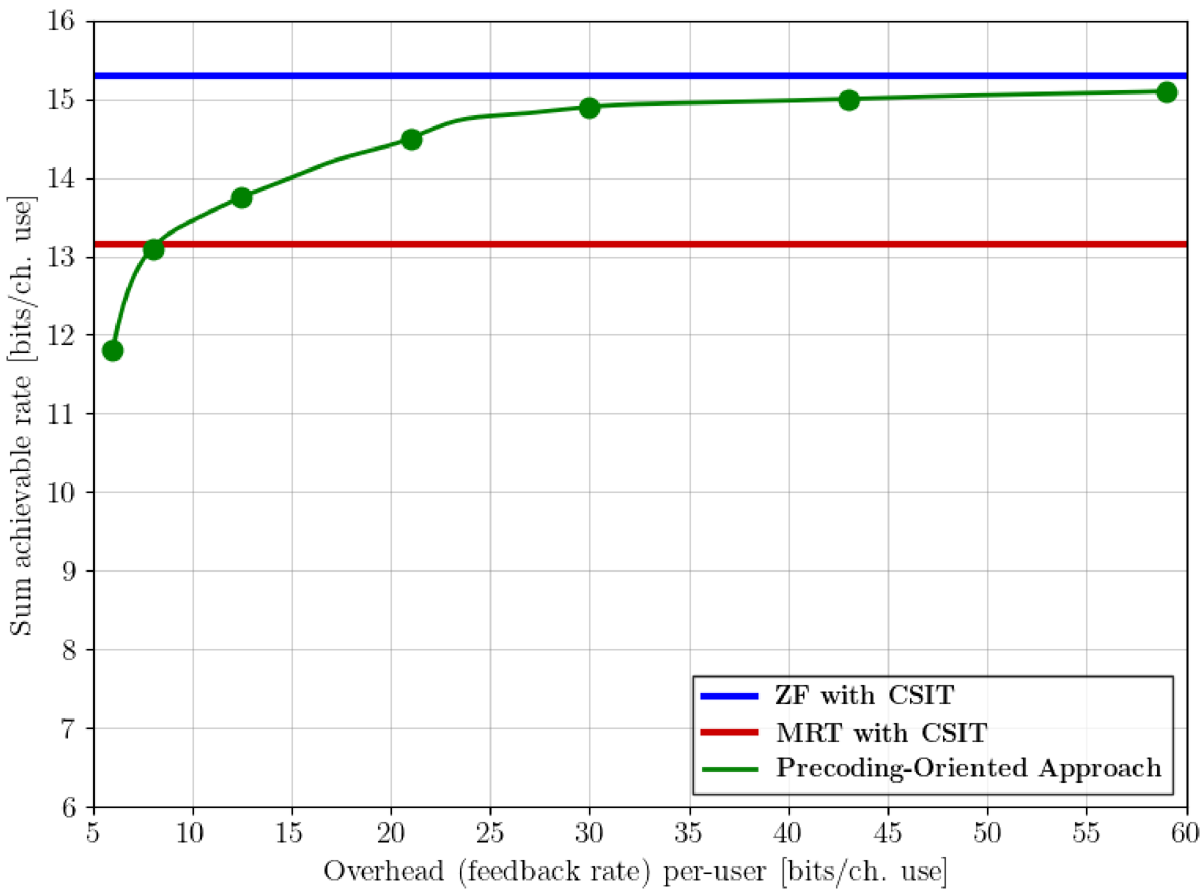


Figure 3.2: Analysis of the tradeoff between the feedback overhead and the system performance.

Chapter 4

Understanding the learned spaces

This chapter examines how the end-to-end pipeline converts channel realizations into task-relevant representations, with the tradeoff parameter λ controlling the balance between feedback overhead and system performance.
We consider two distinct operating regimes for λ:

- Compression-Oriented Regime: Low values (e.g., $\lambda = 2$), where priority is given to compression efficiency (minimizing overhead) over rate maximization.

- Performance-Oriented Regime: High values (e.g., $\lambda = 100$), where priority is given to the maximization of the sum achievable rate

We explore three complementary perspectives of the learned system, analyzing its behavior as λ varies: the variability of the latent representations at different operating points (Section 4.1); the topological structure of the learned manifolds (Section 4.2), specifically how samples are organized into coherent patterns when projected non-linearly; the adaptation of the pilot structure, observing how the learned pilots evolve under different constraints (Section 4.3).
Throughout this chapter, we present the analysis for a single representative user. It is important to note that we verified these results against the second user in the system ($K = 2$) and found them to be identical. This confirms that the learned strategy is symmetric and robust across users.

4.1 Variability of learned latent feedback representations t and precoders v

To understand the learned representations of the variables to be fed back by the UEs, denoted as **t**, and the resulting precoders **v**, we first train the model at various trade-off parameters λ and then analyze the statistics of these variables over 10 000 channel realizations **h**. 4.1 illustrates how the count of unique values varies with λ. As expected, the channel **h** is exogenous to the optimization, showing a constant number of unique instances across all settings. The latent variable **t** exhibits an abrupt phase transition: at $\lambda = 1$, we observe severe compression with only 1 unique realization. However, a minimal increase to $\lambda = 1.5$ is sufficient to restore 9 264 unique values, and for $\lambda \geq 2$, the mapping from **h** to **t** becomes perfectly one-to-one (10 000 unique values).
Conversely, the precoder **v** displays a more gradual monotonic growth in diversity. At low values ($\lambda = 1.5$ and $\lambda = 2$), despite **t** being highly diverse, the decoder generates a limited set of unique precoders (132 and 531, respectively). This suggests that in the low-budget regime, the decoder acts as a clustering mechanism, effectively learning a finite set of patterns. As the budget increases, diversity rises sharply, reaching 6 500 at $\lambda = 5$, and eventually saturates at 10 000 for $\lambda \geq 100$. This trend confirms that the mapping $\mathbf{h} \rightarrow \mathbf{v}$ transitions from a fine set-like behavior (many channels collapsing to the same prototype) to a fully user-specific strategy.
To formally quantify the geometric diversity of the learned representations, we define the pairwise distance metrics $\mu_{d(\cdot)}$ and $\sigma^2_{d(\cdot)}$ presented in 4.2.
Let $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ be the set of $N = 10000$ test samples for a given variable (i.e., $\mathbf{x}$ can be $\mathbf{h}, \mathbf{t}$, or $\mathbf{v}$). We first compute the set of all unique pairwise distances $\mathcal{D}(X)$:

$$\mathcal{D}(X) = \{d(\mathbf{x}_i, \mathbf{x}_j) | 1 \leq i \leq j \leq N\}$$

where $d(\cdot,\cdot)$ is the Euclidean (L_2) distance. The mean pairwise distance $\mu_{d(x)}$ and the covariance of pairwise distances $\sigma^2_{d(x)}$ are the empirical statistics of this set:

$$\mu_{d(h)} = \mathbb{E}_{d \in \mathcal{D}(X)}[d] = \frac{1}{|\mathcal{D}(X)|} \sum_{d_{ij} \in \mathcal{D}(X)} d_{ij}$$

$$\sigma^2_{d(h)} = \mathrm{Var}_{d \in \mathcal{D}(X)}(d) = \frac{1}{|\mathcal{D}(X)|} \sum_{d_{ij} \in \mathcal{D}(X)} (d_{ij} - \mu_{d(x)})^2$$

The mean distance $\mu_{d(x)}$ measures the average separation ("expansion") of the samples in the vector space; the variance $\sigma^2_{d(x)}$ quantifies the structural 'heterogeneity' of this spacing.
We analyze the empirical results for these statistics, shown in 4.2 over the 10000 test samples.

First, as expected, the channel statistics $\mu_{d(\mathbf{h})}$ and $\sigma^2_{d(\mathbf{h})}$ remain constant across all λ, confirming that the input distribution is not learned or affected by the feedback budget.
Focusing on the latent variable $\mathbf{t}$, we observe a sharp transition. At $\lambda = 1$, both mean and variance are essentially zero, indicating a collapse to a single cluster. However, a minimal increase to $\lambda = 1.5$ triggers an immediate expansion ($\mu_{d(\mathbf{t})} \approx 0.75$), which continues monotonically as λ grows, reaching $\mu_{d(\mathbf{t})} \approx 6.87$ at $\lambda = 150$. This aligns with the unique-value counts in 4.1: as the budget increases, the encoder projects $\mathbf{h}$ onto an increasingly larger and richer region of the $\mathbf{t}$-space. The concurrent growth in variance ($\sigma^2_{d(\mathbf{t})}$) suggests that the latent manifold is becoming geometrically complex.
The precoder $\mathbf{v}$ exhibits a different behavior due to the physical power constraints. At $\lambda = 1$, the distance is zero (single-prototype collapse). At $\lambda = 1.5$, the small mean distance ($\mu_{d(\mathbf{v})} \approx 0.25$) suggests that the vectors are distinct but densely packed. From $\lambda = 2$ onwards, $\mu_{d(\mathbf{v})}$ rises sharply and then saturates around ≈ 0.96 for high λ. This saturation is a direct consequence of the power constraint, limiting the maximum possible average separation. Crucially, while the mean saturates, the variance $\sigma^2_{d(\mathbf{v})}$ decreases from 0.085 (at $\lambda = 2$) to 0.021 (at $\lambda = 150$). This implies a regularization of the geometry: as the model refines its strategy, the precoders become more uniformly distributed over the available angular space. Once the model has sufficient overhead to generate effective, interference-aware precoders, any additional bits are likely used for fine-tuning rather than creating drastically different beams. Since all precoders are normalized by the power constraint, they cannot move indefinitely far apart. As all solutions converge towards a set of near-optimal, power-normalized vectors, their average pairwise distance can stabilize or even slightly decrease. In summary, the encoder uses the relaxed penalty to expand the latent space $\mathbf{t}$ indefinitely (increasing μ, increasing σ^2), while the decoder utilizes this information to map inputs onto a regular, highly structured, and power-limited precoding manifold (saturating μ, decreasing σ^2). As λ grows, $\mathbf{t}$ becomes more expressive, while $\mathbf{v}$ becomes more diverse yet more regular (larger but tightly distributed distances), and $\mathbf{h}$ remains unchanged: the learned pipeline transitions from a "finite-set"-like behavior to a stable, task-oriented precoding regime.

4.2 t-SNE representations

In this section, we visualize the learned representations ($\mathbf{t}$ and $\mathbf{v}$ vectors) by projecting them from their original high-dimensional space into a two-dimensional (2D) space.

Table 4.1: Unique-value counts vs. λ.

λ	$\mathbf{h}_k$	$\mathbf{t}_k$	$\mathbf{v}_k$
1	10,000	1	1
1.5	10,000	9,264	132
2	10,000	10,000	531
5	10,000	10,000	6,500
10	10,000	10,000	9,000
25	10,000	10,000	9,500
50	10,000	10,000	9,800
100	10,000	10,000	10,000
150	10,000	10,000	10,000

Table 4.2: Pairwise distance statistics vs. λ (user 0).

λ	$\mu_{d(h)}$	$\sigma^2_{d(h)}$	$\mu_{d(t)}$	$\sigma^2_{d(t)}$	$\mu_{d(v)}$	$\sigma^2_{d(v)}$
1	10.845	8.465	0.000	0.000	0.000	0.000
1.5	10.845	8.465	0.750	1.100×10^{-1}	0.250	0.020
2	10.845	8.465	1.038	1.726×10^{-1}	0.889	0.085
5	10.845	8.465	1.426	1.197×10^{-1}	0.941	0.040
10	10.845	8.465	1.938	3.399×10^{-1}	0.952	0.027
25	10.845	8.465	2.736	4.448×10^{-1}	0.964	0.024
50	10.845	8.465	3.690	6.764×10^{-1}	0.973	0.023
100	10.845	8.465	5.001	1.092	0.967	0.021
150	10.845	8.465	6.868	1.852	0.958	0.021

A standard baseline, Principal Component Analysis (PCA), was considered and rejected because it is a strictly linear transformation. PCA's objective is to find a new set of orthogonal axes that maximize the global linear variance. This methodology is incompatible with out data, output of deep neural networks (complex and highly non-linear functions). There is no reason to assume that the representations learned by the network are separable along simple linear axes; it is far more likely that they exist on a complex, non-linear manifold. Applying a linear projection (PCA) to this inherently non-linear data would fail to capture this structure. Therefore, we selected *t*-SNE, a non-linear dimensionality reduction technique, whose objective is to preserve local neighborhood structure. A more detailed explanation of the *t*-SNE algorithm follows.

t-SNE, short for *t-distributed Stochastic Neighbor Embedding*, is a dimensionality reduction technique that transforms complex, high-dimensional data into a 2D visualization while preserving the local structure of the original data [29]. In essence, it enables the interpretable visualization of otherwise intractable datasets, maintaining the proximity of similar points even in the lower-dimensional representation. In this section, we apply t-SNE reductions in order to obtain a visual sense of the high dimension $\mathbf{t}$ and $\mathbf{v}$ spaces. The algorithm operates through two distinct phases [29]:

1. In the first phase, the algorithm analyzes the data in the original high-dimensional space. For each point, it computes a similarity measure with all other points, converting Euclidean distances into probability. Intuitively, this probability represents "the likelihood that this point is a neighbor of that point in the original high-dimensional space": highly similar points receive elevated probabilities, while dissimilar points receive low probabilities. This conversion is performed through a Gaussian distribution, which assigns decreasing probabilities as distance increases.

2. In the second phase, the algorithm positions all points on a two-dimensional plane, attempting to reproduce these neighborhood relationships. The process begins by randomly initializing point positions on the 2D plane. For each configuration, the algorithm computes new distances between points in the 2D space and converts them into neighborhood probabilities using the Student's t-distribution; these are called "current probabilities". Simultaneously, the algorithm retains the reference neighborhood probabilities computed from the original high-dimensional distances in phase one. The core optimization step then compares the reference probabilities (derived from the original high-dimensional space) with the current probabilities (derived from the 2D layout), and iteratively adjusts the positions of points on the 2D plane to make these two probability distributions as similar as possible. The algorithm minimizes a cost function (Kullback-Leibler divergence[30]) that measures how well the probability of neighborhood in the 2D plane reflects that of the original high-dimensional space, iterating until convergence.

The key insight underlying this choice is that a naive approach would simply apply the same Gaussian distribution used in phase one. However, this creates a fundamental problem: when compressing from high-dimensional space to 2D, a Gaussian kernel produces vanishingly small probabilities at moderate distances, failing to provide sufficient repulsive force between points. This leads to the "crowding problem"[29], where points collapse toward the center of the visualization. By employing the heavy-tailed Student's t-distribution instead, the algorithm ensures that even moderately distant points maintain residual influence, preventing cluster collapse and preserving adequate spacing between groups.
Correctly interpreting a t-SNE visualization is essential. Tight clusters in a t-SNE plot generally reflect groups of points that were proximate in the original space, and empty space between clusters indicates low similarity between groups. However, it is critical to understand that absolute distances between clusters should not be interpreted as quantitatively meaningful: the long-range geometry is an artifact of the optimization procedure and does not represent intrinsic relationships in the data [29]. The algorithm is explicitly designed to preserve local neighborhoods, not global structure. Consequently, the geometry within a cluster is typically interpretable, whereas the spacing between separate clusters, as well as the rotation and reflection of the map, depend critically on tuning parameters and initialization randomness, and therefore should not be over-interpreted.

To probe the geometry learned by our model, we apply t-SNE to two representations:

1. $\mathbf{t}_k$, the latent encoder output for the k-th user
2. $\mathbf{v}_k$, the precoding vector for the k-th user

For clarity and brevity, our analysis focuses on the representations for a single user ($k = 1$), as the channel statistics are identical for all users and preliminary tests confirmed that the resulting embeddings for $k = 2$ are qualitatively and structurally equivalent.
For each representation we compute a baseline 2-D embedding and we recolor the same 2-D coordinates with different scalars to reveal how auxiliary quantities align with local neighborhoods.
Specifically, we use three colorings per embedding:

1. Per-user Rate R_k, that tests whether performance varies smoothly within local neighborhoods of the manifold
2. Channel magnitude vector per user $|\mathbf{h}_k|$, that tests how much the channel norm organizes the local structure
3. When the geometry is built from $\mathbf{t}_k$, we color by the precoder magnitude $|\mathbf{v}_k|$; when the geometry is built from $\mathbf{v}_k$, we color by the latent magnitude $|\mathbf{t}_k|$
4. Channel parameters (Path Gains α and AoD β), analyzed specifically for the latent representation $\mathbf{t}_k$ to determine which features are relevant during the encoding phase

We overlay scalar variables on a single, fixed t-SNE map to test how they align with the neighborhoods uncovered by the embedding. This can reveal several patterns:

- smooth gradients along discovered clusters (indicating that the scalar varies locally and gradually along the manifold)
- localized “hot/cold spots" (suggesting context-specific adjustments to specific regimes)
- near-uniform fields (that implies the scalar is not a principal organizer of the manifold)

t-SNE preserves local neighborhoods; global distances, angles and absolute radii are not metrically meaningful. We therefore interpret color patterns within these clusters, while avoiding metric claims based on exact inter-cluster spacing or angles.
We analyze two operating points, $\lambda = 2$ and $\lambda = 100$, to reveal how the geometry of $\mathbf{t}_k$ and $\mathbf{v}_k$ changes across this spectrum.

4.2.1 Latent Vector $\mathbf{t}_1$

Following the methodology outlined previously, we now analyze the t-SNE embedding for the latent vector $\mathbf{t}_1$ (corresponding to User 1). This projection provides a geometry-only view of the latent space.

