

University of Essex

School of Computer Science and
Electronic Engineering

Deep Compressive Joint Source Channel Coding for Efficient End-to-End Communication Systems

Mohammadamin Jarrahi

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy

August 2026
Colchester

To the people who helped me fix my life while I was fixing my research:

To my wife, ***Pegah***, my home away from home and my permanent fixer. To my parents, ***Nadi*** and ***Maryam***, for the foundation of kindness and the belief that kept me moving. To my sister ***Afsaneh*** and my niece ***Tina***, for the laughter that brought me back to life.

This thesis is as much yours as it is mine.

Acknowledgments

First and foremost, I sincerely thank God, my source, guide, and inspiration. God brought me to Essex, led me to my supervisors, sustained me and my family, and carried us through the many challenges of this research journey. To God alone I owe my deepest gratitude.

I would like to express my deepest gratitude to my supervisor, Dr. Vahid Abolghasemi. He has been more than a supervisor to me. He has been like a big brother. His motivation, belief in me, and constant support made a real difference, especially during difficult moments. He always did everything he could to help me stay on track and move forward.

I am also sincerely grateful to my second supervisor, Dr. Eirina Bourtsoulatze for her guidance and support. She helped me navigate my research direction, solve many problems along the way, and generously spent a great deal of time reading my work carefully, fixing issues, and pushing the quality higher.

This PhD was not completed in easy years. I started in 2022, during a very difficult time for my country, Iran, with protests and tension. I have finished in 2026, again during a period of major protests. Carrying a PhD through that emotional pressure, while being far from home, was not simple, and I will always remember what it took.

To my wife, Dr. Pegah Ghazimoghadam, thank you for being beside me through everything. We grew together, and I could not have done this without your love, patience, and support. I also thank my parents, Nadi and Maryam, back in Iran for their unconditional love and encouragement despite the distance.

Last but not least, I want to thank me. I want to thank me for believing in me, for doing the hard work, for showing up on the days it was difficult, and for not quitting.

I try to carry one reminder with me, especially in research and in life: "Be curious, not judgmental". If people are curious, they ask questions, and questions change everything.

Abstract

This thesis integrates compressive sensing (CS) and deep joint source-channel coding (JSCC) to design bandwidth-efficient, robust wireless image transmission systems optimized for downstream inference. Conventional models employ fixed-dimensional representations that fail to adapt to varying channels, wasting resources on pixel reconstruction when only task-level metrics are required. To resolve these limitations, this work develops a unified family of compressive, adaptive, and task-oriented JSCC frameworks. The first part introduces fixed-rate compressive JSCC schemes embedding learned sampling and two-stage reconstruction directly into end-to-end pipelines. Utilizing block-based sampling, generalized divisive normalization (GDN), and content-adaptive measurements, these architectures significantly improve reconstruction fidelity and robustness over deep JSCC baselines across additive white Gaussian noise (AWGN) and Rayleigh fading channels. The second part extends this to rate-adaptive operation, where the compression ratio dynamically adapts to both image content and channel signal-to-noise ratio (SNR). Within a single model, this framework supports multiple effective rates and delivers superior reconstruction quality at lower average bandwidths than fixed-rate or purely SNR-adaptive alternatives. The final part shifts from reconstruction to semantics, introducing task-oriented JSCC frameworks for applications like detection and recognition that transmit compressed feature embeddings instead of full images. Under severe channel noise, these task-aware architectures maintain high inference accuracy while drastically reducing channel uses, establishing a robust foundation for adaptive, task-aware wireless image transmission.

Index Terms: Joint Source Channel Coding, Compressed Sensing, Semantic Communications, Adaptive JSCC, Task-Oriented JSCC.

Contents

Abstract	iii
List of Tables	vi
List of Figures	xiv
1 Introduction	1
1.1 Motivation and Aim	1
1.2 Existing Challenges	4
1.2.1 Challenges in Designing Deep JSCC Models	5
1.2.2 Challenges Arising from Compressive Components	6
1.2.3 Challenges in Task-Oriented JSCCs	7
1.3 Thesis Objectives	8
1.4 Thesis Contributions	9
1.5 Application in a Practical Communication Scenarios	11
1.6 Thesis Organization	13
2 Literature Review	15
2.1 JSCC for Wireless Image Transmission	15
2.2 Compressive Representations for Communication	20
2.2.1 Compressed Sensing	20
2.2.2 Compressive Sensing as a Tool for Communication Systems	25
2.3 Adaptive JSCC Frameworks	27
2.4 Semantic and Task-Oriented JSCC Frameworks	30

2.5	Evaluation Datasets and Performance Metrics	32
2.5.1	Evaluation Datasets	32
2.5.2	Performance and Communication Metrics	35
3	Deep Compressive JSCC Frameworks	39
3.1	Introduction and Related Works	39
3.2	System Model	42
3.3	BCS-JSCC: Block Compressive Sensing based JSCC for Wireless Image Tran- mission	43
3.3.1	Model Architecture	44
3.3.2	Loss Function	47
3.3.3	Results and Discussions	48
3.4	DCS-JSCC: Deep Compressive JSCC with Normalization and Refinement . . .	56
3.4.1	Model Architecture	56
3.4.2	Results and Discussions	61
3.5	CS-JSCC: Advanced Deep Compressive JSCC with Saliency Aware Sampling .	70
3.5.1	Model Architecture	71
3.5.2	Loss Functions and Training Process	78
3.5.3	Results and Discussions	80
3.6	Overview of the Proposed Fixed-Rate Compressive JSCC Frameworks	97
3.7	Conclusive Remarks	100
4	Rate Adaptive JSCC Frameworks	103
4.1	Introduction and Related Works	103
4.2	System Model	106
4.3	RACS-JSCC: Rate Adaptive Compressive Sampling for Image JSCC	108
4.3.1	Model Architecture	108
4.3.2	Loss Function	114
4.3.3	Results and Discussion	115
4.4	URC-JSCC: Unified Rate Controlled Deep Compressive JSCC	126