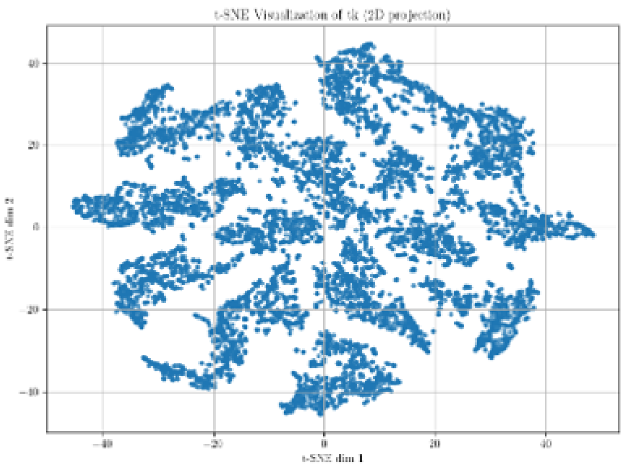


Figure 4.1: *t*-SNE representation of $\mathbf{t}_1$ for $\lambda = 2$

$\lambda = 2$ We begin our analysis with the $\lambda = 2$ operating point. This setting represents the high-compression, low-overhead regime, strongly penalizing the feedback rate. Our analysis, consistent with the methodology outlined, focuses on the latent vector $\mathbf{t}_1$ (the signal fed into the entropy coder) for a single user. The goal is to visualize the geometric structure of the latent space when the model is forced to be extremely efficient.
The t-SNE embedding for this setting, shown in 4.1, exhibits a star-like structure: it is composed of several distinct clusters, which are not uniform blobs but rather take the form of thin, filament-like structures (which we will refer to as “strands") that radiate from a denser central hub. Compact micro-clusters are separated by small gaps along each strand. This topology is typical of neighborhood-preserving embeddings and indicates that the dataset contains multiple, locally coherent regions (i.e., sub-manifolds) in the original feature space. Axes have no semantic meaning and the layout may rotate or flip across runs; only local neighborhoods and separations are intended to be read. Consequently, the existence of distinct blobs/strands is informative, whereas the exact distances or angles between separate strands are not.

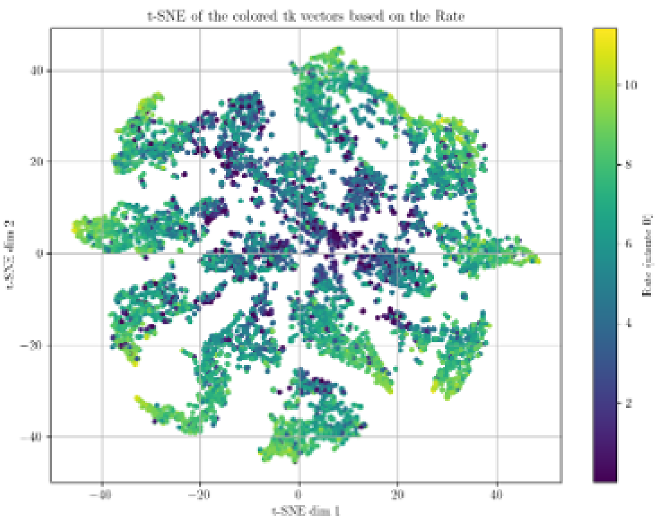


Figure 4.2: t-SNE representation of $\mathbf{t}_1$ for $\lambda = 2$ colored by R_1

We start coloring the embedding according to the precoding vector magnitude $|\mathbf{v}_k|$, as illustrated in 4.4. The structure appears organized: some clusters are entirely salmon-colored (mid values of $|\mathbf{v}|$), while others range between orange and yellow (higher values of $|\mathbf{v}|$). This indicates a correlation between the value of $\mathbf{t}$ and the precoder magnitude determined by the BS. While the magnitude range is relatively narrow, it notably excludes values tending toward zero. This implies that the BS has learned to avoid low-power regions (which would penalize the achievable rate) and instead effectively tries to maximize it. Crucially, all points falling within the same **t**-cluster are mapped by the BS to the same precoder prototype.

Coloring the embedding by the channel magnitude $|\mathbf{h}|$, as shown in 4.3, reveals a different set of patterns. Unlike the previous plot, the change is not purely monotonic along each arm, and the gradient is significantly less pronounced (even though a closer inspection reveals that high $|\mathbf{h}|$ values tend to concentrate at the edges of the islands). The main point is that hot spots and cool segments alternate along many strands. A possible interpretation is that parameters other than channel magnitude influence the latent representation, implying that magnitude itself is not the most dominant feature. Consequently, both strong and weak channels can end up within the same semantic cluster.

Coloring by rate, shown in 4.2, reveals smooth, intra-strand gradients. The darker and purple tones, corresponding to low achievable rates, cluster predominantly near the center, while greener and yellow tones, representing high rates, are more common towards the tips of the strands. Within many of these strands, the color changes smoothly, suggesting that the rate varies gradually along local manifolds rather than jumping abruptly. In other words, the rate is locally correlated with the features that drive those strands. This suggests that a single **t**-cluster is not associated with uniform rate values. The rate variation among the samples could possibly mirror the variations in the channel magnitude, not constant within a given cluster. The rate term depends not only on the precoding vector but also on the channel term: consequently, the rate gradient observed within a cluster coincides with the variation in channel magnitude. A plausible interpretation is that the model clusters the latents based on their requirement for a specific precoder configuration, rather than clustering them based on the final target outcome (i.e., the rate).

Finally, we re-color this fixed geometry using the four distinct physical parameters that generate the channel (4.5, 4.6). As defined in the Channel Model Section (2.3), these parameters are the path gains $(\alpha_{1,0}, \alpha_{2,0})$ and the AoDs $(\beta_{1,0}, \beta_{2,0})$, which are unique for each propagation path (indexed by $\ell \in \{1, ..., L_p\}$), and each user (indexed by $k \in \{1, ..., K\}$). The Path Gain $\alpha_{\ell,k}$ represents the signal strength of the ℓ-th path to the k-th user; the AoD $\beta_{\ell,k}$ is the real-valued angle of the ℓ-th path for the k-th user, and it represents the physical direction from which that path originates at the BS array. In our simulation setup, the channel vector for each user is generated as the superposition of $L_p = 2$ distinct propagation paths. Consequently, the physical parameters governing the channel consist of two independent pairs of variables: two AoDs (β_1, β_2), and two Path Gains (α_1, α_2). To analyze their individual impact on the learned representation, we present the

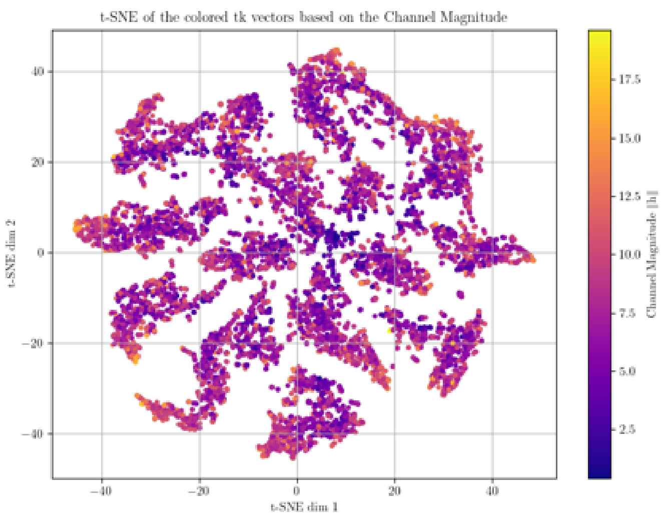


Figure 4.3: t-SNE representation of $\mathbf{t}_1$ for $\lambda = 2$ colored by $|\mathbf{h}_1|$

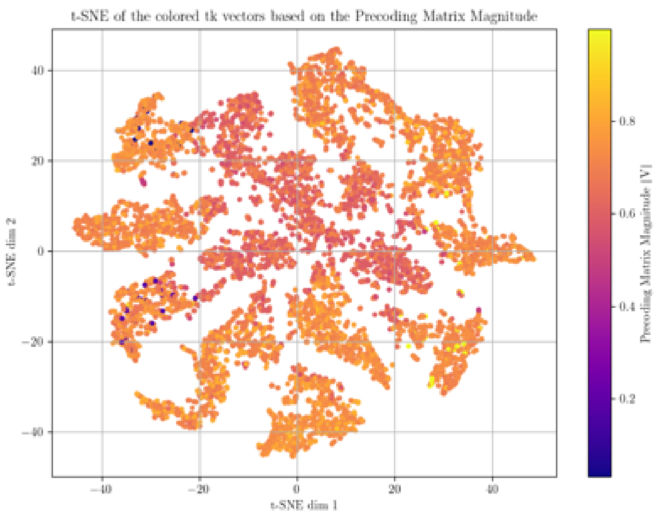
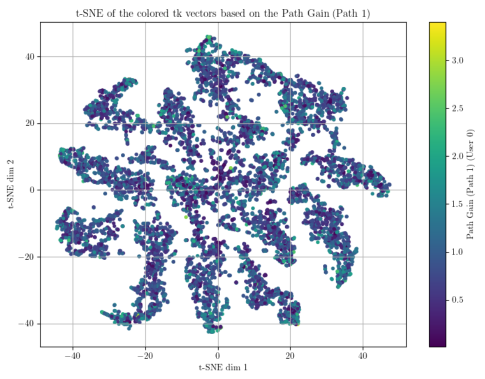


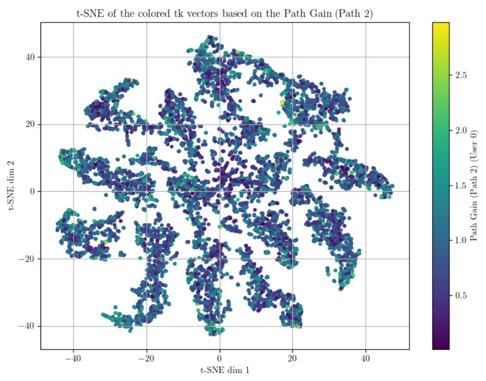
Figure 4.4: t-SNE representation of $\mathbf{t}_k$ for $\lambda = 2$ colored by $|\mathbf{v}_k|$

t-SNE visualizations organized into two comparative figures: 4.5 presents the latent space colored by the Path Gain of the first path (α_1, left) alongside the Path Gain of the second path (α_2, right); 4.6 presents the latent space colored by the AoD of the first path (β_1, left) alongside the AoD of the second path (β_2, right).
The *t*-SNE maps colored by the path gains $\alpha_{1,0}$ and $\alpha_{2,0}$ (4.5) show no discernible structure; this visualization shows a chaotic, uncorrelated distribution. Within any single cluster, we see a complete mix of colors: dark purple points (low gain) are mixed with yellow points (high gain). There is no gradient or separation. This lack of correlation demonstrates that the latent manifold's geometry is not organized by the channel strength (gain) of the individual L_p paths. The encoder has learned that this specific information, the absolute gain of a path, is secondary or irrelevant for the precoding task, therefore it "compresses away" this information.
Instead, considering the AoDs, the plots in 4.6) show a more organized structure with respect to the ones related to the α coefficient. For example, certain islands are predominantly dark blue (low angles), while others are yellow-green (higher angles). This proves that the spatial direction of the user is the one of the primary features driving the clustering process: "where the UE is located with respect to the BS" is one of the most critical piece of semantic information for the precoding task.
Throughout our analysis, we visualized the latent spaces colored by the parameters of the first path (Path 1). However, we verified that the visualizations corresponding to the second path (Path 2) yield identical structures and color distributions. Since the two paths are drawn from identical distributions and contribute symmetrically to the channel sum (2.1), the Neural Network (NN) correctly treats them as interchangeable, without learning any artificial bias or preference for one path index over the other.

$\lambda = 100$ We now turn our analysis to the $\lambda = 100$ operating point, to observe how the latent geometry of $\mathbf{t}_1$ adapts when performance, rather than extreme compression, is the primary driver. The map shown in 4.7 still has a filament motif, but the strands are much narrower and smoother, and more continous, forming loop-like ribbons that wind around the center. Under a higher rate emphasis, neighbor relationships become tighter and inter-strand scatter decreases. When coloring the $\mathbf{t}_k$ embedding by the per-user rate R_k (4.8), we observe a stark contrast to the $\lambda = 2$ case. The coloring is now fairly uniform, dominated by mid-to-high rate values (green), with scattered purple “hot spots" and a few localized lighter spots. The strong gradient based on the rate, which was evident at $\lambda = 2$ (4.2), has disappeared. At $\lambda = 2$, the model learns the simplest possible mapping, without sending complex precoding information as the small rate reward isn't worth the high overhead cost. At $\lambda = 100$, the model is forced to learn a more complex and “expensive" strategy to achieve higher rates, even though this information costs

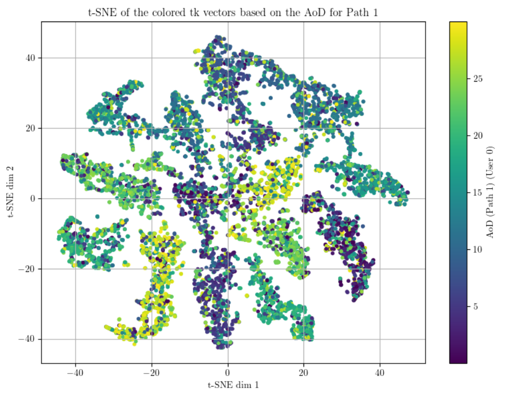


(a) Path 1 Gain Coefficient for User 0 ($\alpha_{1,0}$)

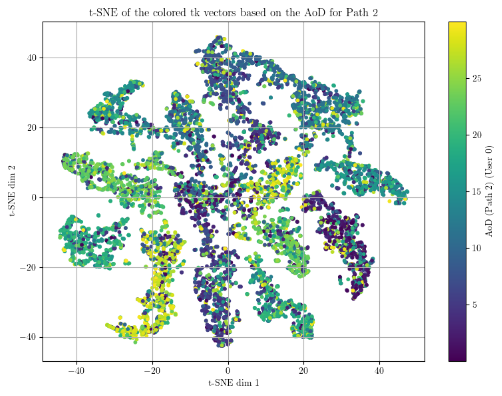


(b) Path 2 Gain Coefficient for User 0 ($\alpha_{2,0}$)

Figure 4.5: t-SNE representation of $\mathbf{t}$ colored by the Gain coefficients for User 0 for Path 1 and Path 2 ($\alpha_{1,0}, \alpha_{2,0}$)



(a) Path 1 AoD for User 0 ($\beta_{1,0}$)



(b) Path 2 AoD for User 0 ($\beta_{2,0}$)

Figure 4.6: t-SNE representation of of $\mathbf{t}$ colored by the AoDs for User 0 for Path 1 and Path 2 ($\beta_{1,0}, \beta_{2,0}$)

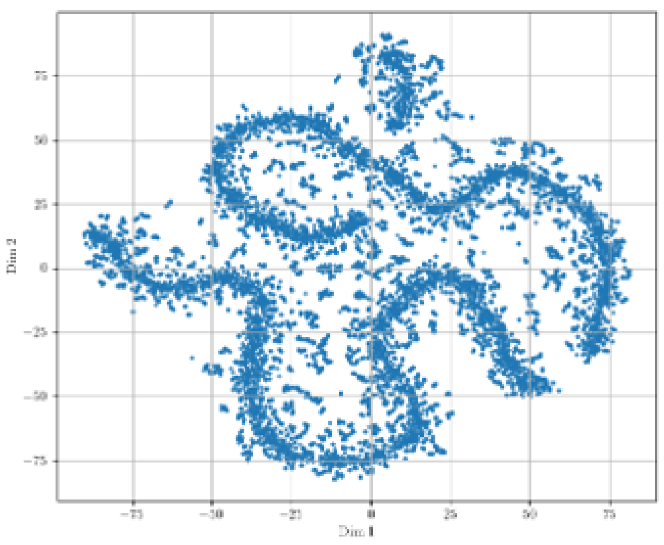


Figure 4.7: t-SNE representation of $\mathbf{t}_k$ for $\lambda = 100$

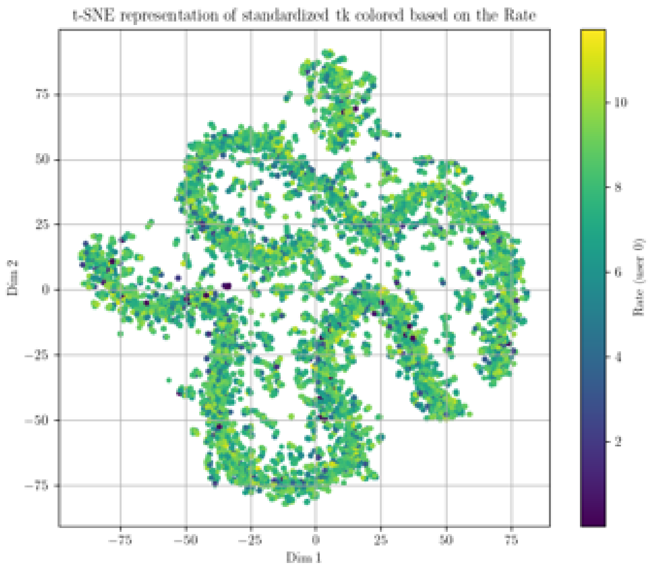


Figure 4.8: t-SNE representation of $\mathbf{t}_k$ for $\lambda = 100$ colored by R_k

more overhead. The model has learned to compute the optimal precoding solution required to actively maximize the rate. The lack of a visible gradient suggests that the model maintains high performance throughout almost the entire space.