4.4.1	Model Architecture	126
4.4.2	Loss Function	134
4.4.3	Results and Discussion	135
4.5	Overview of the Proposed Rate-Adaptive Compressive JSCC Frameworks . . .	148
4.6	Conclusive Remarks	151
5	Task-Oriented JSCC Frameworks	154
5.1	Introduction and Related Works	154
5.2	TCS-JSCC: Task-Oriented JSCC with Adaptive Deep Compressed Sensing . .	156
5.2.1	System Model	157
5.2.2	Model Architecture	158
5.2.3	Loss Function	163
5.2.4	Results and Discussion	164
5.3	FRCS-JSCC: Task-Oriented JSCC for Face Recognition over Wireless Networks	182
5.3.1	System Model	182
5.3.2	Model Architecture	184
5.3.3	Loss Function	194
5.3.4	Training Strategy	197
5.3.5	Results and Discussion	198
5.4	Comparison of the Proposed Task-Oriented Compressive JSCC Frameworks . .	211
5.5	Conclusive Remarks	213
6	Conclusions and Future Directions	216
6.1	Conclusions	216
6.2	Future Work	218
	Bibliography	221

List of Tables

3.1	Ablation study isolating the performance impact of various configurations in DCS-JSCC (Kodak dataset, $R = 1/6$, $\text{SNR}_{\text{train}} = 4$ dB).	67
3.2	Layer configuration of CS-JSCC. Output sizes are given as Channels $\times$ Height $\times$ Width. B is the block size, K is the kernel size, S is the stride, and P is the padding. . .	81
3.3	Effect of global target sampling rate SR on reconstruction quality (Kodak dataset, $R = 1/6$, $\text{SNR}_{\text{train}}=2$ dB, $\text{SNR}_{\text{test}}=0$ dB).	91
3.4	Ablation study on the number of sampling channels (s) in the proposed CS-JSCC system. Results are averaged over the CIFAR-10 test set with global sampling rate $SR = 0.1$	93
3.5	Component-level ablation study on CIFAR-10 under AWGN channel with $SR = 0.1$, $R = 1/6$, and $\text{SNR}_{\text{test}} = 7$ dB.	94
3.6	Complexity and latency comparison of different JSCC-based image transmission methods. Latency values are reported per image and averaged over the test set. .	95
3.7	Qualitative comparison of BCS-JSCC, DCS-JSCC, and CS-JSCC.	99
4.1	Ablation study evaluating the impact of the sampling-rate selection module in RACS-JSCC on the CIFAR-10 test set over an AWGN channel.	122
4.2	Complexity and latency comparison of RACS-JSCC and baseline JSCC-based image transmission methods. Latency values are reported per image and averaged over the Kodak test set.	123
4.3	Ablation studies on core components of URC-JSCC for CIFAR-10 over an AWGN channel. PSNR values are reported at two representative SNRs. The average sampling ratio $\bar{\alpha}$ is reported only where it is meaningful.	143

4.4	Complexity and latency comparison of URC-JSCC and baseline JSCC-based image transmission methods. Latency values are reported per image and averaged over the Kodak test set.	145
4.5	Ablation study on loss function hyper-parameter λ in Eq. (4.18) for URC-JSCC on CIFAR-10 over an AWGN channel. Values are reported at representative SNR_{test} points.	146
4.6	Qualitative comparison of RACS-JSCC and URC-JSCC.	149
4.7	Comparison between developed frameworks of fixed-rate compressive JSCC (Chapter 3) vs. rate-adaptive compressive JSCC (Chapter 4).	152
5.1	Sensitivity to loss-weight hyperparameters. Each row changes the strength of one loss component while keeping the other components at their default values. For the feature-consistency component, the overall strength is adjusted by setting λ_{mse} and λ_{cos} to the same value (weak or strong), while λ_{rate} and λ_{IoU} remain fixed.	170
5.2	Component ablation study isolating SRNet and FiLM across representative SNR values.	171
5.3	Sensitivity to the depth of the residual refinement module. L denotes the number of 3×3 convolutional layers in the residual branch. Parameter counts are exact (weights only). The $L = 2$ row corresponds to the proposed configuration and matches the main results.	174
5.4	Block size sensitivity analysis for TCS-JSCC at input resolution 256×256 input. $N = (256/B)^2$ represents the number of blocks per image, while BW denotes the effective bandwidth compression ratio (as defined in the manuscript).	175
5.5	Implementation-level architecture of FRCS-JSCC.	193
5.6	Latency breakdown (ms) for FRCS-JSCC and baselines (batch size = 1). Neural-network components are timed on an NVIDIA A100 GPU, while JPEG and LDPC use standard CPU implementations; reported latencies are end-to-end on the same machine.	205

5.7	Ablation study on architectural component contribution at 0, 10 and 20 dBs. Entries indicate mean drops in LFW verification accuracy (percentage points) relative to full FRCS-JSCC, averaged over five runs, with standard deviation.	206
5.8	Contribution of individual loss components in the final FRCS-JSCC objective. Each row removes one auxiliary term while keeping the remaining losses and training settings unchanged.	207
5.9	Sensitivity of LFW accuracy (%) with respect to individual scaling of λ_{feat} , λ_{diff} , and λ_{rate} . The factor 1.0 corresponds to the default weights used in the main experiments.	208
5.10	Qualitative comparison of the proposed task-oriented JSCC frameworks in Chap- ter 5: TCS-JSCC (face detection) and FRCS-JSCC (face recognition).	212

List of Figures

1.1	Overview of the thesis road-map, linking key challenges in wireless image transmission to the proposed research directions and expected outcomes.	10
2.1	Components of the conventional separate source-channel coding processing pipeline for a typical point-to-point image transmission system	16
2.2	Components of classic JSCC frameworks for a typical point-to-point image transmission system	17
2.3	Components of the deep JSCC for a typical point-to-point image transmission system	18
2.4	Example of a grayscale image (left) and its transform-domain coefficients (right), showing that most coefficients are near zero while a small subset carries the dominant structure.	22
2.5	Schematic of the CS procedure.	23
3.1	Components of the system model for JSCC frameworks.	42
3.2	Architecture of the proposed BCS-JSCC framework	44
3.3	Performance of the proposed BCS-JSCC versus varying compression ratios over an AWGN channel on CIFAR-10 dataset	49
3.4	Performance of BCS-JSCC and baseline methods as PSNR versus channel SNRs over an AWGN channel on CIFAR-10 dataset (compression ratio=1/6)	51
3.5	LPIPS comparison of BCS-JSCC and baseline methods over the AWGN channel	51
3.6	Performance of BCS-JSCC and baseline methods as PSNR versus channel SNRs over an AWGN channel on Kodak dataset (compression ratio=1/6)	53

3.7	LPIPS comparison of BCS-JSCC and baseline methods under different test SNR values.	53
3.8	Normalized training and validation MSE loss convergence of BCS-JSCC. . . .	54
3.9	Example of reconstructed image produced by the proposed BCS-JSCC and the baseline methods with compression ratio of $1/6$, $\text{SNR}_{\text{train}}$ of 1 dB and SNR_{test} values of 2, 10 and 20 dB	55
3.10	Architecture of the proposed DCS-JSCC framework	57
3.11	Performance of DCS-JSCC and baseline methods as PSNR versus channel SNRs over an AWGN channel on CIFAR-10 dataset (compression ratio= $1/6$)	62
3.12	LPIPS comparison of DCS-JSCC and baseline methods over the AWGN channel on the CIFAR-10 dataset.	63
3.13	Performance of DCS-JSCC and baseline methods as PSNR versus channel SNRs over an AWGN channel on Kodak dataset (compression ratio= $1/6$)	65
3.14	LPIPS comparison of DCS-JSCC and baseline methods over the AWGN channel on the Kodak dataset. Lower LPIPS indicates better perceptual image quality. .	66
3.15	Normalized convergence of L_{int} , L_{deep} , and L_{total} in DCS-JSCC.	68
3.16	Example of reconstructed image produced by the proposed DCS-JSCC and the baseline methods with compression ratio of $1/6$, $\text{SNR}_{\text{train}}$ of 1 dB and SNR_{test} values of 2, 10 and 20 dB	69
3.17	Architecture of the proposed CS-JSCC framework	71
3.18	Proposed architecture for CS-JSCC encoder	72
3.19	Visualization of the saliency-based sampling rate assignment mechanism in CS-JSCC: (a) block partitioning, (b) saliency map, (c) normalized block weights, and (d) corresponding block-wise sampling rate assignment.	73
3.20	Proposed design for CS-JSCC decoder	76
3.21	Performance of the proposed CS-JSCC versus varying compression ratios over an AWGN channel with $R = 1/12$ in CIFAR-10 dataset	83
3.22	Performance of the proposed CS-JSCC versus varying compression ratios over an AWGN channel with $R = 1/6$ in CIFAR-10 dataset	84

3.23	Performance of the proposed CS-JSCC versus varying compression ratios over an Rayleigh channel with $R = 1/6$ in CIFAR-10 dataset	84
3.24	Performance of the proposed CS-JSCC versus varying compression ratios over an Rayleigh channel with $R = 1/3$ in CIFAR-10 dataset	85
3.25	LPIPS comparison of CS-JSCC and baseline methods over the AWGN channel on the CIFAR-10 dataset at $R = 1/12$	86
3.26	LPIPS comparison of CS-JSCC and baseline methods over the Rayleigh fading channel on the CIFAR-10 dataset at $R = 1/3$	87
3.27	Performance of the proposed CS-JSCC versus varying compression ratios over an AWGN channel with $R = 1/6$ in Kodak dataset	88
3.28	Performance of the proposed CS-JSCC versus varying compression ratios over an Rayleigh channel with $R = 1/6$ in Kodak dataset	89
3.29	LPIPS comparison of CS-JSCC and baseline methods over the Rayleigh fading channel on the Kodak dataset at $R = 1/6$	90
3.30	Comparison between hyper-parameter values of L_2 loss function (Kodak Dataset - $R = 1/6$ - $SNR_{\text{train}} = 4$ dB - $SNR_{\text{test}} = 2$ dB)	92
3.31	Normalized loss convergence of CS-JSCC: (a) Stage 1 MSE-based initial reconstruction loss L_1 , and (b) Stage 2 hybrid SSIM and ℓ^1 reconstruction loss L_2	96
3.32	Examples of reconstructed images produced by the proposed CS-JSCC and the baseline methods for bandwidth compression ratio $R = 1/6$ and SNR values of 2 and 10 dB	97
3.33	Envelope PSNR comparison of the suggested fixed-rate compressive JSCC frameworks on Kodak dataset over an AWGN channel at compression ratio $k/n = 1/6$	100
4.1	Components of the system model for rate-adaptive JSCC frameworks	106
4.2	Architecture of the proposed RACS-JSCC encoder.	109
4.3	Architecture of the sampling rate selection module.	110
4.4	Architecture of the proposed RACS-JSCC decoder.	113

4.5	Performance of the proposed RACS-JSCC versus varying SNR_{test} over an AWGN channel on CIFAR-10 dataset (compression ratio $k/n = 1/6$)	116
4.6	LPIPS comparison of RACS-JSCC and baseline methods over the AWGN channel on the CIFAR dataset.	117
4.7	Performance of the proposed RACS-JSCC versus varying SNR_{test} over an AWGN channel on Kodak dataset (compression ratio $k/n = 1/6$)	119
4.8	LPIPS comparison of RACS-JSCC and baseline methods over the AWGN channel on the Kodak dataset.	119
4.9	Example of block-wise sampling rate selection produced by RACS-JSCC encoder under a AWGN channel at $\text{SNR}_{\text{test}}=1\text{dB}$. The left image is the corrupted (noisy) input, and the right image visualizes the rate selection map from the RACS-JSCC sampling rate selection module. Each colour on rate map denotes one of the three sampling branches: high (red), medium (yellow), and low (green) rate.	120
4.10	Example of block-wise sampling rate selection produced by RACS-JSCC encoder under a Rayleigh fading channel at $\text{SNR}_{\text{test}}=1\text{dB}$. The left image is the corrupted (noisy) input, and the right image visualizes the rate selection map from the RACS-JSCC sampling rate selection module. Each colour on rate map denotes one of the three sampling branches: high (red), medium (yellow), and low (green) rate.	121
4.11	Reconstruction quality of an exemplary image (PSNR/SSIM) with RACS-JSCC and baseline methods under the compression ratio $k/n = 1/6$ and $\text{SNR}_{\text{train}}=4\text{dB}$	124
4.12	Architecture of the URC-JSCC encoder.	127
4.13	Architecture of a GSM-inspired linear downsampling module used as the URC-JSCC sampling network [170] (K_r is 3 in this figure).	128
4.14	Architecture of the proposed decoder.	132
4.15	PSNR (y-axis), and bandwidth compression (boxed numbers) against varying SNR_{test} values [0-20 dB] for CIFAR-10 dataset over an AWGN channel. Deep JSCC and ADJSCC operate at a fixed bandwidth compression ratio of $1/6$. . .	138