When coloring the $\mathbf{t}_1$ embedding by the precoding vector magnitude $\mathbf{v}_1$ (4.9), we

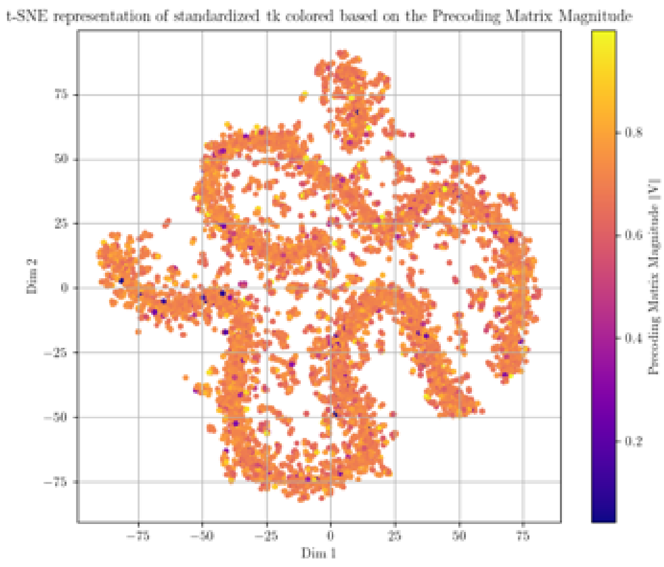


Figure 4.9: t-SNE representation of $\mathbf{t}_k$ for $\lambda = 100$ colored by $|\mathbf{v}_k|$

observe a largely uniform orange-to-yellow field. The very slight gradient visible in 4.4, where the central hub was a duller salmon and the outer arms were a brighter yellow, has now completely vanished. The color field is far more homogeneous, indicating that the BS has learned a strategy that decouple power allocation ($|\mathbf{v}_k|$, the color) from $\mathbf{t}_k$. The BS applies a policy of high power allocation. To maximize the rate, the optimal strategy is to saturate the power constraint for each individual sample, effectively utilizing as much power as possible. The encoder network has learned a representation $\mathbf{t}$ such that, regardless of its location within the latent space, it consistently ensures maximum rate performance and full power utilization. Consequently, the continuous variability of the input does not interfere with the stability of the output. In this scenario, the model is not concerned with economizing on bits. Coloring based of the specific channel realization magnitude (4.19), it successfully identifies an optimal mapping for both $\mathbf{t}$ and $\mathbf{v}$ to maximize the rate. In this case, the representation is continuous and preserves the variance of $\mathbf{h}$. The network has not prioritized channel magnitude as the fundamental information. It does not organize samples based on this metric alone but considers other latent features; consequently, neighboring points in the space may exhibit different channel magnitudes yet are grouped together due to other shared similarities considered important by the encoder.

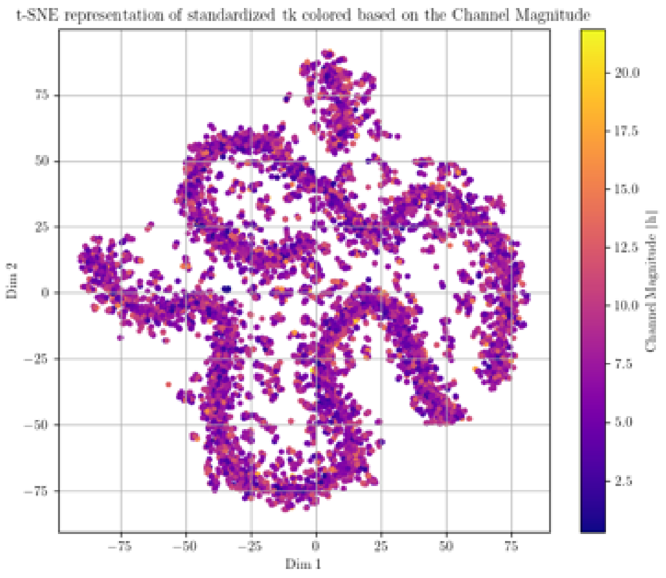
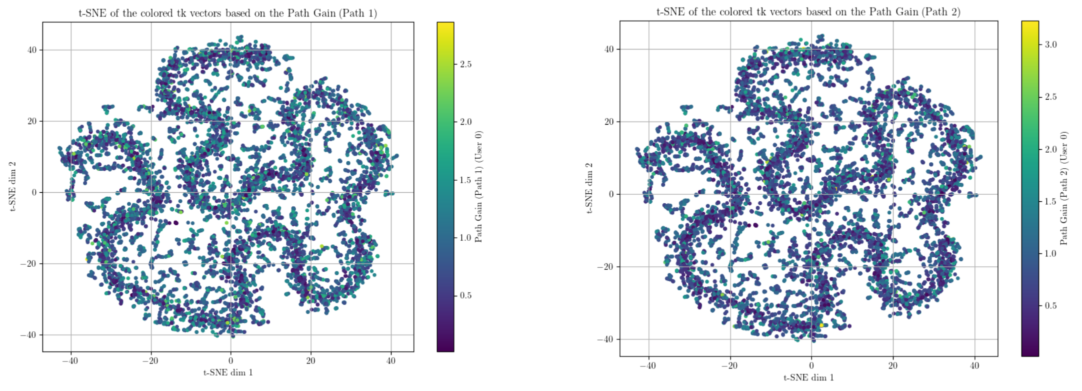
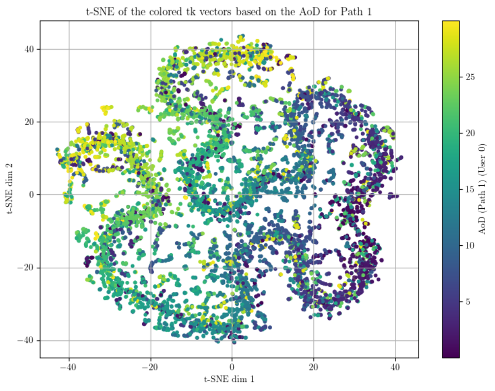


Figure 4.10: t-SNE representation of $\mathbf{t}_k$ for $\lambda = 100$ colored by $|\mathbf{h}_k|$

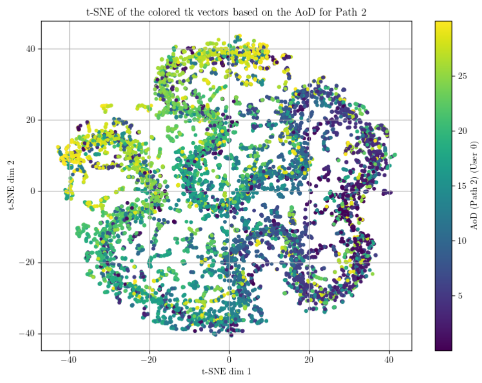


(a) Path 1 Gain Coefficient for User 0 ($\alpha_{1,0}$) **(b)** Path 2 Gain Coefficient for User 0 ($\alpha_{2,0}$)

Figure 4.11: t-SNE representation of $\mathbf{t}$ colored by the Gain coefficients for User 0 for Path 1 and Path 2 ($\alpha_{1,0}, \alpha_{2,0}$)



(a) Path 1 AoD for User 0 ($\beta_{1,0}$)



(b) Path 2 AoD for User 0 ($\beta_{2,0}$)

Figure 4.12: t-SNE representation of $\mathbf{t}$ colored by the AoDs for User 0 for Path 1 and Path 2 ($\beta_{1,0}, \beta_{2,0}$)

4.11 and 4.12 show the *t*-SNE visualizations colored by, respectively, the path gains coefficients α and the AoDs β in the $\lambda = 100$ regime.
The *t*-SNE maps colored by the path gains $\alpha_{1,0}$ and $\alpha_{2,0}$ (4.5), as in the $\lambda = 2$ regime, show no discernible structure. In sharp contrast, the maps colored by the AoDs $\beta_{1,0}$ and $\beta_{2,0}$ reveal a strong gradient. There is an evident and clear visual progression: the colors transition smoothly from dark purple to yellow across the entire manifold structure.
Geometry appears to be random if the underlying variable is uncorrelated with $\mathbf{t}$, as it happens with $\alpha_{\ell,k}$. The presence of this strong gradient for β shows that the encoder f_θ has learned to organize its latent geometry based on this physical parameter. The model has learned to represent the AoD with high fidelity, preserving the granularity of the physical parameter. The encoder has learned that all channels $\mathbf{h}_k$, regardless of the gain coefficients α, are functionally similar for the precoding task if they share a similar angle β. The encoder's learned function maps all of these functionally similar channels to similar latent vectors $\mathbf{t}_k$, i.e. it assigns them all to the same “neighborhood" in the 16-dimensional latent space. Whenever the BS receives a latent vector $\mathbf{t}_k$ from that specific neighborhood, then it will already know which precoding strategy to apply; $\mathbf{t}_k$ gives the decoder all the relevant information for the precoding strategy.
When the BS transmits the desired signal to the first User (using a precoding vector $\mathbf{v}_1$), the same signal also travels to the second user, where it will be received as interference. The goal of the precoder $\mathbf{v}_1$ is therefore to both maximize the signal in the desired direction (i.e., to align $\mathbf{v}_1$ to the channel $\mathbf{h}_1$, achieving a large $\mathbf{h}_1^H\mathbf{v}_1$), and to minimize the signal in the unintended direction (this is achieved when $\mathbf{v}_1$ is “orthogonal" to the channel $\mathbf{h}_2$, yielding $\mathbf{h}_2^H\mathbf{v}_1 \approx 0$). These plots show that the AoD β is a crucial information to solve this optimization problem (i.e., find a $\mathbf{v}_1$ aligned with $\mathbf{h}_1$ but orthogonal to $\mathbf{h}_2$), while α is irrelevant. The encoder, is learning to isolate β, is learning to extract the only piece of information that the BS needs to solve the interference problem and perform the precoding task.

4.2.2 Precoding Vector $\mathbf{v}_k$

$\lambda = 2$ Looking at the *t*-SNE of the Precoding Vector $\mathbf{v}_k$, with $\lambda = 2$ (4.13) the embedding fragments in multiple compact pockets separated by clear gaps around a loose center. There is no single continous path threading the entire set.
This reflects that, with $\lambda = 2$, $\mathbf{v}_k$ occupies several discrete local modes, small regions in which samples are very similar to one another but not connected to other modes.

Coloring the embedding by rate values (4.14) reveals that the rate varies heterogeneously across the pockets, exhibiting a broader dynamic range—from purple

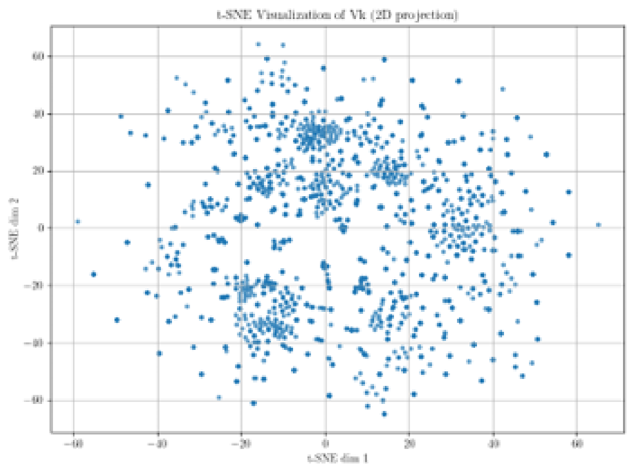


Figure 4.13: t-SNE representation of $\mathbf{v}_k$ for $\lambda = 2$

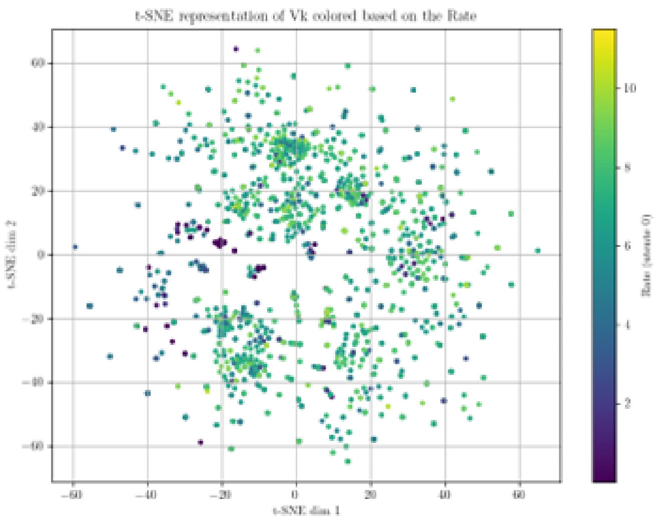


Figure 4.14: t-SNE representation of $\mathbf{v}_k$ for $\lambda = 2$ colored by R_k

(lower rate) to yellow (higher rate). While colors are relatively consistent within a given pocket, the rate level changes noticeably between different pockets. There is clear variability: while specific zones are dominated by lighter points and others by darker tones, they are not perfectly clustered. This aligns with the fact that even when the BS selects the optimal precoder (and thus the correct cluster), the final achievable rate remains dependent on the actual channel realization, causing the result to vary on a case-by-case basis. Consequently, the clusters are not monochromatic, as they would be if the rate were determined exclusively by the precoder choice.

Coloring based on the latent vector magnitude $|\mathbf{t}_k|$ (4.15) reveals that the color

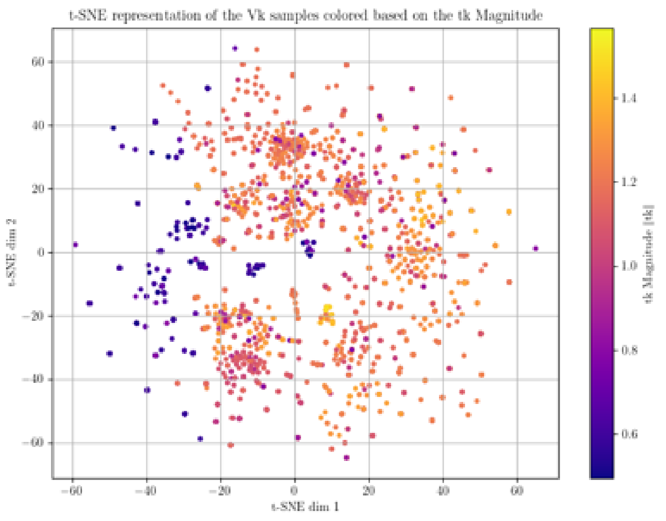


Figure 4.15: t-SNE representation of $\mathbf{v}_k$ for $\lambda = 2$ colored by $|\mathbf{t}_k|$

varies substantially across pockets. Several islands on the lower-left side are purple (indicating low $|\mathbf{t}_k|$), whereas others, especially towards the upper-right, are orange and yellow (high $|\mathbf{t}_k|$). A distinct color gradient is visible from left to right, highlighting a marked shift in magnitude. The $\lambda = 2$ setting thus results in diverse magnitude regimes and separated modes. A strong structure is evident here: some clusters are entirely light (higher magnitude), while others are entirely dark (lower magnitude). This demonstrates that the magnitude of the latent vector $\mathbf{t}$, input to the BS, is a determining factor in the selection of the precoder $\mathbf{v}$, and a discriminative feature for deciding which strategy to adopt.

Coloring the embedding by the channel vector magnitude $|\mathbf{h}_k|$ (4.16) reveals substantial variation across pockets, ranging from purple (low values) to orange and yellow (higher values). However, the colors appear chaotic and not on a gradient base, confirming that the choice of precoding is independent of channel magnitude. From the perspective of the BS, both weak and strong channels sharing the same

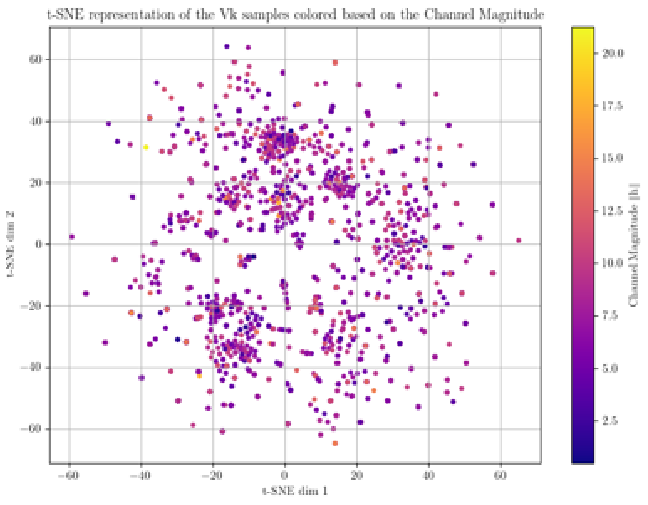


Figure 4.16: t-SNE representation of $\mathbf{v}_k$ for $\lambda = 2$ colored by $|\mathbf{h}_k|$

direction require the same precoder configuration. Therefore, the clusters are likely defined by spatial directionality, irrespective of the channel strength.

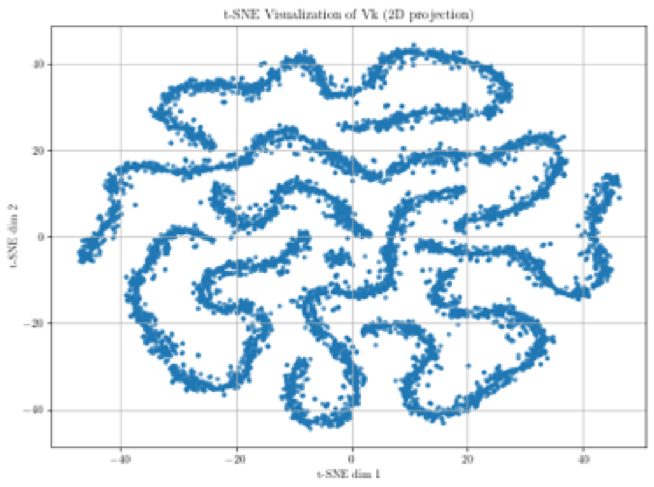


Figure 4.17: *t*-SNE representation of $\mathbf{v}_k$ for $\lambda = 100$

$\lambda = 100$ In 4.17, with $\lambda = 100$, the embedding for $\mathbf{v}_1$ collapses in a serpentine manifold. This smooth, continous structure contrasts sharply with the discrete structure of the $\lambda = 2$ regime (4.13), which suggests that the BS has learned a discrete and limited codebook of precoders. The continous manifold in 4.17 reflects the behaviour of the system: the decoder acts as a smooth, generative function. The encoder provides a rich and continous feebdback vector $\mathbf{t}_1$, and the decoder uses this rich input to calculate a unique precoder $\mathbf{v}_k$ perfectly tailored to that specific feedback.

Coloring the $\mathbf{v}_1$ manifold by the Rate R_1 (4.18), the entire “serpentine manifold" identified in 4.17 is now colored almost uniformly with mid-to-high rates. This is in contrast with the $\lambda = 2$ case (4.14), where purple points were much closer with respect to $\lambda = 100$. What happens here is that we see sparse purple points, that represent rare cases (e.g. severe interference), but not a systemic failure. Crucially, there is still one small purple cluster that remains, and that represents a small cluster of irrecoverable channels.

Colors in the channel range (4.19) are mostly mid-high, with short and smooth variations along the curve and a few localized dips: $|\mathbf{h}_k|$ is broadly uniform, with gentle and local adjustments. A continuous and complex blend of colors is visible along the curves, indicating that continuous variations in $\mathbf{h}$ map into continuous variations in $\mathbf{v}$. This confirms that the model has learned to decouple the spatial direction (which likely determines the precoder) from the channel

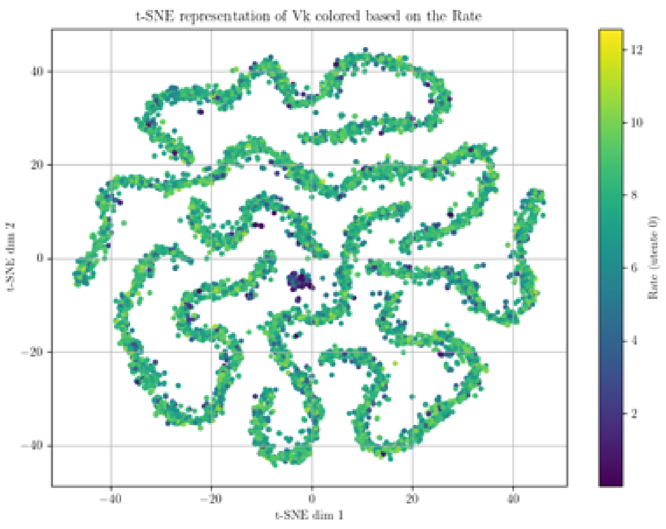


Figure 4.18: t-SNE representation of $\mathbf{v}_k$ for $\lambda = 100$ colored by R_k

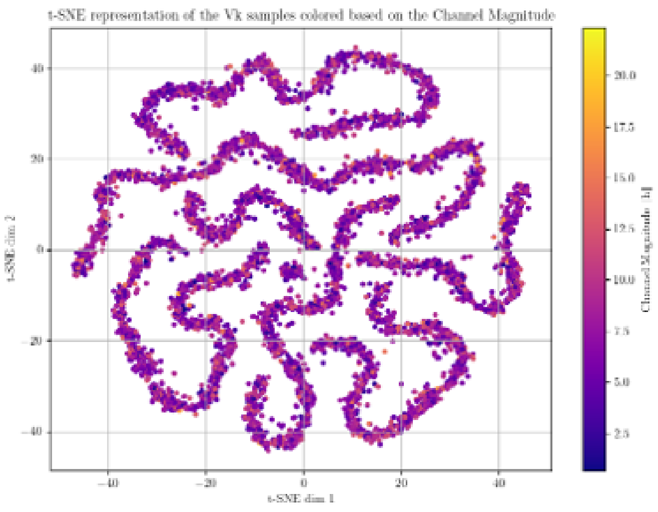


Figure 4.19: t-SNE representation of $\mathbf{v}_k$ for $\lambda = 100$ colored by $|\mathbf{h}_k|$

magnitude, treating the latter as irrelevant. Indeed, even if the magnitude changes, the embedding remains in the same region of the t-SNE graph: neighboring points—which correspond to similar precoding vectors—are associated with diverse channel magnitudes.

Colors in 4.20 are mostly mid-high, characterized by smooth, short-range gradients along the curve and a few localized low-magnitude dips (the purple segments). There are no abrupt jumps from strand to strand. Instead, solutions concentrate on a continuous family of $|\mathbf{t}_k|$ whose magnitude varies smoothly along the manifold, with only small adjustments along the path. A very clear gradient structure is evident here: there is a visible progression from purple to yellow along the curves. This indicates that the magnitude of $\mathbf{t}$ serves as a strong organizational feature. However, unlike the $\lambda = 2$ case where it separated discrete clusters, here it defines a continuous trajectory within the precoder space. Furthermore, this reflects the absence of an overhead constraint: for every minimal variation in the latent vector $\mathbf{t}$, there is a corresponding continuous and fluid variation in the precoder $\mathbf{v}$. Increasing

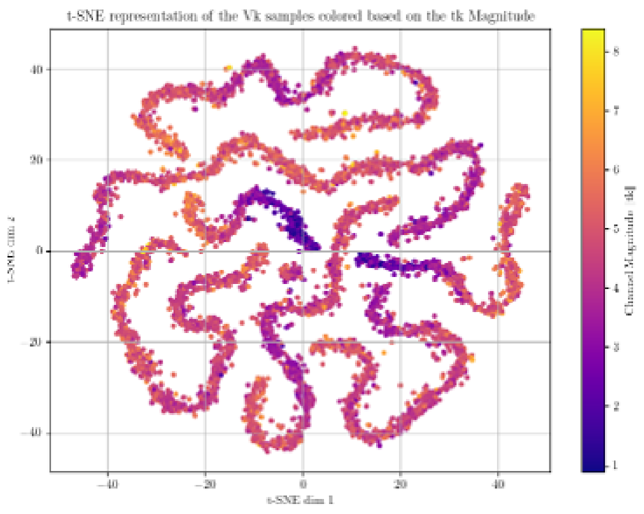


Figure 4.20: t-SNE representation of $\mathbf{v}_k$ for $\lambda = 100$ colored by $|\mathbf{t}_k|$

λ regularizes the feasible $\mathbf{v}_k$ solutions in a coherent trajectory, giving stable groups with smooth local variations; the milder weight allows different and alternative operating pockets for the precoding vector, with a “mode-hopping" behavior that is potentially useful for diversity but less smooth to control.

4.3 Pilots Analysis

In a classical communication system, the pilot matrix $\tilde{\mathbf{X}}$ is a fixed, pre-defined component. In our end-to-end framework, this assumption is removed. As explained in Section 2.4 and in Section 3.2.1, the $N_t \times L$ pilot matrix $\tilde{\mathbf{X}}$ is not fixed; it is a set of trainable parameters, jointly optimized with the encoder f_θ and precoder g_ϕ. We seek to answer the following question: What properties does the task-oriented pilot matrix, learned from data, possess? To open this "black box", we conducted three distinct analyses, comparing the Randomly Initialized pilots against the final Trained Pilots from both the $\lambda = 2$ (compressed) and $\lambda = 100$ (high-rate) models.

4.3.1 Orthogonality Analysis (Gram Matrix)

The first analysis tests whether the network discovers a solution of orthogonal pilots. To do this, we compute the $L \times L$ Gram Matrix $\mathbf{G}$ of the pilots, defined as

$$\mathbf{G} = \tilde{\mathbf{X}}^H \tilde{\mathbf{X}} \quad \in \mathbb{C}^{L \times L}$$

This matrix is a complete "correlation map" for the $L = 8$ pilot vectors. Each element $G_{i,j}$ is the inner product $\tilde{\mathbf{x}}_i^H \tilde{\mathbf{x}}_j$, which mathematically measures the alignment (correlation) between pilot $\tilde{\mathbf{x}}_i$ and pilot $\tilde{\mathbf{x}}_j$. Consequently, the diagonal elements ($i = j$) represent the power of each pilot ($\|\tilde{\mathbf{x}}_i\|^2$), which is constrained to $P = 1$. As seen in all three plots (4.21, 4.22, 4.23), the perfectly yellow diagonal confirms that the power constraint was correctly enforced. The off-diagonal elements ($i \neq j$) represent the cross-correlation ("inter-pilot interference"). A value of 0 (dark purple) signifies perfect orthogonality, which is the ideal for classical channel estimation. A non-zero value (blue/green) indicates a non-orthogonal basis.
We now compare the results for our three models:

1. Base Model: The plot in 4.21 serves as our control. As expected from a random, un-trained matrix, the off-diagonal elements show non-zero correlation ("bluish" squares, ≈0.4), indicating the pilots are not orthogonal at initialization.

2. High-Rate Model ($\lambda = 100$): The plot in 4.23 shows the "cleanest" matrix among the three. The off-diagonal elements are almost all dark purple, indicating the model has learned to generate a set of near-orthogonal pilot vectors. In classical communication theory, the prerequisite for maximizing rate is often to first obtain the most accurate CSI possible, i.e. minimizing the channel estimation MSE. As foundational texts like Tse and Viswanath [**Tse2005Fundamentals**] explain, the optimal "classical" solution for minimizing the estimation error is to use orthogonal sequences, as this mathematically isolates the channel paths and removes inter-pilot interference. Our result for

$\lambda = 100$ is the empirical proof that the E2E optimizer, when unconstrained by compression and tasked only with maximizing rate, autonomously rediscovers this fundamental engineering principle. It learns that the optimal path to maximum rate is to first solve the minimum-MSE estimation problem, and it does so by converging to orthogonal pilots.

3. Low-Overhead Model ($\lambda = 2$): The plot in 4.22 reveals a cleaner matrix than the random baseline, but it is visibly less orthogonal (more “polluted") than the $\lambda = 100$ model. This demonstrates that this model, when forced to solve the true trade-off, actively abandons the perfect orthogonality learned by the $\lambda = 100$ model. This is not a failure: the learned model is not perfectly orthogonal, and suggests this might not be needed when channel reconstruction is not directly the end-to-end goal.

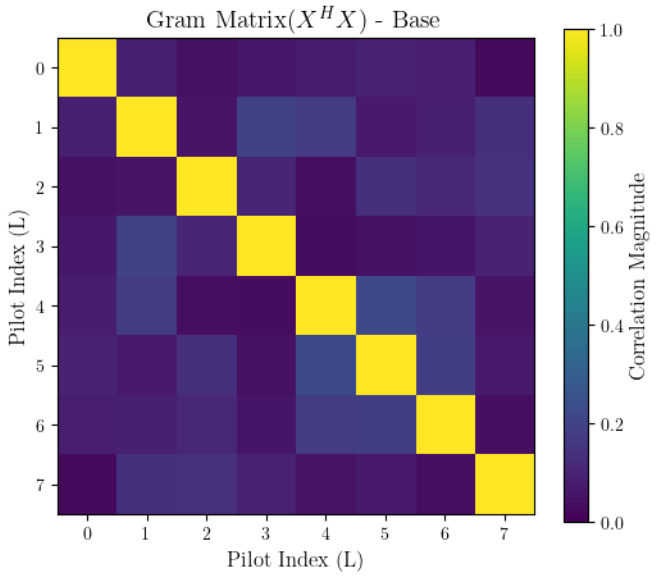


Figure 4.21: Gram Matrix for the Base Model.

4.3.2 Effective Dimensionality Analysis (SVD)

The scope of this second analysis is to test how many of the $L = 8$ available pilot dimensions the network has learned to use.
To do this, we compute the Singular Value Decomposition (SVD), a fundamental matrix factorization that decomposes our $N_t \times L$ pilot matrix $\tilde{\mathbf{X}}$ into $\mathbf{U\Sigma V}^H$. The critical component for this analysis is the diagonal matrix $\mathbf{\Sigma}$, which contains the $L = 8$ singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_8$. Each singular value σ_i mathematically

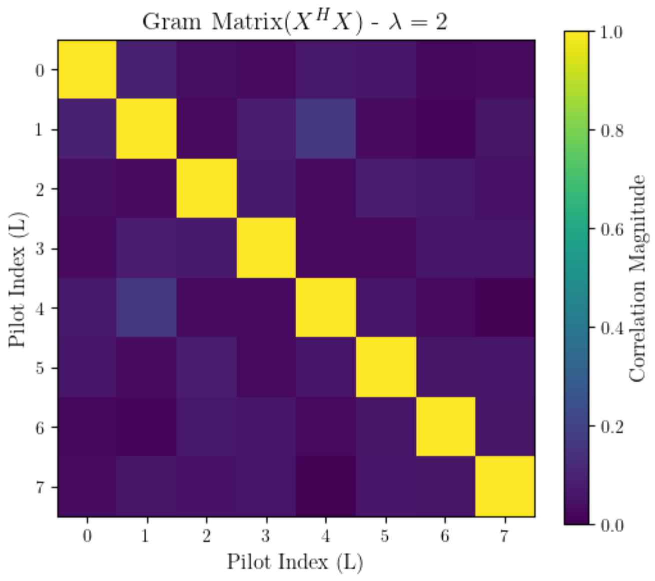


Figure 4.22: Gram Matrix for $\lambda = 2$ Model.

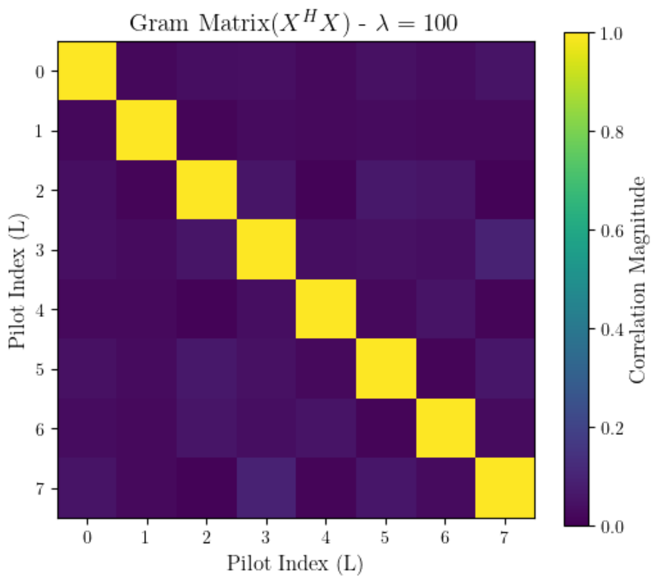


Figure 4.23: Gram Matrix for $\lambda = 100$ Model.

represents the “importance" associated with one of the matrix’s 8 fundamental dimensions. The number of non-zero singular values tells us the “effective rank" (i.e., how many dimensions the matrix truly uses). Our initial hypothesis is that a simple model might use all 8 dimensions (thus a full rank $\tilde{\mathbf{X}}$ matrix, with $L = 8$), while a more intelligent model might learn that the channel’s task relevant information exists in a lower-dimensional subspace and so “turn off" the redundant pilots, resulting n a low-rank solution.
We now compare the SVD plots for our three models.

A trend emerges moving from the baseline case, to $\lambda = 2$, to $\lambda = 100$.

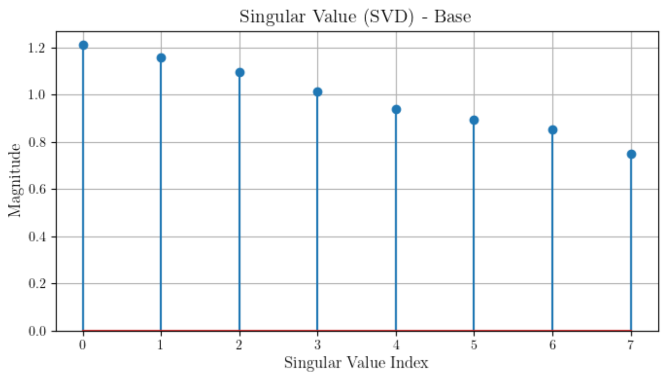


Figure 4.24: Singular Value Decomposition for the Base Model.

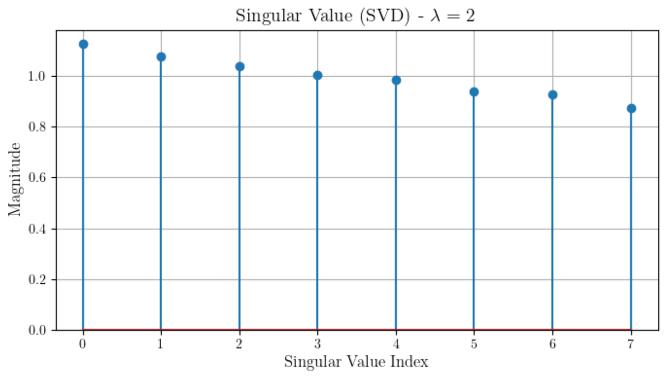


Figure 4.25: Singular Value Decomposition for $\lambda = 2$ Model.