4.16	LPIPS comparison of URC-JSCC and baseline methods over the AWGN channel on the CIFAR-10 dataset.	139
4.17	PSNR (y-axis), and bandwidth compression (boxed numbers) against varying SNR_{test} values [0-20 dB] for Kodak dataset over a Rayleigh channel. Deep JSCC and ADJSCC operate at a fixed bandwidth compression ratio of $1/6$	140
4.18	LPIPS comparison of URC-JSCC and baseline methods over the AWGN channel on the Kodak dataset.	141
4.19	Rate-distortion comparison of URC-JSCC and baseline methods as PSNR versus compression ratio (k) for the CIFAR-10 dataset over an AWGN channel at $\text{SNR}_{\text{test}} = 3$ dB.	142
4.20	Reconstruction quality of an exemplary image (PSNR/SSIM) for URC-JSCC versus baseline methods at different SNR values. Deep JSCC and ADJSCC use the compression ratio $k/n = 1/6$	147
4.21	PSNR comparison of URC-JSCC and RACS-JSCC on the Kodak dataset over an AWGN channel as a function of SNR_{test} ; annotated boxes indicate the bandwidth ratio (k/n) at selected operating points.	150
5.1	System model of proposed TCS-JSCC.	157
5.2	Block diagram of the proposed TCS-JSCC framework and architecture design of the encoder and decoder networks.	159
5.3	Block diagram of the proposed SRNet network.	160
5.4	F1-score (y-axis), and bandwidth compression (boxed numbers) against varying SNR values [0-20 dB] for TCS-JSCC vs. benchmark methods.	167
5.5	Performance metrics of TCS-JSCC for face detection task across SNR levels: (a) recision and (b) recall.	168
5.6	Empirical distribution of SRNet rate decisions across blocks at representative SNR values. Each bar reports the percentage of blocks sampled at each rate in $\mathcal{R} = \{0.2, 0.4, 0.8\}$	169
5.7	Illustrative SRNet block-wise sampling rate selection at $\text{SNR} = 1$ dB. Left: clean image. Right: per-block sampling rate map selected from $\mathcal{R} = \{0.2, 0.4, 0.8\}$	169

5.8	Visual comparison of detection outputs for SSCC, Deep JSCC+MTCNN, and TCS-JSCC at SNR levels of 20 dB, 10 dB, and 1 dB. Yellow boxes indicate ground truth, while red boxes show the detection results produced by each method. The overplotted numbers represent the detection confidence score. . . .	176
5.9	F1-score versus SNR for the evaluated methods under the same detection protocol.	177
5.10	License plate detection results at different SNR values. Yellow boxes indicate detections on clean input, red boxes indicate detections under noisy channel conditions.	178
5.11	Block-wise sampling rate selection for license plate detection at low SNR. Red denotes high sampling density, orange medium, and green low.	179
5.12	Top-1 accuracy versus SNR for the evaluated methods.	181
5.13	System model of the proposed FRCS-JSCC framework	183
5.14	Detailed architecture of proposed FRCS-JSCC including encoder and decoder networks	185
5.15	Detailed architecture of MobileFaceNet [197].	186
5.16	Rank-1 identification accuracy on MegaFace versus SNR over an AWGN channel	201
5.17	Rank-1 identification accuracy on MegaFace versus SNR over a Rayleigh fading channel	202
5.18	LFW verification accuracy versus CuPP at SNR = 10 dB for FRCS-JSCC, Deep-JSCC+ArcFace and JPEG+LDPC+ArcFace.	203
5.19	Convergence behaviour of the FRCS-JSCC training strategy compared with fully joint training from the beginning	209
5.20	Qualitative verification robustness of the methods. For each method, corrupted images at 20, 10, and 2 dB SNRs are matched against a clean anchor. The first row shows same-identity verification, and the second row shows different-identity rejection.	210

Introduction

1.1 Motivation and Aim

Wireless image transmission has become a core requirement in modern intelligent systems, including augmented and virtual reality, autonomous vehicles, remote sensing, and large-scale surveillance [1]. These applications frequently operate under strict latency, power, and bandwidth constraints, and must maintain reliable performance in wireless environments that experience rapid channel variations, fading, and severe noise. Transmitting visual information in such conditions is challenging because the communication system must balance three competing objectives: efficient compression of high-resolution image and video data, robustness against unpredictable channel impairments, and low end-to-end delay [2].

Conventional wireless image transmission pipelines are built upon the principle of separate source and channel coding. In this paradigm, an image is first compressed using methods such as JPEG or JPEG2000 and is then protected by forward error correction codes such as low-density parity-check (LDPC), Turbo or polar codes. Although this modular workflow is optimal in the asymptotic regime of infinitely long block code lengths, it exhibits significant performance degradation in practical scenarios where latency is constrained, and the channel Signal-to-Noise Ratio (SNR) changes unpredictably. Even small mismatches between the assumed and actual channel conditions can lead to abrupt drops in reconstruction quality, while conservative channel

codes result in inefficient use of bandwidth [3]. These limitations motivate the development of communication strategies that operate reliably under realistic channel dynamics.

Joint source channel coding (JSCC) provides a communication strategy that addresses these issues in which source representation and channel transmission are designed jointly to optimize end-to-end performance under bandwidth, latency, and channel constraints, rather than being treated as strictly independent stages [4]. Depending on the design paradigm, JSCC may take the form of analytically derived mappings, hybrid analog-digital schemes, or data-driven end-to-end models. Classical JSCC algorithms attempt to exploit the statistical properties of the source together with the channel characteristics to improve end-to-end distortion [5]. However, these methods rely on simplified analytical models and cannot efficiently handle the complexity or high dimensionality of natural images. The introduction of deep learning has significantly advanced the capabilities of JSCC. Deep JSCC architectures replace hand-crafted transforms with neural encoders and decoders that learn to adapt the transmitted representation to the channel conditions [6]. This class of methods has demonstrated resilience to SNR variations, improved performance at very low bandwidth compression ratios, and more stable performance degradation profiles compared with separation-based approaches [7], [8]. Despite these advantages, many existing deep JSCC models operate at fixed compression ratios and are trained for fixed SNR values. As a result, they require multiple networks or frequent retraining to support different bandwidth budgets and channel qualities, which limits their practicality in dynamic wireless environments [9].