The base case exhibits noticeable decay in its singular values, with a clear drop in magnitude, indicating that their relative importance is progressively lower. This is expected of a random matrix.
The $\lambda = 100$ model (4.26) shows the most noticeable change. The trend is nearly flat, with all eight singular values converging to a high, uniform magnitude (close to

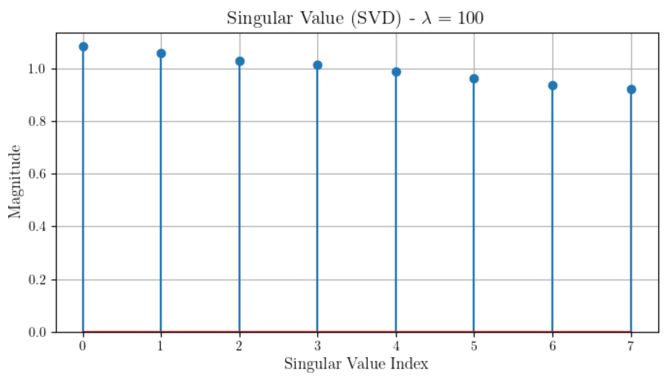


Figure 4.26: Singular Value Decomposition for $\lambda = 100$ Model.

1.0). This reflects a full-rank, near-orthogonal matrix: the model has well learned to make all $L = 8$ pilot dimensions equally important and linearly independent. This finding closely relates to our Gram Matrix analysis (4.23): the $\lambda = 100$ model learns to create a nearly orthogonal set of pilots, aiming for the perfect channel estimate. A property of an orthogonal matrix is that all its singular values are equal; furthermore, a set of orthogonal vectors is linearly independent, which confirms the model has learned a non-redundant sensing basis.
The $\lambda = 2$ model (4.25) lies between these two scenarios: the spectrum is flatter than the random baseline but less flat (more decaying) than the $\lambda = 100$ model. This confirms that this model learns a solution "less orthogonal" than the high-rate model.
Our conclusion from this SVD analysis is that both trained models learn to use a full rank ($L = 8$) pilots matrix, as evidenced by the 8 significant, non-zero singular values.

4.3.3 Physical Domain Analysis (Beampattern)

This final analysis tests whether the learned pilots, $\tilde{\mathbf{X}}$, have learned the physical statistics of the channel model (specifically, that the channel $\mathbf{h}$ exists only within a specific angular sector). This is a critical test of the end-to-end optimization, as the system's pilot power P is a finite resource. A baseline pilot sequence is "agnostic" to the channel's statistics; its beampattern "looks" in all directions, wasting significant power by sensing angles where no channel actually exists. This results in a weak signal ($\mathbf{h}_k^H \tilde{\mathbf{X}}$), and, consequently, in a low SNR for the received vector $\tilde{\mathbf{y}}$ at the UE. A fully learned model, however, should discover this statistical prior and optimize its pilots accordingly. The importance of learning this distribution is that the model can stop wasting power, focusing its entire sensing energy only on the sector where the channel actually lives. By concentrating all the available power P in the relevant subspace, the learned pilots induce a much stronger signal, maximizing the effective SNR of the input signal $\tilde{\mathbf{y}}_k$ to the encoder f_θ. A higher quality input, in turn, allows for a more accurate semantic compression (mapping to the $\mathbf{t}$ prototypes), a better precoder $\mathbf{v}$, and, ultimately, a higher final sum-rate $\mathcal{R}$.
To validate this, we compute the beampattern $B_\ell(\beta)$ for each of the $L = 8$ learned pilot vectors $\tilde{\mathbf{x}}_\ell$. The beampattern is a property of an antenna array that describes its directional gain as a function of a physical angle β. The N_t antennas at the BS transmit the N_t components of the pilot vector $\tilde{\mathbf{x}}_\ell$. It arises because the N_t antennas transmit the N_t components of the pilot vector with different, specific phases; in some directions, the phases align and the waves sum constructively (forming "main lobes"), while in others they sum destructively (forming "sidelobes").

The formula mathematically quatifies this physical effect:

$$B_\ell(\beta) = \left|\mathbf{a}_t(\beta)^\dagger\, \tilde{\mathbf{x}}_\ell\right|^2 \tag{4.1}$$

where $\tilde{\mathbf{x}}_\ell$ is the learned pilot vector, and $\mathbf{a}_t(\beta)$ is the ULA steering vector (from 2.2) that represents the ideal phase delays a signal would experience from direction β. The inner product $\mathbf{a}_t(\beta)^H\tilde{\mathbf{x}}_\ell$ thus measures the "alignment" of our learned pilot with that specific direction β. A high value for $B_\ell(\beta)$ signifies that our learned pilot $\tilde{\mathbf{x}}_\ell$ is strongly aligned with the direction β,and is, in fact, transmitting significant power towards that direction.
The analysis of the three plots reveals a clear learned strategy.
The beampatterns of the un-trained pilots (4.27) are unfocused and chaotic. Their energy is spread almost uniformly across the entire angular space, from -90° to +90°. This is a "naive" sensing strategy. Part of the power is wasted probing directions where the channel **h** is physically absent: the resulting signal **y** received by the user is weak. In fact, in our specific scenario, the relevant angular sector is defined between -30° and +30° (3.3).
The plots for the two trained models show a transformation (4.28 for $\lambda = 2$, 4.29 for $\lambda = 100$): both have learned to focus their energy. The main lobes of all the 8 pilots are now concentrated exclusively inside the [-30°, +30°] data sector (marked by the red dashed lines), while the sidelobes outside this sector are heavily suppressed. The E2E model, after observing thousands of channel realizations, has learned the channel statistics. Specifically, it has learned that probing outside the [-30°, +30°] sector is useless, and it therefore adapts its 64 complex pilot weights to concentrate its power P only on the relevant subspaces. This is the learned classical solution: the model's goal is to get a complete, low-MSE estimate of the entire channel **h**, so it illuminates the entire relevant sector uniformly.
As the $\lambda = 100$ model, also the $\lambda = 2$ model has learned to focus its energy inside the [-30°, +30°] sector. However, its beampatterns are slightly different: they are "spikier" and feature deep valleys (nulls) within the main sector. The "peaks" (high gain) might be the model focusing its power on angles it has learned are semantically crucial for classification. The valleys might be the model intentionally "turning off" in directions it has learned are semantically irrelevant to its task; moreover, this model might have learned to actively instruct the antennas of the BS to create destructive interference in specific angular directions it has learned to be semantically irrelevant for its task, focusing the finite power P only on the features that matter.

The first conclusion for this analysis is that the model learns the channel's physical statistics: it discovers that the channel exists only in the $[-30°, +30°]$ sector and focuses all the $L = 8$ pilots in this relevant subspace. Secondly, by showing 8

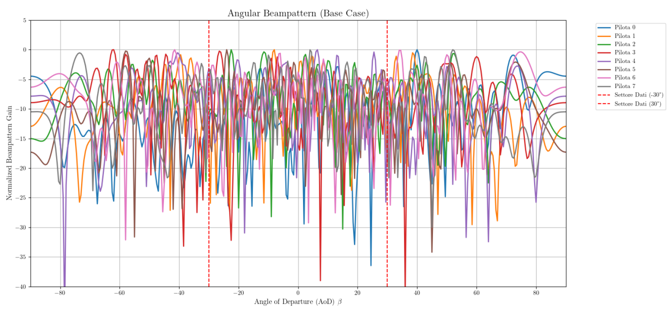


Figure 4.27: Pilots Angular Beampattern for the Base Model.

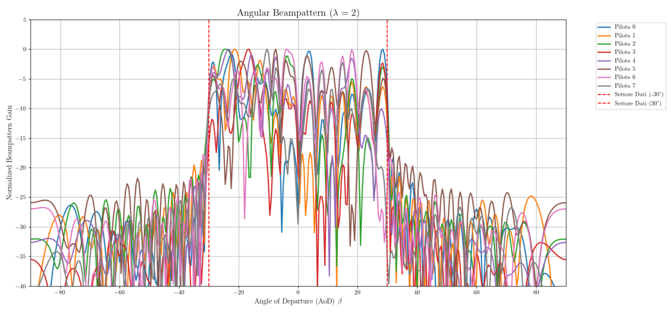


Figure 4.28: Pilots Angular Beampattern for $\lambda = 2$ Model.

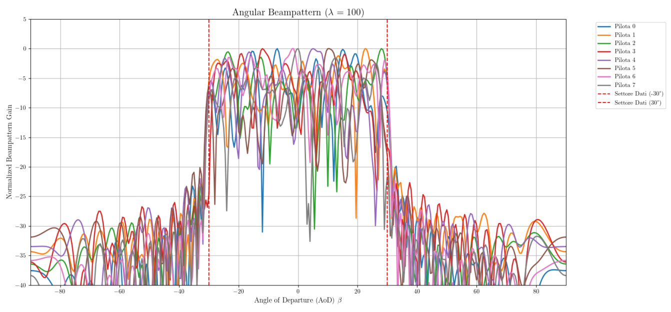


Figure 4.29: Pilots Angular Beampattern for $\lambda = 100$ Model.

significant singular values, the model demonstrates that it requires all $L = 8$ pilots to capture the necessary information within that sector. The model's strategy moght be to use its full-rank, focused pilots to deliver the cleanest possible signal ($\mathbf{y}$) to the UE; the UE's encoder, on its turn, will perform the task-oriented compression, quantizing the input signal in the learned $\mathbf{t}$ prototypes.

Chapter 5

Clustering

This chapter presents a clustering analysis applied to the vector representations within the system: the original channel vector **h**, the learned latent vector **t** and the learned precoder vector **v**. First, the methodological rationale for the choice of the DBSCAN algorithm is discussed Section 5.1, highlighting its advantages over traditional partitioning algorithms for this discovery-oriented task. Following this, a technical overview of DBSCAN [31] is provided Section 5.2, detailing its fundamental parameters, their selection strategy, and the validation methodology. The last part of the chapter, in Sections 5.5, 5.3, 5.4 is dedicated to the experimental analysis. Clustering results are presented for all three vectors, comparing the learned representations (generated by models trained with different λ weights) against the channel data.
Throughout this chapter, we present the analysis for a single representative user. It is important to note that we verified these results against the second user in the system ($K = 2$) and found them to be identical. This confirms that the learned strategy is symmetric and robust across users.

5.1 Methodological Rationale

The selection of DBSCAN for this analysis, as opposed to algorithms like K-Means, was a deliberate methodological decision. The primary research question was to discover how many significant, high-frequency patterns the network learned: we want the number of clusters k to be an unknown output of the analysis, not a required input.
DBSCAN is suited for this task for two key reasons. Firstly, unlike K-Means, which forces the data into a predefined number of clusters, k, DBSCAN is a density-based algorithm that determines the number of clusters organically based on the data structure. Secondly, DBSCAN differs from K-Means in its noise-handling capability.

In fact, K-Means is a partitioning algorithm, which means it is algorithmically required to assign every single data point to a cluster, and it has no mechanism for identifying or excluding outliers. In our context, noise is not to be imagined as random, spurious data; instead, it encompasses two distinct categories of non-clustered points. The first category consists of infrequently used vectors that the encoder learned but appear too rarely to satisfy the `min_samples` density threshold. The second category is transitional vectors: these are sparse data points lying in the low-density regions between the stable, high-frequency prototypes, and thus do not belong to any single cluster. The application of K-Means would lead to contamination, in which these rare and transitional vectors would be assigned to the nearest high-density cluster; therefore, the centroid of that cluster would no longer represent the true stable precoding vector prototype, but it would be a distorted value, influenced by the noise points. DBSCAN, as a density-based algorithm, defines a cluster as a region that satisfies a minimum density threshold (`min_samples`); therefore, the high-frequency codewords are correctly identified as dense clusters, whereas the rare vectors that fail to meet it are identified as noise and excluded from the cluster analysis.

5.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that defines clusters as contiguous regions of high-density points [31]. Its objective is to identify these dense regions and separate them from low-density regions, whose points are labeled as noise (outliers).

The algorithm relies on two fundamental parameters:

- ε **(epsilon)**: The radius that defines a "neighborhood" around a point.
- **`min_samples`**: The minimum number of points that must be within the ε-neighborhood of a point for that region to be considered "dense".

5.2.1 Fundamental Definitions

Based on these parameters, DBSCAN classifies every point in the dataset into one of three categories:

1. Core point: A point p is a *core point* if its ε-neighborhood, denoted as $N_\varepsilon(p) = \{q \in \mathcal{X} : d(p,q) \leq \varepsilon\}$, contains at least `min_samples` points (including p itself).

$$|N_\varepsilon(p)| \geq \texttt{min\_samples}$$

2. Border point: A point p is a *border point* if it is not a core point, but falls within the ε-neighborhood of at least one core point.

3. Noise point: A point p is a *noise point* (or outlier) if it is neither a core point nor a border point.

The clustering process involves connecting *core points* that are “reachable" from one another (i.e., within each other’s ε-neighborhood). A final cluster consists of all connected core points, plus all border points associated with them. Noise points are not assigned to any cluster.

5.2.2 Parameter Interpretation and Selection

The choice of ε and `min_samples` is crucial for the algorithm’s outcome.

- ε(Scale): Defines the spatial scale at which density is measured. A larger ε considers wider neighborhoods, lowering the required minimum density and leading to larger clusters and less noise.

- **`min_samples`**(Density Threshold): Defines the density threshold. Increasing it will only consider the densest regions, leading to more noise and potential fragmentation of clusters.

Specifically, these two parameters define a minimum local density threshold, ρ_{min}. A point is “dense” (core) if the density in its ε-neighborhood exceeds this threshold.

It is important to note that in high-dimensional spaces (such as the $d = 16$ latent space analyzed), the volume of the ε-neighborhood grows exponentially with the dimension ($V \propto \varepsilon^d$). Consequently, the algorithm can be highly sensitive to small variations in ε.

Estimating ε via the “K-distance Knee"

Since manually selecting ε is difficult, a heuristic known as the *k-distance knee plot* is often used.

1. Set $k =$ `min_samples`.

2. For each point in the dataset, calculate the distance to its k-th nearest neighbor.

3. These distances are then sorted in ascending order and plotted.

This plot typically shows a curve with a “knee" or “elbow," which is a point where the slope changes sharply. This point represents the transition between dense regions (with low k-distances) and sparse regions (where k-distances begin to rise rapidly). The ε value corresponding to this knee is a good candidate to represent the scale of the dense structures within the data.

5.2.3 Validation through Stability Analysis

The knee heuristic provides only an estimate. To validate the parameter choice, a stability analysis is performed by testing a range of ε values (e.g., a “sensitivity band" of $\pm 20\%$ around the estimated knee).

The goal is to find a “stability plateau": a range of ε values where the clustering outcome remains qualitatively consistent. This stability is assessed using internal validation metrics (implemented via the `scikit-learn` library [32]), recorded for each run:

1. Number of clusters: A stable number indicates a robust partition.
2. Noise fraction: A low and stable noise fraction is desired.
3. Silhouette Score: An internal index, computed only on non-noise points. It measures clustering quality (from -1 to 1). A high value indicates that points are close to their own cluster members (cohesion) and far from other clusters (separation).

If a plateau is observed in this sensitivity band (i.e., stable cluster count and noise fraction, with good quality scores), it can be concluded that DBSCAN has identified a genuine density-based structure. Otherwise, it is concluded that the data does not form robust, density-separated clusters.

5.3 Latent Representations t

We run DBSCAN across a small grid of ε values within the band and `min_samples` $=$ 32, recording the number of clusters and the noise fraction.
The selection of the parameter `min_samples` $= 32$ for the $d = 16$ dimensional latent vector $\mathbf{t}$ is a deliberate methodological choice. This value is based on a widely accepted heuristic for high-dimensional clustering, which recommends setting `min_samples` $\geq 2 \times d$. This conservative approach is crucial in high-dimensional spaces, as it provides a stricter definition of density and enhances robustness against noise, preventing the information of spurious or insignificant micro-clusters.

5.3.1 $\lambda = 2$ regime

The parameter sensitivity analysis for ε was conducted using a two-stage, coarser-to-fine approach. First, a coarse exploratory sweep was performed to identify regions of potential stability. The stability analysis was performed by sweeping over a range of ε values: this range was not chosen manually but was determined automatically using the k-distance knee plot heuristic, as described in Section 5.2. First, the k-distance was calculated for every point in the test dataset, with k set to `min_samples` = 32. These distances (the distance from each point to its 32nd nearest neighbor) were then sorted in ascending order to create the k-distance plot. Following the methodology, two heuristic boundary points were extracted from this curve: the Kneedle Point, and the Q95 Point. The first one is the point where the curve bends the most (the “knee"), after excluding the initial flat segment where the distances are very small; the second one is the 95th percentile of the k-distances, serving as a conservative upper bound. This automated process defined the sweep interval of ε, which is then sampled at 30 linearly-spaced points. The DBSCAN algorithm was run on the full dataset for each of these 30 ε values to generate the final values presented in the table.
This initial pass revealed that the number of clusters was very unstable at very small and large ε values, but suggested a promising region of interest between $\varepsilon \approx 0.06$ and $\varepsilon \approx 0.12$. Second, a fine-grained validation sweep was conducted within this specific range to precisely map the stability of the solution. The results of this high-resolution analysis are presented in 5.1, which confirms the existence of a robust stability plateau.
5.1 provides a validation of the clustering structure discovered in the 16-dimensional latent space. The plot displays a 2D t-SNE projection of the latent vector **t**, where each point is colored according to the cluster label assigned by DBSCAN. The parameters used (ε=0.0871, `min_samples`=32) were selected directly from the stability plateau analysis (5.1, 5.3.1). The 11 clusters identified by DBSCAN in the 16D space map perfectly to distinct and well-separated islands in the 2D t-SNE projection. This confirms the good Silhouette score (≈ 0.87) observed in the stability analysis: the “islands" are extremely compact (high intra-cluster cohesion), and are separated by significant empty space (high inter-cluster separation). The noise points (colored gray) are not randomly distributed. Instead, they primarily populate the sparse regions between the dense cluster islands: this suggests they might represent transitional or less-frequent latent prototypes, rare latent representations, filtered out by the clustering parameters. The quantitative distribution of samples across these identified clusters and the noise category is reported in 5.2. It reveals that the learned set is highly skewed and non-uniform. We observe a single dominant prototype (Cluster 3) which accounts for 26.48% of the dataset, suggesting it captures a statistically prevalent or 'default' state of the channel. The remaining

10 prototypes act as 'specialized' states, with frequencies ranging between 3% and 8%.
This regime can be described as a mechanism of learned semantic compression: these 11 clusters are not arbitrary; they are defined by their relevance to the final task, i.e. precoding. Through the end-to-end optimization, the network has learned that the varied channel realizations within a single cluster are semantically equivalent, meaning that they all lead to the same precoding decision to maximize the rate $\mathcal{R}$. The model acts on this discovery by learning to map all these fine-grained channel variations (which are irrelevant to the final task) to the same, single prototype vector: only the task-relevant information is preserved. This entire mechanism is learned, as the network autonomously discovers this clustered structure as the optimal solution to the trade off posed by the loss function.

Table 5.1: Sweep Results on $\mathbf{t}$ for $\lambda = 2$ regime (Coarse View) - ε Range: [0.013, 0.305], `min_samples`: 32

ε	**`min_samples`**	**Clusters**	**Noise (%)**	**Silhouette**
0.013	32	8	0.780	0.884
0.023	32	13	0.526	0.863
0.034	32	13	0.413	0.854
0.044	32	13	0.357	0.823
0.055	32	13	0.315	0.789
0.065	32	11	0.275	0.797
0.076	32	11	0.248	0.866
0.086	32	11	0.222	0.856
0.096	32	11	0.198	0.842
0.107	32	11	0.178	0.830
0.117	32	11	0.151	0.767
0.128	32	8	0.129	0.622
0.138	32	7	0.113	0.541
0.149	32	6	0.101	0.453
0.180	32	2	0.062	0.302
0.190	32	1	0.058	—
0.201	32	1	0.052	—
0.211	32	1	0.047	—
0.222	32	1	0.043	—
0.295	32	1	0.017	—
0.305	32	1	0.014	—

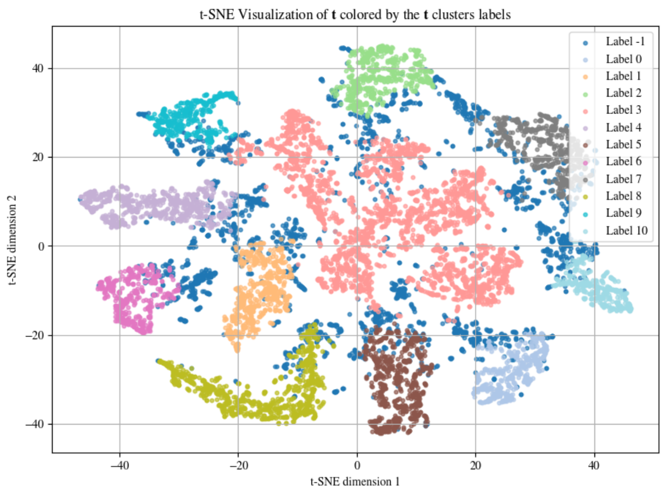


Figure 5.1: t-SNE representation colored by the DBSCAN clustering results for the Latent Vector $\mathbf{t}$ for $\lambda = 2$.

Table 5.2: Number of samples for each Cluster Label (User 1)

Label	**Samples**
3	2648
8	783
5	614
4	602
1	577
7	549
2	503
0	417
6	416
9	349
10	327
-1	2215

5.3.2 λ=100 regime

5.3 presents the DBSCAN clustering results for the latent vector $\mathbf{t}$ produced by a model trained with $\lambda = 100$. To perform this analysis, we applied the same two-stage, coarse-to-fine sweep strategy defined previously for the $\lambda = 2$ regime (Section 5.3.1): the ε range boundaries were determined using the weighted k-distance plot, and the resulting interval was sampled at 30 linearly-spaced points.
The results from this analysis stand in contrast to the previously analyzed model. A total lack of a stable clustering solution is observed. The number of clusters immediately disintegrates: it begins at 11, jumps to 21, drops to 17 and 7, and then rapidly collapses to a single cluster. This chaotic behavior is coupled with two other indicators of unstructured data. Firstly, there is an extremely high noise fraction, starting at 90% for the smallest ε. Secondly, the Silhouette scores are poor (starting at a mediocre 0.412) and degrade instantly, and the ε values are more than an order of magnitude larger than those of the stable plateau found in the previous model ($\varepsilon \approx 0.013 - 0.305$). This indicates that data is significantly more sparse and lacks the nearly "quantized" structure seen previously. In this high-rate regime, the encoder does not converge to a small, finite set of representative prototype vectors; instead, it uses the entire 16-dimensional space to produce a continuous and unstructured set of points. This result is a direct consequence of the loss function's objective. By prioritizing the achievable rate $\mathcal{R}$ almost exclusively, the $\lambda = 100$ model has learned not to compress. To achieve its goal, the BS requires highly detailed and precise channel information for every realization: the resulting continuous, non-clustered latent space implies that the user must feed back a

unique, high-precision vector **t** for every channel instance. This procedure demands a large feedback overhead $\mathcal{O}$.
This stands in opposition to the $\lambda = 2$ regime of “learned semantic compression", in which the system learned to quantize the entire channel space into 11 stable prototype vectors: this would allow the user to feed back a simple, low-bit index and to exploit a minimal feedback overhead. We can therefore conclude that the clustered structure is not an inherent property of data. It is a behavior learned by the network, and it is only present when the optimization is forced to balance the critical trade off between achievable rate and feedback overhead.

Table 5.3: Sweep Results on **t** for $\lambda = 100$ regime - ε Range: [0.918, 2.198], `min_samples`: 32

ε	**`min_samples`**	**Clusters**	**Noise (%)**	**Silhouette**
0.918	32	11	0.902	0.412
0.989	32	21	0.792	0.347
1.060	32	17	0.688	0.263
1.131	32	7	0.616	0.126
1.202	32	3	0.559	0.101
1.274	32	2	0.516	0.075
1.345	32	2	0.475	0.074
1.416	32	2	0.433	0.072
1.487	32	1	0.392	—
1.558	32	1	0.352	—
1.629	32	1	0.311	—
1.700	32	1	0.267	—
1.771	32	1	0.227	—
1.842	32	1	0.186	—
1.913	32	1	0.151	—
1.984	32	1	0.116	—
2.055	32	1	0.088	—
2.127	32	1	0.066	—
2.198	32	1	0.050	—

5.4 Precoding Vectors v

We applied the same stability analysis of Section 5.3 to the precoding vector **v**.

5.4.1 λ=2 regime

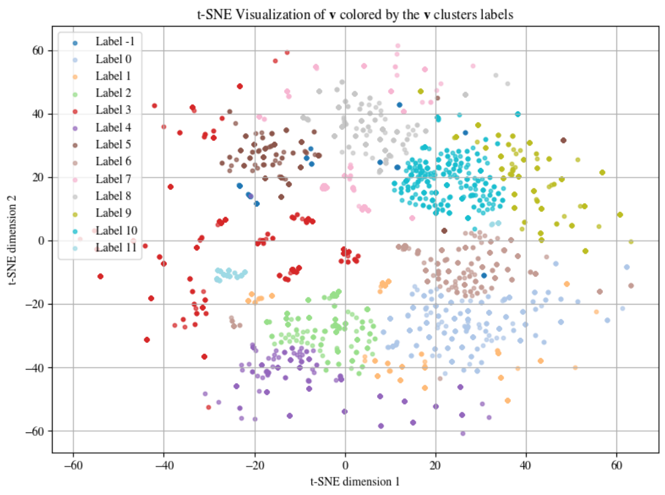


Figure 5.2: *t*-SNE representation colored by DBSCAN clustering results for the Latent Vector $\mathbf{v}$ for $\lambda = 2$.

To analyze the BS's learned policy, we simulated the full end-to-end precoding pipeline using a batch size of 10,000. Over the test set of 10,000 channel realizations, we analyze the generated precoding vectors $\mathbf{v}_k \in \mathbb{C}^{N_t \times 1}$ associated with the k-th user (where $N_t = 64$). The model trained with $\lambda = 2$ produces only M=531 unique precoding vectors in the batch, while the remaining ones are exact duplicates. This finding is the proof that the BS network has learned a finite, discrete set of "precoding actions" rather than a continuous mapping, consistent with our hypothesis of a structured compression mechanism. The next key question is how these $M = 531$ unique precoding vectors are used.
A frequency distribution analysis is visualized through a set of complementary plots:

- The Rank-Frequency plot (5.4) reveals that the usage of precoding vectors follows a heavy-tailed distribution. The y-axis (Count) represents the absolute frequency, i.e., how many times a specific unique precoding vector appears in the dataset. The x-axis (Rank) orders these vectors by popularity: Rank 1 is the single most frequent vector, Rank 2 is the second most frequent, and so on. The resulting curve clearly shows a small "head" of frequently used prototypes (high count, low rank), followed by a massive "long tail" of hundreds of specialized vectors that are selected only rarely (often just once or twice).

- The Cumulative Coverage plot (5.3) illustrates the cumulative probability mass function of the top-k most frequent precoding vectors. In this case, we set k=20, considering the 20 most frequent codewords. The curve exhibits a sharp initial rise, reaching a cumulative coverage of 0.480 (48.0%) with just the top-20 unique vectors. This result signifies that nearly half of all precoding decisions made by the network (in a dataset of 10,000 samples) map to a tiny subset ($\approx$ 4%) of the 531 available prototypes. This confirms the highly concentrated nature of the learned policy, where a small "set" of robust strategies handles the vast majority of channel scenarios: this is precisely the structure that will be discovered by our DBSCAN analysis.

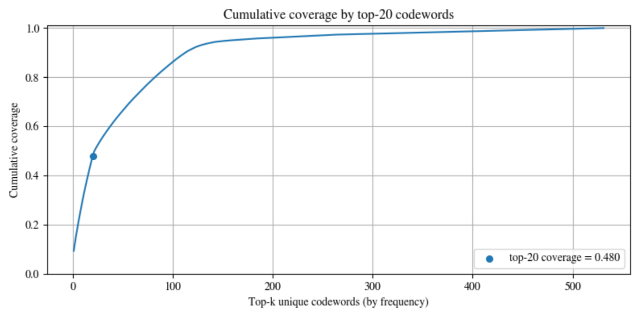


Figure 5.3: Cumulative Coverage by the top-20 codewords

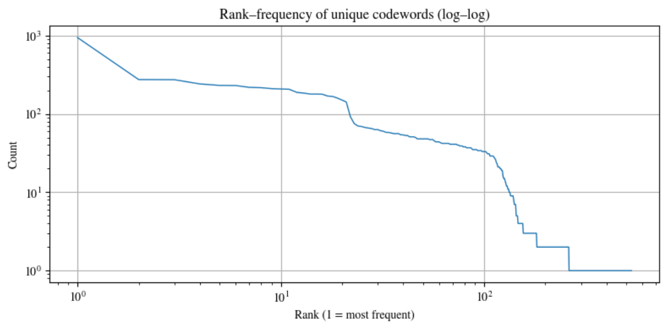


Figure 5.4: Rank-frequency of unique codewords

The table shown in 5.4 presents the core findings from the DBSCAN stability analysis on the 64 dimensional precoding vector $\mathbf{v}$, conducted using the identical ε-sweep methodology (based on the k-distance heuristic) described in Section 5.3.1. The most immediate finding is the scale of the analysis. The BS network is not producing a continuous output. Like the UE's encoder, it has learned to act as a quantizer, mapping inputs to a discrete set of prototype precoder vectors. At a fine ε scale, the system reveals three stable plateaus: at 18 clusters (from $\varepsilon \approx 0.083$ to

$\varepsilon \approx 0.150$); this is immediately followed by another robust plateau at 17 clusters (Silhouette $\approx$ 0.87-0.90 and a very low noise fraction), and at 12 clusters (from $\varepsilon \approx$ 0.299 to $\varepsilon \approx 0.449$). The BS network does not just mirror the 11 clusters from the latent space $\mathbf{t}$; it maps the 11 clusters in $\mathbf{t}$ to richer set of 12-to-17 distinct "precoding actions". The quantitative distribution of samples across these identified clusters is reported in 5.5.
5.2 provides a visual validation of the clustering structure discovered in the 64-dimensional space. The plot displays a 2D t-SNE projection where each point is colored according to the cluster label assigned by DBSCAN, using the parameters (ε=0.299, `min_samples`=128) selected directly from the stability plateau analysis (5.4, 5.4.1). Comparing this plot to the t-SNE projection of $\mathbf{t}$ (5.1), while the $\mathbf{t}$ clusters were dense and compact islands, the $\mathbf{v}$ clusters are more sparse and resemble a cloud. This could take in consideration the multi-user nature of the BS processing. A $\mathbf{t}$ cluster represents a simple, single-user channel state. A $\mathbf{v}$ cluster, in contrast, represents a precoding action that the BS chooses in response to the joint information from all users.
A contingency matrix is a data visualization tool used to analyze the relationship between two categorical variables. The Contingency Table shown in 5.5 visualizes the mapping of the 11 stable $\mathbf{t}$-clusters (the 11 "codeword" categories learned by the UE's encoder, arranged on the y-axis) to the 12-cluster of the $\mathbf{v}$ vector (the 12 "precoder" categories learned by the BS, on the x-axis). Each cell in the matrix displays two key pieces of information: the absolute count (the number of samples that belong to both the cluster in that row and the cluster in that column), and the row-normalized percentage (percentages calculated based on that row's total only, not the grand total of the whole table). The analysis reveals two distinct types of behaviors: deterministic and contextual mappings. In the first scenario, the vast majority of samples are handled by a simple and direct lookup policy. For example, $\mathbf{t}-$cluster 1 maps 100% of its members to $\mathbf{v}$-cluster 6, and $\mathbf{t}$-cluster 2 maps 100% of its members to $\mathbf{v}$-cluster 9. This demonstrates that, for most inputs, the system has learned an efficient and quantized "codebook". The second case is the exception: the matrix also reveals a non-deterministic mapping for $\mathbf{t}$-cluster 7. This row is "smeared" across two different $\mathbf{v}$-clusters: 90.7 % of its samples are mapped to $\mathbf{v}$-cluster 1, while the remaining 9.3% are mapped to $\mathbf{v}$-cluster 11.

We could interpret the 10 deterministic rows as "simple" channel states, and row 7, the "exception", as a more complex or ambiguous one: a one-to-many, non-deterministic mapping. The final precoder choice is not solely determined by $\mathbf{t}$-cluster 7, but it might be contingent on other information (for example, the channel report from the other user in the system), which causes this smeared distribution. In fact, while the $\mathbf{t}$-cluster is generated by a single user based only on its own channel, the $\mathbf{v}$-cluster, i.e. the precoder, is chosen by the BS, which sees the reports from all users. The 9.1 % split might represent the cases where the

BS has to replace its default precoder and choose an alternative (for example, to manage interference).

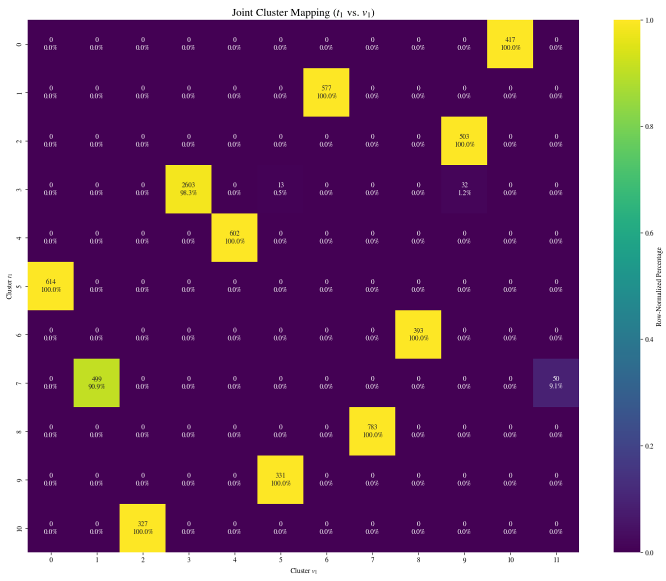


Figure 5.5: Contingency Table representing the mapping between the clusters of $\mathbf{t}$ and the clusters of $\mathbf{v}$ for $\lambda = 2$.