Compressed sensing (CS) offers a sampling-based approach to dimensionality reduction that is well aligned with wireless image transmission, where limited bandwidth and energy motivate reducing the amount of data before transmission [10]. By exploiting the approximate sparsity or compressibility of images in suitable transform domains, CS acquires a compact set of linear measurements that can be conveyed over the channel with substantially fewer samples than uniform Nyquist sampling would require [11]. A practical advantage in JSCC context is that the sensing operation is a simple linear projection, which can be implemented with low computational complexity at the transmitter and can therefore act as an early compression mechanism within the communication pipeline. This benefit, however, typically comes with a receiver-side cost in

classical CS, since accurate recovery often depends on iterative reconstruction algorithms and on sensing-matrix conditions that may be difficult to satisfy for diverse natural images and noisy wireless channels [12]. Recent deep learning based CS frameworks mitigate these limitations by learning the sensing and reconstruction stages jointly, replacing iterative solvers with fast feed-forward networks and enabling the measurement process to adapt to image statistics, thereby improving practical reconstruction quality at low sampling ratios [13].

A common way to reduce transmission load in bandwidth-limited visual links is to acquire and transmit a reduced set of linear measurements rather than full-resolution samples, which aligns naturally with the CS paradigm. In classical wireless image transmission, this idea appears most prominently in analog and hybrid analog-digital schemes, where CS-style projections are computed at the transmitter and then mapped directly to channel symbols via analog modulation, as in SoftCast and SparseCast [14], [15]. Such approaches can reduce the number of transmitted coefficients and provide graceful quality degradation compared with fragile entropy-coded bitstreams; however, the sensing operation is typically designed independently of the channel and receiver, which limits the ability to adapt measurements to time-varying channel conditions or to image-dependent structure, and can shift substantial computational burden to the receiver through iterative reconstruction under noise. Motivated by these limitations, more recent work has explored integrating CS modules into source and channel coding (both joint and separate) architectures, where CS-like sampling can be implemented within the encoder as a dimensionality-reduction stage that precedes channel mapping and subsequent refinement [16], [17]. Early attempts, including block-based CS-JSCC pipelines and related hybrid designs, report improvements in reconstruction quality under limited bandwidth and low SNR conditions [18], [19], [20], [21]. Nevertheless, many of these architectures still rely on fixed sampling ratios or only loosely coupled sensing and communication components, which restricts fine-grained adaptation to channel variability and to local image complexity.

These considerations motivate the central focus of this thesis. The objective is to develop deep learning based JSCC frameworks in which CS operates as an internal and trainable component of the encoder, enabling efficient dimensionality reduction while preserving information that is critical for end-to-end communication performance. The thesis examines several interconnected

research directions within this unified JSCC viewpoint. The first direction is the design of deep compressive JSCC architectures where the sensing matrices and neural encoder are trained jointly to produce compact source representations that remain robust under bandwidth and SNR constraints. The second direction is the development of rate-adaptive JSCC mechanisms in which the compression ratio is selected based on input characteristics and channel conditions, allowing the system to adjust its transmission strategy dynamically. The third direction is task-oriented JSCCs, where the communication framework is optimized to preserve task-specific features rather than full pixel-level fidelity. In this research direction, the objective shifts from pixel-accurate reconstruction to semantic fidelity, and the transmitter allocates channel resources to preserve task-relevant information for downstream inference such as detection or recognition, potentially operating directly on feature or embedding representations when full image recovery is unnecessary.

By addressing these research challenges, the thesis aims to develop a unified and practical family of JSCC frameworks that support efficient dimensionality reduction, strong robustness across a wide range of SNR values, and high performance on both reconstruction-based and task-based objectives. These systems are designed to operate effectively under the constraints and variability of real-world wireless communication environments.

1.2 Existing Challenges

Wireless image transmission systems face several persistent challenges that are not fully resolved by either classical separation-based pipelines or recent deep learning based JSCC models. These challenges arise due to the multi-dimensional nature of the problem [8]: high resolution images must be compressed efficiently, the transmitted representation must remain robust under severe channel impairments, and the system must adapt to significant variations in bandwidth and SNR. When the goal is not only reconstruction but also task-level inference, additional complexities arise related to the preservation of semantic information and the geometry of feature manifolds [22]. The integration of compressive sampling inside a JSCC encoder provides new opportunities but also introduces new constraints that must be addressed. This section outlines these challenges in detail and forms the basis for the contributions developed throughout this thesis.

1.2.1 Challenges in Designing Deep JSCC Models

Deep JSCC has shown strong potential for wireless image transmission, yet several limitations restrict its adoption in real-world systems. These limitations are summarized as follows.

- **Adaptation to time-varying channel conditions:** Wireless channels often experience rapid fluctuations in SNR due to fading, mobility, interference, and environmental changes. Most deep JSCC models are trained at one or a few SNR points, which causes a mismatch between training and deployment conditions [23]. When the channel quality deviates from the training regime, the learned mapping from images to channel symbols becomes suboptimal, and reconstruction quality deteriorates. Retraining or maintaining multiple SNR-specific models is impractical in real systems, especially when the transmitter or receiver operates on resource-limited hardware [24].
- **Inflexible compression ratio and fixed dimensionality of channel symbols:** Classical deep JSCC encoders output a channel vector of fixed length. This design implicitly assumes that the available bandwidth is constant. In practice, uplink and downlink bandwidth may vary rapidly due to congestion, scheduling, or system-level constraints. A JSCC model that cannot adjust its compression ratio, wastes channel resources during favorable conditions and suffers severe quality degradation when bandwidth is scarce. Designing a single model that supports several compression ratios without instability is challenging, since the encoder, decoder, and reconstruction pathways must all remain consistent across multiple rates [25].
- **Inefficiency of pixel domain reconstruction for task-driven applications:** Many applications, such as surveillance, industrial inspection, facial analytics, and autonomous navigation, require only a subset of the information present in a high-dimensional image. Traditional JSCC systems reconstruct the entire image, which forces the encoder to dedicate channel resources to textures, backgrounds, and low-frequency details that may be irrelevant for downstream tasks. This inefficiency becomes problematic when bandwidth is limited or when the channel introduces severe distortions. Moreover, pixel-level losses such as Mean Square Error (MSE) or Structural Similarity Index Measure (SSIM) do not reflect task

relevance and may encourage the transmission of details that do not contribute to the final decision [26].

- Difficulty preserving semantic structure in noisy conditions: When channel noise is high, the decoder must reconstruct a representation that remains semantically meaningful. For example, a small perturbation in the pixel domain may translate into a large shift in the feature embedding used by a recognition or detection model, causing the system to misclassify or fail to detect important objects. Ensuring that semantic or identity information remains intact under strong noise is substantially more difficult than preserving pixel fidelity [27].

1.2.2 Challenges Arising from Compressive Components

Incorporating CS as part of the JSCC encoder provides a principled mechanism for dimensionality reduction. However, doing so introduces several additional technical challenges that must be addressed for the system to be reliable.