Before analyzing the joint multi-user policy, we must first understand the frequency distribution of the learned prototypes for each user independently. 5.5 and 5.6 present the absolute sample counts for each of the 12 $\mathbf{v}$-clusters found for User 1 and User 2, respectively (including noise points, labeled -1). The primary discovery is that the network's learned policy is not uniform. A “balanced" policy would use all 12 prototypes roughly equally. These tables, instead, prove that the network has learned a highly skewed strategy that relies on a few fundamental prototypes and on many “specialist" ones. For User 1, Cluster 3 accounts for 3094 samples ($\sim$ 31.3 % of the non-noise data). For User 2, Cluster 2 accounts for 3054 samples ($\sim$ 30.9 %). One possible interpretation could be that the network heavily relies on this single dominant prototype as its “default" action. In contrast, the network also learns “niche" solutions. For User 1, Cluster 11 (180 samples) and, for User 2, Cluster 8 (177 samples) are the less populated clusters: we could interpret these clusters as specialized solutions, each used in less than 2% of cases, probably reserved for rare interference scenarios.
The second finding comes from comparing the two tables. This comparison reveals that the two cluster count distributions are statistically identical. Both distributions reveal the same underlying structure: a dominant cluster with $\approx$ 3000 samples

Table 5.4: Sweep Results on **v** for $\lambda = 2$- ε Range: 0.033, 0.482], `min_samples`: 128

ε	`min_samples`	**Clusters**	**Noise (%)**	**Silhouette**
0.033	128	29	0.230	0.896
0.050	128	24	0.142	0.930
0.067	128	20	0.106	0.911
0.083	128	18	0.101	0.892
0.010	128	18	0.092	0.891
0.116	128	18	0.069	0.897
0.133	128	18	0.058	0.894
0.150	128	18	0.046	0.890
0.166	128	17	0.036	0.875
0.183	128	17	0.025	0.868
0.200	128	17	0.024	0.868
0.216	128	17	0.024	0.867
0.233	128	17	0.023	0.867
0.249	128	17	0.021	0.866
0.266	128	15	0.020	0.811
0.283	128	13	0.019	0.769
0.299	128	12	0.013	0.848
0.316	128	12	0.012	0.848
0.332	128	12	0.012	0.847
0.349	128	12	0.011	0.846
0.366	128	12	0.011	0.846
0.382	128	12	0.010	0.845
0.399	128	12	0.009	0.844
0.416	128	12	0.009	0.844
0.432	128	12	0.007	0.841
0.449	128	12	0.006	0.840
0.465	128	9	0.002	0.648
0.482	128	8	0.000	0.593

Table 5.5: Number of samples for each Cluster Label (User 1)

Label	Samples
3	3094
7	905
6	751
0	746
9	740
4	705
1	622
10	610
2	544
8	524
5	455
11	180
-1	124

Table 5.6: Number of samples for each Cluster Label (User 2)

Label	Samples
2	3054
5	906
7	740
0	726
3	706
10	677
11	655
6	591
9	558
1	558
4	538
8	177
-1	114

(3094 for User 1 vs 3054 for User 2), a single niche cluster with ≈ 180 samples (180 for User 1 vs 177 for User 2), and a series of intermediate clusters with similar counts (for User 1, the counts of $\{905, 751, 746, 740\}$ are comparable to the second User's $\{906, 740, 726, 706\}$). The only apparent difference is in the naming of the labels: for example, the largest prototype is labeled "Cluster 3" for User 1 but

“Cluster 2" for User 2. This is a benign and expected labeling artifact. It occurs simply because the DBSCAN algorithm was run independently on the two datasets and assigned its arbitrary numerical labels in a different orders.

Table 5.7: Internal Composition of the $\mathbf{v}$-Clusters (User 1).

Cluster ID	**Total Samples**	**Unique Vectors**	**Dominant Prototype**	**Dominance (**
3	3094	48	ID 27	30.4%
7	905	33	ID 352	30.2%
6	751	50	ID 263	28.0%
0	746	47	ID 188	27.6%
9	740	27	ID 345	31.2%
4	705	42	ID 134	30.9%
1	622	30	ID 159	30.2%
10	610	65	ID 446	29.2%
2	544	54	ID 267	29.2%
8	524	40	ID 425	34.2%
5	455	30	ID 153	33.0%
11	180	18	ID 396	32.2%
-1	124	47	ID 319	26.6%

To validate the quality of the discovered clusters, we analyzed their internal composition in terms of unique vectors. As established, the dataset contains significant redundancy ($\approx 95\%$ of the total samples are duplicates). A critical requirement for a robust clustering is consistency, i.e. identical inputs (duplicates) must be assigned to the same cluster. Our analysis confirms that this condition is met. For every unique vector in the dataset, all of its duplicates were assigned to the same cluster (or all to noise). Furthermore, 5.7 reveals that each cluster is consists of a single Dominant Prototype, which typically accounts for $\approx 30\%$ of the cluster’s total mass. Surrounding this one, we find a set of 20-60 less frequent unique vectors. This suggests that the "Precoding Action" represented by a cluster is not a single rigid vector, but a family of closely related strategies. The network relies on a primary vector (the Dominant Prototype) for the most common variations of that channel state, but it has also learned a library of fine-grained variations to handle specific micro-variations in the interference environment, all while keeping "Cluster ID decision" constant.

5.8 details the clustering assignment for the top-20 most frequent unique vectors, a subset accounting for the highest density peaks in the dataset. For every high-frequency vector, from the dominant ID 27 (941 copies) to the 20th ranked ID 153 (150 copies), the algorithm consistently assigns all copies to a single, unique cluster label. Cluster 3, the largest cluster, is composed of multiple distinct, highly-frequent vectors (e.g., ID 27, ID 143, ID 274, ID 369, ID 47, ID 41, ID 57, ID 89, ID 34, ID 58) which occupy 10 of the top-20 ranks. This proves that the whole cluster might represents a broad, densely populated region containing a rich family of related precoding strategies.

Table 5.8: Clustering Consistency Validation on the Top-20 Most Frequent Unique Vectors.

Rank	**Vector ID**	**Total Copies**	**Assigned Cluster**	**N_{labels} Found**
1	27	941	3	1
2	143	274	3	1
3	352	273	7	1
4	274	241	3	1
5	345	231	9	1
6	369	230	3	1
7	134	218	4	1
8	47	216	3	1
9	263	210	6	1
10	41	208	3	1
11	188	206	0	1
12	159	188	1	1
13	57	184	3	1
14	89	179	3	1
15	425	179	8	1
16	446	178	10	1
17	34	169	3	1
18	58	167	3	1
19	267	159	2	1
20	153	150	5	1

A possible interpretation could be that the BS uses the encoder's compressed feedback to select a "Precoding Prototype" (one of the 12-18 $\mathbf{v}$-clusters), which defines the general spatial direction of the beam. The joint information from all the users is used to calculate a continuous, fine-grained adjustment to the selected prototype, ensuring proper interference management.

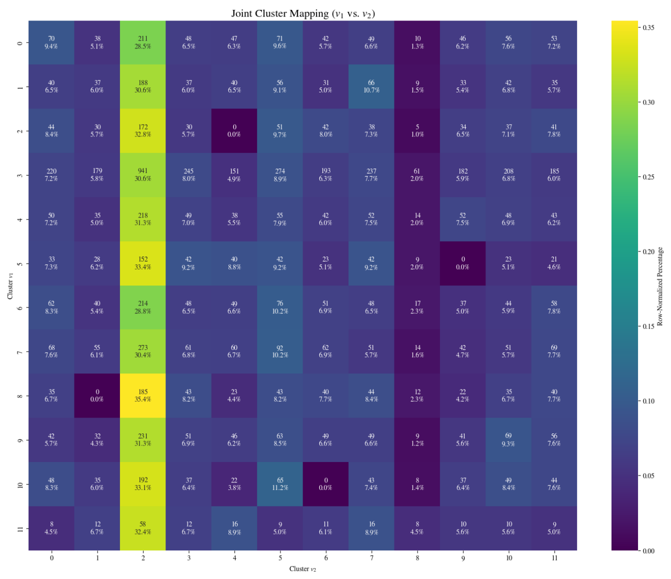


Figure 5.6: t-SNE representation of $\mathbf{h}_k$

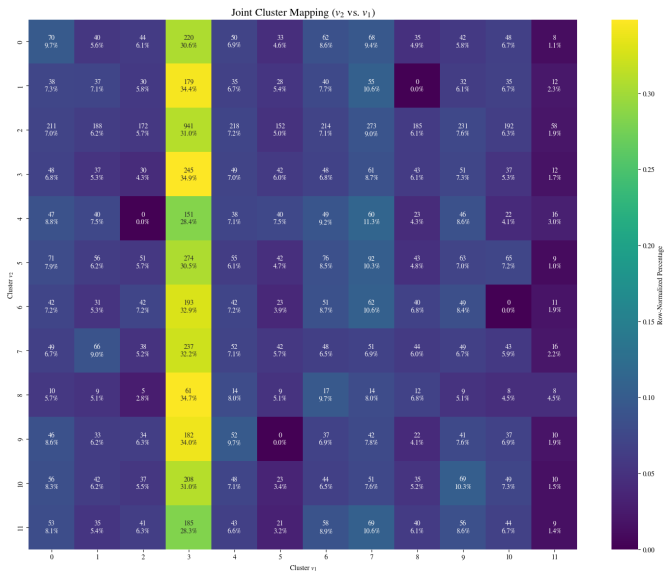


Figure 5.7: t-SNE representation of $\mathbf{h}_k$

To analyze the learned multi-user precoding policy, we plot two complementary contingency matrices. 5.6 presents the Contingency Matrix mapping the 12 learned $\mathbf{v}_1$ clusters (y-axis) against the 12 learned $\mathbf{v}_2$ clusters (x-axis), while 5.7 shows the inverse.
This skewed distribution (estabilished in 5.5 and 5.6) is the direct cause of the “magnet effect" we observe in both matrices. 5.6 shows that, regardless of the $\mathbf{v}_1$ state, the most probable outcome for User 2 is to be assigned to its most common prototype. 5.7 shows the same logic in reverse: the most probable outcome for User 1 is its default prototype, $\mathbf{v}_1 = 3$, regardless of the $\mathbf{v}_2$ state. The pair ($\mathbf{v}_1 = 3$, $\mathbf{v}_2 = 2$) represents the most common scenario, accounting for 941 samples (nearly 10% of the total test data). An interpretation could be that the BS network, when faced with two identical users, exploits a learned symmetric “default" strategy, i.e. it assigns the single most commmon prototype to both users.
Finally, this analysis defines the BS network's policy as “non-continuous" and adaptive. A “uniform" policy would show each row as uniformly colored; instead, in our plots each row is a mixture of bright yellow (high-probability “default" states) and dark blue/purple (low-probability states) cells. A possible interpretation could be that the network has not learned a finite set of joint strategies. It relies on a dominant “default" pair ($\mathbf{v}_1 = 3$, $\mathbf{v}_2 = 2$) for the most common scenarios, but for all other complex interference cases ($\approx$ the 90%), it abandons this default and actively computes a continuous, adaptive joint solution.

5.4.2 λ=100 regime

5.9 presents the clustering results for the 64-dimensional precoding vector $\mathbf{v}$, generated by the $\lambda = 100$ model. We applied the same heuristic ε-sweep methodology (Section 5.3.1) to determine the analysis range and perform the stability test.
The DBSCAN results show a complete failure to find any stable, quantized structured. Firstly, there is no stability plateau: the number of clusters is chaotic (jumping from 10 to 24 to 19) before its inevitable collapse to 1, proving the algorithm is not discovering any meaningful, robust structure. Secondly, the analysis begins at an ε value over 40 times larger than that of the previous model ($\varepsilon \approx 4.4$ vs. $\varepsilon \approx 0.1$). This confirms the data is not a set of “discrete and repeated codewords" but it is a continuous, sparse manifold. The Silhouette score is mediocre (0.59) and it collapses instantly, while the initial noise fraction is extremely high (83 %), confirming the data is sparse and vast majority of it does not belong to any clustered region.
In the $\lambda = 2$ regime, the clustered $\mathbf{v}$ space proved that the BS had learned a finite “set" of precoding actions: it learned a small set of robust and optimal precoders and chose the best one.
The $\lambda = 100$ model shows a continuous space, which means that the BS has not

learned a finite set; instead, it has learned a more complex policy. For every single unique vector **t** it receives, it performs a unique, complex calculation to generate a "one-of-a-kind" precoder **v**. The complete picture is that this model achieves high rate exploiting overhead both to send the continuous **t** and for the complex precoding policy to generate **v**.

Table 5.9: Sweep Results on **v** for $\lambda = 100$- ε Range: [4.447, 7.228], `min_samples`: 128

ε	**`min_samples`**	**Clusters**	**Noise (%)**	**Silhouette**		
4.447	128	10	0.832	0.593		
4.579	128	14	0.715	0.544		
4.711	128	24	0.510	0.499	900	0.826
4.844	128	19	0.396	0.411	580	1.145
4.976	128	17	0.325	0.358	466	1.429
5.109	128	20	0.247	0.292	386	1.460
5.241	128	15	0.190	0.241	377	1.773
5.374	128	11	0.156	0.203	363	2.149
5.506	128	8	0.131	0.170	371	2.559
5.639	128	9	0.102	0.162	350	2.443
5.771	128	7	0.093	0.141	353	2.822
5.904	128	8	0.066	0.117	307	2.645
6.036	128	6	0.057	0.122	355	3.156
6.169	128	5	0.048	0.102	348	3.451
6.301	128	4	0.042	0.082	342	3.877
6.434	128	3	0.034	0.064	342	4.413
6.566	128	3	0.026	0.063	342	4.424
6.700	128	2	0.019	0.048	341	5.025
6.831	128	1	0.016	—	—	—
6.964	128	1	0.013	—	—	—
7.096	128	1	0.008	—	—	—
7.228	128	1	0.007	—	—	—

5.5 Channel Realizations

A critical hypothesis of this work is that the original channel vector $\mathbf{h} \in \mathbb{C}^{64}$ should not be clusterizable. This hypothesis is based on its fundamental generating process. All channel realizations are generated according to the same geometric multipath model (2.1). In our simulation setup, all samples from all users are drawn from

the same statistical distributions: they share the same angular support, the same number of paths, and identically distributed complex path gains. Because there is no distinction between sectors or propagation conditions, data originates from a single, continuous law rather than from a mixture of distinct contributions. The resulting vectors form a smooth, non-linear array manifold, and not a set of discrete, separable groups.
To experimentally validate this theoretical assumption, we subjected $\mathbf{h}$ to the same sweep and visual analysis as the learned vectors $\mathbf{t}$ and $\mathbf{v}$.

First, our quantitative clustering analysis confirmed the hypothesis (5.10): the DBSCAN stability sweep failed to find any stable, dense clusters.
The Cluster column is 1 for the entire sweep: the algorithm never, at any scale, discovers a stable partition with more than one cluster. The algorithm simply transitions from classifying most of the data as noise (for ε lower than 3.643) to absorbing the entire dataset into a single, contiguous group once the radius is large enough. The Noise column confirms this: the analysis begins with 83.4% of the data still classified as noise. This demonstrates the extreme sparsity of the $\mathbf{h}$ data, as most points do not have 128 neighbors even within this large radius. The Silhouette Score is undefined across the entire range: this is the logical consequence of the cluster count never exceeding 1. Since the Silhouette score measures inter-cluster separation (separation between clusters), it cannot be computed when only one cluster exists.

Second, to visually support why this failure occurred, we employed dimensionality reduction. We again rejected PCA, as it is a linear projected technique fundamentally unsuited for the non-linear structure of $\mathbf{h}$. We choose t-SNE, and our logic was as follows: if even t-SNE cannot visually separate any distinct groups, then we can confidently conclude that no clusters exist.
5.8 shows the resulting t-SNE plot. The visualization reveals a single, contiguous point cloud with the absence of dense “islands" and “empty space". The plot provides a visual proof that $\mathbf{h}$ is a non-clusterizable continuum.

Table 5.10: Sweep Results on $\mathbf{h}$ - ε Range: [3.643,11.562], `min_samples` = 128

ε	`min_samples`	**Clusters**	**Noise (%)**	**Silhouette**
3.643	128	1	0.834	—
3.916	128	1	0.774	—
4.190	128	1	0.714	—
4.463	128	1	0.650	—
4.736	128	1	0.575	—
5.009	128	1	0.496	—
5.282	128	1	0.422	—
5.555	128	1	0.362	—
5.828	128	1	0.302	—
6.374	128	1	0.204	—
6.920	128	1	0.133	—
7.466	128	1	0.083	—
8.012	128	1	0.048	—
8.559	128	1	0.026	—
9.105	128	1	0.014	—
9.651	128	1	0.008	—
10.197	128	1	0.004	—
10.470	128	1	0.003	—
11.016	128	1	0.002	—
11.562	128	1	0.001	—

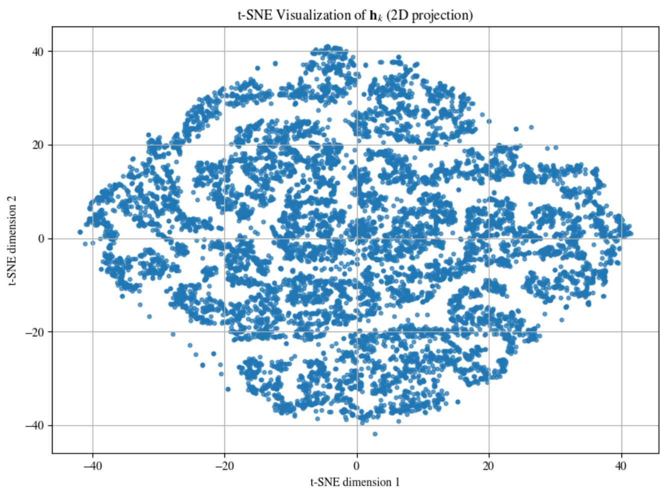


Figure 5.8: t-SNE representation of $\mathbf{h}_k$

To visualize the decision boundaries learned by the system, we project the discrete cluster labels discovered in the downstream spaces (latent $\mathbf{t}$, 5.3.1, and the precoding $\mathbf{v}$, 5.4.1) back on the t-SNE visualization of the original channel vector $\mathbf{h}$. 5.9 displays the $\mathbf{h}$-manifold colored by the 11 $\mathbf{t}$-labels (the UE encoder's prototypes), while 5.10 displays the same manifold colored by the 12 $\mathbf{v}$-labels (the BS's actions). In both plots, points sharing the same label are not randomly scattered across the map like noise; instead, they aggregate into coherent, contiguous neighborhoods. This confirms that the model has learned a rule of "semantic proximity": channels that are physically similar (neighboring on the continuous $\mathbf{h}$ manifold) are consistently mapped to the same semantic prototype, whether at the encoder output ($\mathbf{t}$) or the precoder output ($\mathbf{v}$). Crucially, this result does not imply that the input channel $\mathbf{h}$ itself is clustered. As established, the $\mathbf{h}$ space is a continuous, non-clusterizable structure. Rather, these plots demonstrate that the end-to-end system has learned to "tessellate" this continuous physical space. The visible "islands" of color are effectively the decision regions carved out by the UE encoder (f_θ) and propagated by the BS network (g_ϕ), partitioning the infinite continuum of physical channels into a finite set of actionable semantic categories.