- Learning sampling operators that remain stable across content variations: Natural images exhibit significant diversity in terms of spatial structure, texture, illumination, and sparsity patterns. A fixed hand crafted measurement matrix is unable to handle this variability effectively. Learned sensing matrices improve sampling efficiency but must be trained jointly with the JSCC encoder and decoder. Because the sampling ratio affects the dimensionality of the channel vector, any instability in the sampling operator propagates directly into the channel mapping and subsequent reconstruction [18].
- Sensitivity of compressive measurements to channel impairments: In CS based systems, the measurement vector is a linear and highly compressed representation of the original signal. Even small distortions introduced by Additive White Gaussian Noise (AWGN) or Rayleigh fading can lead to large reconstruction errors. When CS is part of a JSCC encoder, the decoder must simultaneously compensate for compression loss and channel noise. Balancing these forms of distortion is difficult, especially under low SNR conditions where the decoder may receive severely corrupted measurements [28].

- Supporting multiple sampling ratios in a single trainable model: Rate adaptive systems require several sampling matrices corresponding to different compression levels. Training all rates jointly complicates gradient flow and destabilizes convergence. Sampling matrices with different output dimensions must remain compatible with a unified decoding architecture, which requires careful design of inverse projections, normalization, and feature processing layers [29].

1.2.3 Challenges in Task-Oriented JSCCs

In semantic communication scenarios, the goal is not to reconstruct the original image but to preserve task-specific information. This introduces further challenges:

- Maintaining the geometry of task embeddings under compression and noise: For models such as MobileFaceNet or ArcFace, identity information lies on a structured manifold in a high-dimensional space [30], [31]. Compression and channel noise may push the embedding outside this manifold, causing large drops in recognition performance. Ensuring that transmitted features remain close to the clean embedding distribution is challenging, particularly when the system operates at low bitrates or low SNR levels.
- Designing decoders that adapt to both the task and the channel: For detection or recognition tasks, the decoder must perform two roles: recovering a semantically meaningful representation and correcting distortions introduced by the channel [32]. SNR conditioned Feature-wise Linear Modulation (FiLM) layers or diffusion-based refiners provide mechanisms for adaptation, but integrating these methods into an end-to-end trainable JSCC system increases architectural complexity.
- Balancing compression efficiency with task accuracy: Aggressive rate reduction may remove task-relevant features, while conservative compression wastes bandwidth. Learning a rate selection policy that adapts to both the input complexity and the channel conditions requires careful design of differentiable control modules and loss functions [33].

The combined challenges in deep JSCC, compressive sampling, rate adaptation, and task-oriented transmission indicate that existing solutions are insufficient for emerging wireless image

transmission systems. There is a need for unified architectures that:

- Integrate CS as an internal, trainable, and flexible component of the JSCC framework.
- Support multiple compression ratios and adapt to the channel SNR.
- Preserve task-specific features despite strong channel impairments.
- Enable feature domain transmission for recognition when pixel reconstruction is unnecessary.

These challenges motivate the development of the deep compressive, rate-adaptive, and task-oriented JSCC frameworks presented in the following chapters.

1.3 Thesis Objectives

Motivated by the challenges identified in the previous section, the primary objective of this thesis is to develop unified deep learning-based JSCC frameworks in which CS operates as an internal, trainable component of the whole framework. The overarching aim is to enable efficient dimensionality reduction, strong robustness across a wide range of SNR values, and high performance on both reconstruction-based and task-based objectives with CS-based JSCC frameworks. To achieve this, the research is guided by three specific objectives:

- **To design deep compressive JSCC architectures for robust image reconstruction:** The first objective focuses on integrating learned CS-based operators directly into convolutional JSCC pipelines. The goal is to jointly train the sensing matrices and deep-based encoder and decoders to produce compact source representations that maintain high reconstruction fidelity under fixed bandwidth constraints and varying channel impairments, such as AWGN and Rayleigh fading.
- **To develop adaptive JSCC mechanisms:** The second objective is to overcome the limitations of fixed-rate and fixed-SNR models by creating a system that can adjust its transmission strategy dynamically. This involves designing frameworks that select compression ratios on a per-block basis according to local image content and channel

SNR. A key aim here is to support multiple sampling rates within a single trained model using novel architectures, thereby optimizing bandwidth usage without requiring network retraining.

- **To establish task-oriented JSCC frameworks for semantic communication:** The final objective shifts the focus from pixel-accurate reconstruction to semantic fidelity. The aim is to optimize the communication framework specifically for downstream inference applications such as detection and classification. This involves designing systems that allocate channel resources to preserve only task-relevant features or identity embeddings, enabling efficient and accurate edge intelligence even under severe wireless noise.

1.4 Thesis Contributions

This thesis studies deep learning based JSCC for wireless image transmission, with a particular focus on compressive architecture, rate adaptivity, and task-oriented operation, as shown in Fig. 1.1. Across several architectures and application scenarios, the work develops JSCC systems in which CS modules are integrated as trainable components of the framework rather than as independent subsystems. The proposed methods address the challenges identified in the previous section by enabling dimensionality reduction, dynamic rate selection, and semantic preservation under realistic wireless channel conditions. The main contributions of the thesis are summarized as follows.

- **Deep compressive JSCC for wireless image transmission (Chapter 3):** The first contribution is a family of deep compressive JSCC architectures for wireless image transmission. In these designs, a learned sensing operator is inserted at the encoder architecture and trained jointly with a convolutional JSCC framework. This operator compresses the input to a lower-dimensional representation that is directly mapped to channel symbols, without passing through a separate source coder. The designed encoders and decoders are trained end-to-end over AWGN and fading channels, with the compression module optimized to preserve visually and structurally important components of the image under a symbol budget. The resulting framework consistently improves peak SNR (PSNR) and SSIM