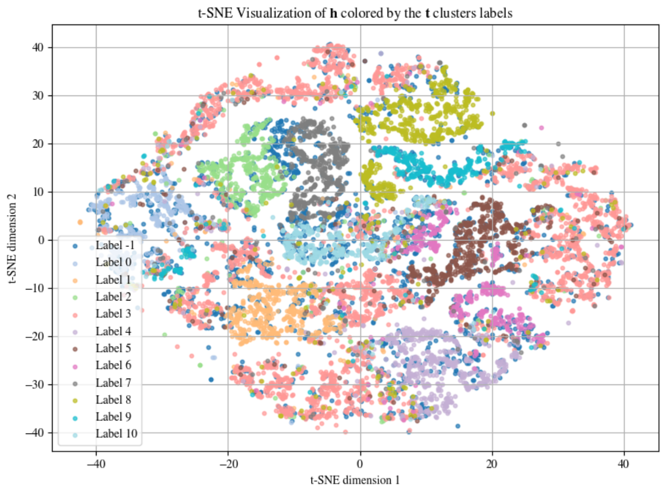


Figure 5.9: t-SNE Visualization of $\mathbf{h}$ colored by the $\mathbf{t}$ clusters labels

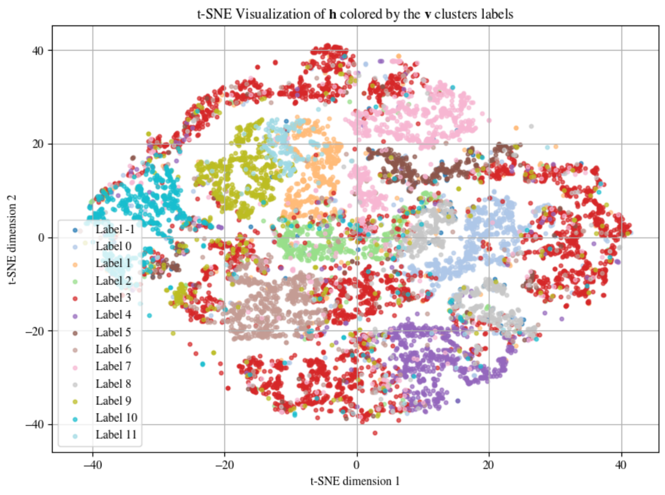


Figure 5.10: t-SNE Visualization of **h** colored by the **t** clusters labels

Chapter 6

Conclusion

This thesis set out to interpret the end-to-end, precoding-oriented CSI feedback framework of Carpi et al. [13], which, while high-performing, operates as a “black box”. Our goal was to move beyond performance metrics and apply a novel methodological framework from Explainable AI (XAI) to “open” this box and understand the internal strategy learned by the neural networks. We demonstrated that the network’s learned functions change based on the rate-overhead trade-off, as regulated by λ.
Our XAI methodology combined non-linear dimensionality reduction (for visualization) and quantitative clustering-based methods, to show the following:

1. We showed that, for low values of λ (e.g., $\lambda = 2$), the trade off is solved by actively developing a discrete, internal structure. Our clustering analysis found a stable set of 11 prototypes for the latent vector $\mathbf{t}$, and a “hierarchical" structure for the precoder $\mathbf{v}$. This hierarchy was revealed by the discovery of multiple stable plateaus in the DBSCAN sweep: a fine-grained “micro-set" of 17-18 prototypes, which, at a larger density scale, merged into a dominant “macro-set" of 12 prototypes.

2. We showed that this structure is a consequence of the compression constraint. By removing the constraint, setting λ to higher values (e.g., λ=100) this structured behavior vanished, leaving a continuous, non-clusterizable latent space. The BS learns a “hybrid" strategy: "deterministic" mappings for low-overhead scenarios, and "context-aware" mappings for high-overhead ones.

This work confirms that the network learns an interpretable strategy, but it opens several deeper questions about the nature of this strategy.
Future work could explore:

- Physical Nature of the High-Rate Latent Representation: A critical open question concerns the physical interpretability of the continuous latent vector

$\mathbf{t}$ learned in the high-rate regime ($\lambda = 100$). Does this vector estimate the explicit channel gains ($\mathbf{h}$), or does it extract the underlying physical parameters (α, β)? Our findings strongly suggest the former: the convergence of the pilots to an orthogonal basis (i.e. the classical optimality condition for minimizing channel estimation MSE) indicates that the system optimizes for a faithful reconstruction of the channel waveform rather than parameter extraction. Future work could investigate whether each latent component t_i encodes a complex, non-linear combination of all physical parameters (i.e., $t_i = f(\alpha_1, \alpha_2, \beta_1, \beta_2)$), or specific latent components focus exclusively on distinct physical factors.

- Analytical Approximation of the Learned Mapping ($\mathbf{t} \rightarrow \mathbf{v}$): While our contingency analysis revealed a mapping exists between the latents $\mathbf{t}$ and the prototypes $\mathbf{v}$, the actual function driving it remains a black box. A compelling direction for future work is to derive an analytical approximation of this learned function. Specifically, research should investigate whether the network effectively implements an adaptive approximation of the Regularized Zero-Forcing (RZF) strategy ($\mathbf{V} = \mathbf{H}^H(\mathbf{H}\mathbf{H}^H + \alpha\mathbf{I})^{-1}$), with $\mathbf{H} \approx f(\mathbf{t})$. The hypothesis is that the network could dynamically tune the regularization parameter (α) based on the interference level detected in the latents: it may learn to increase it, prioritizing power maximization (MRT-like), if the information extracted from the latent vectors reveals orthogonal channels, or to decrease it, prioritizing interference minimization (ZF-like).

- Physical Interpretation of the “Default" Precoding Vector Prototype: Our analysis revealed a highly skewed policy where a single precoding prototype accounts for over 30% of all decisions. A critical next step is to investigate the physical characteristics of this specific vector and what makes it so universally applicable. Does it correspond to a specific beamforming strategy?

- Explicit Control of the Prototype Space: Our analysis revealed that the network learns a set of precoding prototypes as a solution to the rate-overhead trade-off. A promising direction for future research is to transition from implicit discovery to explicit control. Future work could investigate methods to fix the exact number of precoding prototypes the network is allowed to learn. This could be achieved by introducing a new regularization term in the loss function that forces the continuous latent space to collapse into exactly that number of discrete states.

- Generalization across channel models: Our analysis was conducted on a specific sparse, geometric channel model, with $L_p = 2$. It could be interesting to see how the learned strategy adapts to different channel physics. If the model is retrained on a non-sparse, rich-scattering Rayleigh channel, would the learned codebook vanish? Or would it learn a different set of prototypes?

Bibliography

[1] Emil Björnson, Erik G. Larsson, and Thomas L. Marzetta. «Massive MIMO: Ten Myths and One Critical Question». In: *IEEE Communications Magazine* 54.2 (Feb. 2016), pp. 114–123. DOI: 10.1109/MCOM.2016.7402270. URL: https://doi.org/10.1109/MCOM.2016.7402270 (cit. on pp. 1, 2).

[2] Wei Chen, Weixiao Wan, Shiyue Wang, Peng Sun, Geoffrey Ye Li, and Bo Ai. «CSI-PPPNet: A One-Sided One-for-All Deep Learning Framework for Massive MIMO CSI Feedback». In: *IEEE Transactions on Wireless Communications* 23.7 (July 2024), pp. 7599–7611. DOI: 10.1109/TWC.2023.3342735. URL: https://doi.org/10.1109/TWC.2023.3342735 (cit. on p. 2).

[3] 3rd Generation Partnership Project (3GPP). *NR; Radio Resource Control (RRC); Protocol specification.* Technical Specification TS 38.331. Version 17.0.0. Release 17. ETSI, May 2022. URL: https://www.etsi.org/deliver/etsi_ts/138300_138399/138331/17.00.00_60/ts_138331v170000p.pdf (cit. on p. 2).

[4] 3rd Generation Partnership Project (3GPP). *NR; Physical layer procedures for data.* Technical Specification TS 38.214. Version 18.2.0. Release 18. ETSI, May 2024. URL: https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/18.02.00_60/ts_138214v180200p.pdf (cit. on p. 2).

[5] 3rd Generation Partnership Project (3GPP). *NR; Physical channels and modulation.* Technical Specification TS 38.211. Version 18.6.0. Release 18. ETSI, Apr. 2025. URL: https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/18.06.00_60/ts_138211v180600p.pdf (cit. on p. 2).

[6] Foad Sohrabi, Kareem M. Attiah, and Wei Yu. «Deep Learning for Distributed Channel Feedback and Multiuser Precoding in FDD Massive MIMO». In: *IEEE Transactions on Wireless Communications* 20.7 (2021), pp. 4044–4057. DOI: 10.1109/TWC.2021.3055202 (cit. on pp. 3–6, 9, 10, 12).

[7] X. Rao and V. K. N. Lau. «Distributed compressive CSIT estimation and feedback for FDD multi-user massive MIMO systems». In: *IEEE Transactions on Signal Processing* 62.12 (2014), pp. 3261–3271 (cit. on p. 3).

[8] 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». In: *IEEE Journal on Selected Areas in Communications* 26.8 (Oct. 2008), pp. 1341–1365. DOI: 10.1109/JSAC.2008.080828 (cit. on p. 3).

[9] S. S. Nair and S. Bhashyam. «Hybrid beamforming in MU-MIMO using partial interfering beam feedback». In: *IEEE Communications Letters* 24.7 (July 2020), pp. 1548–1552. DOI: 10.1109/LCOMM.2020.2999004 (cit. on p. 3).

[10] M. R. Castellanos, V. Raghavan, J. H. Ryu, O. H. Koymen, J. Li, D. J. Love, and B. Peleato. «Channel-reconstruction-based hybrid precoding for millimeter-wave multi-user MIMO systems». In: *IEEE Journal on Selected Topics in Signal Processing* 12.2 (May 2018), pp. 383–398. DOI: 10.1109/JSTSP.2017.2780983 (cit. on p. 3).

[11] A. Alkhateeb, G. Leus, and R. W. Heath. «Limited feedback hybrid precoding for multi-user millimeter wave systems». In: *IEEE Transactions on Wireless Communications* 14.11 (Nov. 2015), pp. 6481–6494. DOI: 10.1109/TWC.2015.2475695 (cit. on p. 3).

[12] G. Dietl and G. Bauch. «Linear precoding in the downlink of limited feedback multiuser MIMO systems». In: *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*. Washington, DC: IEEE, Dec. 2007, pp. 4359–4364 (cit. on p. 3).

[13] Fabrizio Carpi, Sivarama Venkatesan, Jinfeng Du, Harish Viswanathan, Siddharth Garg, and Elza Erkip. «Precoding-oriented Massive MIMO CSI Feedback Design». In: *Proceedings of the 2023 IEEE International Conference on Communications (ICC), SAC Machine Learning for Communications and Networking Track*. Feb. 2023. DOI: 10.1109/ICC45855.2023.10042240. URL: https://ieeexplore.ieee.org/document/10042240 (cit. on pp. 4–7, 9, 16, 21, 22, 81).

[14] C.-K. Wen, W.-T. Shih, and S. Jin. «Deep Learning for Massive MIMO CSI Feedback». In: *IEEE Wireless Communications Letters* 7.5 (2018), pp. 748–751. DOI: 10.1109/LWC.2018.2818160 (cit. on p. 4).

[15] J. Guo, C.-K. Wen, S. Jin, and G. Y. Li. «Overview of Deep Learning-Based CSI Feedback in Massive MIMO Systems». In: *IEEE Transactions on Communications* (2022). Early Access. DOI: 10.1109/TCOMM.2022.3144857 (cit. on p. 4).

[16] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. «Variational Image Compression with a Scale Hyperprior». In: *ICLR*. 2018 (cit. on pp. 5, 10, 12, 13).

[17] Mohammad B. Mashhadi, Qianqian Yang, and Deniz Gündüz. «Distributed Deep Convolutional Compression for Massive MIMO CSI Feedback». In: *IEEE Transactions on Wireless Communications* 20.4 (2021), pp. 2621–2633. DOI: 10.1109/TWC.2020.3040609 (cit. on pp. 5, 9, 12, 13).

[18] X. Chen and coauthors. «Implicit CSI Feedback for FDD Massive MIMO via Deep Learning». In: *IEEE Transactions on Communications* 70.xx (2022). (single-user beamforming with task-oriented training; update with exact metadata from your bib source), pp. xxxx–xxxx (cit. on p. 6).

[19] Natasha Devroye, Neshat Mohammadi, Abhijeet Mulgund, Harish Naik, Raj Shekhar, György Turán, Yeqi Wei, and Milos Zefran. «Interpreting Deep-Learned Error-Correcting Codes». In: *IEEE International Symposium on Information Theory, ISIT 2022, Espoo, Finland, June 26 - July 1, 2022*. IEEE, 2022, pp. 2457–2462. DOI: 10.1109/ISIT50566.2022.9834599. URL: https://doi.org/10.1109/ISIT50566.2022.9834599 (cit. on p. 8).

[20] N. Devroye, A. Mulgund, R. Shekhar, Gy. Turán, M. Žefran, and Y. Zhou. *Interpreting Training Aspects of Deep-Learned Error-Correcting Codes – extended ArXiv version*. June 2023. URL: https://arxiv.org/abs/2305.04347 (cit. on p. 8).

[21] N. Devroye, A. Mulgund, R. Shekhar, Gy. Turán, Y. Wei, and M. Žefran. «Evaluating interpretations of deep-learned error-correcting codes». In: *2022 60th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. Sept. 2022 (cit. on p. 8).

[22] A. Mulgund, N. Devroye, Gy. Turán, and M. Žefran. «Decomposing the Training of Deep Learned Turbo codes via a Feasible MAP Decoder». In: *International Symposium on Topics in Coding*. Sept. 2023 (cit. on p. 8).

[23] Y. Zhou, N. Devroye, Gy. Turán, and M. Žefran. «Interpreting Deepcode, a Learned Feedback Code». In: *2024 IEEE International Symposium on Information Theory (ISIT)*. 2024, pp. 1403–1408. DOI: 10.1109/ISIT57864.2024.10619390 (cit. on p. 8).

[24] Y. Zhou, N. Devroye, Gy. Turan, and M. Zefran. «Higher-order Interpretations of Deepcode, a Learned Feedback Code». In: *2024 60th Annual Allerton Conference on Communication, Control, and Computing*. 2024, pp. 1–8. DOI: 10.1109/Allerton63246.2024.10735282 (cit. on p. 8).

[25] R. Shekhar, N. Devroye, Gy. Turán, and M. Žefran. «Interpreting KO Codes». In: *International Symposium on Information Theory (ISIT*. Ann Arbor, USA, June 2025 (cit. on p. 8).

[26] Y. Zhou, N. Devroye, Gy. Turán, and M. Žefran. «On Non-Linearities of Simple Learned AWGN Feedback Codes». In: *International Symposium on Information Theory (ISIT)*. Ann Arbor, USA, June 2025 (cit. on p. 8).

[27] Jérémy Bégain, Fabien Racapé, Samuel Feltman, and Anshuman Pushparaja. *CompressAI: A PyTorch Library and Evaluation Platform for End-to-End Compression Research.* [Online; accessed 8-September-2025]. 2020. arXiv: `2011.03029 [cs.CV]`. URL: `https://arxiv.org/abs/2011.03029` (cit. on pp. 19, 20).

[28] Jarosław Duda. «Asymmetric Numeral Systems». In: *arXiv preprint arXiv:1311.2540* (2013) (cit. on p. 20).

[29] Laurens Van der Maaten and Geoffrey Hinton. «Visualizing data using t-SNE». In: *Journal of Machine Learning Research* 9.11 (2008), pp. 2579–2605 (cit. on pp. 28, 29).

[30] Solomon Kullback and Richard A Leibler. «On information and sufficiency». In: *The Annals of Mathematical Statistics.* Vol. 22. 1. 1951, pp. 79–86 (cit. on p. 29).

[31] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. «A density-based algorithm for discovering clusters in large spatial databases with noise». In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96).* AAAI Press. 1996, pp. 226–231 (cit. on pp. 57, 58).

[32] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 60).