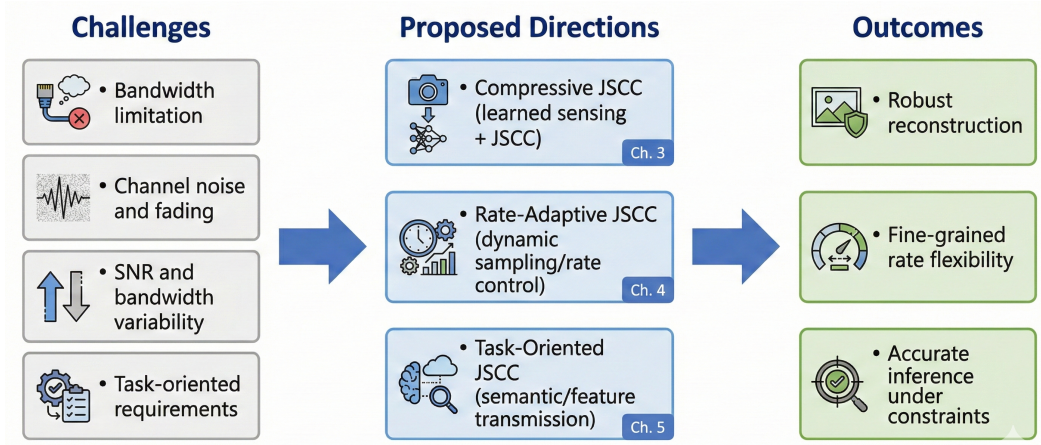


Figure 1.1: Overview of the thesis road-map, linking key challenges in wireless image transmission to the proposed research directions and expected outcomes.

compared with conventional DeepJSCC and separation-based baselines across a range of bandwidth compression ratios and SNR values.

- Deep rate-adaptive compressive JSCC architectures (Chapter 4):** The second contribution is a deep rate-adaptive JSCC framework that operates on image blocks and assigns each block a sampling rate according to its content and the channel SNR. Unlike fixed-rate JSCC models, the encoder includes a rate selection module that chooses among discrete sampling rates on a per block basis. Each block is then passed through a learned sampling network that preserves local structure at the chosen rate, and the resulting measurements are processed by a hybrid CNN-Transformer JSCC framework. The convolutional layers capture local textures and stabilize statistics, while the Transformer layers model long-range dependencies and enforce global consistency even when different blocks are sampled at different rates. Rate flags are transmitted as protected side information to guide block-specific decoding. This design extends earlier architectures by introducing rate-adaptive sampling operators and a CNN-Transformer structure which demonstrates consistent gains in PSNR and SSIM over state-of-the-art rate-adaptive-and fixed-rate JSCC baselines on CIFAR-10 and ImageNet datasets.
- Task-oriented compressive JSCC architectures (Chapter 5):** This thesis develops a unified task-oriented JSCC framework that transmits only the information required for downstream inference tasks. The first part focuses on task-oriented compressive JSCC,

where the encoder uses an adaptive deep compressive module and selects the sampling rate based on the input and the channel SNR. Rather than reconstructing pixels, the receiver decodes the transmitted measurements into task-relevant feature maps that are fed to a task head. The system is trained jointly so that channel resources are allocated to regions and features that are most important for the target task. The proposed framework was evaluated on multiple applications, including face detection and image classification, demonstrating that the approach is not limited to a specific task. The second part extends this idea to feature-domain transmission using face recognition as a representative application. The encoder first extracts identity embeddings and transmits a rate-adaptive compressive projection of these embeddings, avoiding pixel-level reconstruction altogether. The decoder performs SNR-conditioned reconstruction and applies a lightweight diffusion-based refiner before classification with a fine-tuned ArcFace head. Face recognition was selected as a challenging semantic communication use case because it relies on preserving highly discriminative embedding representations rather than visual reconstruction. These designs show that JSCC can be optimized for semantic accuracy rather than pixel fidelity and that transmitting compressed features or embeddings leads to better accuracy versus channel use trade-offs and higher robustness under AWGN and Rayleigh fading than related baselines.

Overall, the thesis advances the state-of-the-art in deep JSCC by showing how compressive sampling, rate adaptation, and task-oriented design can be combined in a principled way within a single end-to-end learning framework, and by demonstrating the benefits of such designs for both generic image transmission and face-centric visual intelligence over wireless channels.

1.5 Application in a Practical Communication Scenarios

The proposed frameworks can be applied in a practical wireless image transmission or edge intelligence system where a camera-equipped device communicates with an edge server, base station or remote receiver over a bandwidth-limited and noisy wireless link. Typical examples include surveillance cameras, autonomous platforms, remote sensing devices, mobile robots

and edge-based visual analytics systems. In such scenarios, the transmitter has limited power and bandwidth, while the receiver is expected either to reconstruct the image or to perform a downstream task such as detection, classification or recognition.

A practical deployment of the proposed reconstruction-oriented frameworks in Chapters 3 and 4 would involve the following steps. First, the camera captures an image and normalizes it at the transmitter. Second, the encoder applies a learned compressive sensing module to reduce the source dimension before channel mapping. In the fixed-rate case, a predefined sampling ratio is used, while in the rate-adaptive case the system estimates the channel SNR and selects suitable sampling rates for image blocks according to both image content and channel condition. Third, the selected rate information is transmitted as a short protected control header when required. Fourth, the compressed representation is mapped to channel symbols, normalized to satisfy the transmit power constraint and sent over the wireless channel. At the receiver, channel equalization is applied if channel state information is available, the JSCC decoder recovers the received representation, and the reconstruction network produces the final image. The reconstructed image can then be used by a human observer or by a downstream machine vision system.

For the task-oriented frameworks in Chapter 5, the deployment procedure is similar but the receiver does not need to reconstruct the full image. Instead, the transmitter extracts task-relevant features or embeddings, compresses them using the learned rate-adaptive projection, applies SNR-aware modulation and sends only the semantic representation through the channel. At the receiver, the decoded features are passed directly to the task head, such as a detection, classification or recognition module. This reduces the number of transmitted symbols because irrelevant background or pixel-level details do not need to be preserved when they are not required for the final task.

In a real system, the models would be trained offline using representative source data and channel simulations, and then deployed at the transmitter and receiver. During operation, the receiver can estimate the channel SNR using pilot symbols and feed this estimate back to the transmitter through a low-rate control link. The transmitter then uses this SNR estimate for rate selection and SNR-conditioned encoding. When the channel or data distribution changes

significantly, the model can be fine-tuned or updated periodically. Therefore, the proposed contributions can be integrated into a practical communication pipeline by combining offline training, SNR estimation and feedback, adaptive compressive encoding, protected side information transmission, wireless channel transmission, receiver-side decoding and either image reconstruction or task-level inference.

1.6 Thesis Organization

The remainder of this thesis is organized into the following five chapters:

- Chapter 2 provides a literature review and background on JSCC, CS, and deep learning based task-oriented and semantic communication. It reviews classical and modern JSCC methods, discusses the CS concept, and then summarizes recent developments in rate adaptive and task-oriented communication methods.
- Chapter 3 introduces deep compressive JSCC for wireless image transmission. The chapter presents architectures in which a learned compressive sampling module is integrated into a convolutional JSCC encoder, and the entire system is trained end-to-end over AWGN and fading channels. The chapter reports the performance improvements obtained in terms of PSNR and SSIM and discusses the advantages of incorporating compressive sampling inside the JSCC architecture.
- Chapter 4 investigates rate adaptive JSCC and develops block-based and hybrid CNN-Transformer architectures that support several sampling rates within a single model. A rate selection module assigns sampling rates to image blocks based on content statistics and channel SNR, and the decoder reconstructs the image using rate-specific information. The chapter evaluates the performance of this approach under varying bandwidth conditions and demonstrates its advantages over fixed-rate schemes.
- Chapter 5 presents the unified task-oriented JSCC framework for face-related tasks. The chapter first describes an adaptive JSCC system for face detection, where the encoder selects the sampling rate based on the input and the channel state, and the decoder

produces task-relevant features for detection. The chapter then extends the same idea to face recognition, where identity embeddings are transmitted directly through a rate-adaptive projection and reconstructed with an SNR conditioned decoder followed by a diffusion-based refiner. The chapter compares both designs with related baselines and analyzes their performance under AWGN and Rayleigh fading.

- Chapter 6 concludes the thesis by summarizing the main contributions, discussing limitations and practical considerations, and outlining potential directions for future work in deep compressive and task-oriented JSCC frameworks.

Literature Review

The rapid growth of wireless image transmission systems has stimulated extensive research into methods that can deliver images and features reliably under bandwidth, latency, and SNR constraints [8], [9]. Recent advances in deep learning have reshaped this field by enabling end-to-end trainable systems that jointly optimize compression, robustness to channel impairments, and reconstruction quality. At the same time, CS has emerged as an important framework for compact signal representation, and its integration with deep learning has produced flexible data-driven measurement models with strong practical performance [13], [34]. Beyond reconstruction, recent research has increasingly focused on adaptation to varying channel conditions and on communication strategies aligned with downstream inference tasks rather than pixel-level fidelity alone [35], [36]. This chapter reviews the literature at the intersection of JSCC, deep learning-based image transmission, and CS. It first examines classical and deep JSCC models, then surveys CS and its modern adaptations, and finally discusses rate-adaptive and task-oriented JSCC architectures.

2.1 JSCC for Wireless Image Transmission

JSCC has long been recognized as a fundamental alternative to the classical principle of source-channel separation. In conventional wireless image transmission systems including separate source-channel coding frameworks, shown in Fig. 2.1, an image is first compressed using a

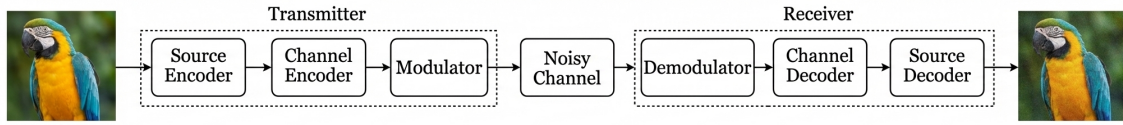


Figure 2.1: Components of the conventional separate source-channel coding processing pipeline for a typical point-to-point image transmission system

source coder such as JPEG, JPEG2000, or BPG and the resulting bitstream is then protected by a channel code such as LDPC or turbo codes before modulation. This pipeline is supported by the separation theorem, which ensures optimality when blocklengths are arbitrarily large and when the encoder has perfect channel knowledge [37]. In realistic wireless environments, these assumptions rarely hold. Mobile and heterogeneous networks often operate under strict delay constraints and short blocklengths, and channel conditions may vary unpredictably due to fading, interference, and mobility [38]. Under such constraints, separation-based designs frequently exhibit the cliff effect, where small deviations from the assumed SNR lead to abrupt degradation in reconstruction quality. Bitstream errors produced by compressed images propagate catastrophically through entropy decoders, resulting in block artefacts, desynchronization, or complete decoding failure. These drawbacks limit the practical robustness of separation-based visual communication pipelines [6].

Classic JSCC frameworks, as shown in Fig. 2.2, were originally proposed to overcome these limitations by jointly designing compression and error protection. Early methods include pure analog mappings and hybrid digital-analog transmission schemes such as SoftCast and related linear transforms [14], [39], [40]. These approaches bypass entropy coding and map pixel intensities directly to channel coefficients through linear transforms. As a result, they provide smooth performance degradation as a function of SNR and avoid the brittle behavior observed in fully digital schemes. Hybrid analog-digital designs introduce additional redundancy to improve robustness while still maintaining analog like graceful degradation. Although these methods improve resilience at moderate bandwidths and SNRs, their reliance on linear operators restricts their representational capability. Linear transforms cannot fully exploit the complex structure of natural images and therefore cannot match the reconstruction accuracy obtainable with modern nonlinear models. Surveys on JSCC fundamentals [8], [41], [42] highlight that analog and hybrid methods remain useful reference points but are not competitive for high fidelity visual

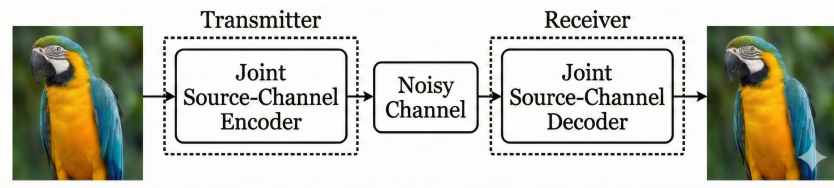


Figure 2.2: Components of classic JSCC frameworks for a typical point-to-point image transmission system

reconstruction.

The rapid progress of deep learning has transformed the development of JSCC by enabling data-driven nonlinear mappings from images to channel symbols [6], [7], [43], [44], [45], [46], [47], [48]. Deep JSCC employs neural networks as both the encoder and decoder, trained jointly over differentiable channel models, as shown in Fig. 2.3. The seminal work Deep JSCC [6] demonstrated that convolutional encoder-decoder architectures trained directly through an AWGN or Rayleigh fading model can outperform traditional compression schemes paired with powerful channel codes. Deep JSCC offers two practical advantages compared with separation-based designs. First, it delivers graceful degradation with respect to SNR because it operates in the real domain and does not rely on fragile bitstreams. Second, it maintains useful reconstruction quality even when channel uses are severely constrained, often surpassing JPEG2000 and LDPC-based baselines at low bandwidths. Building on these results, numerous extensions and architectural variations have been proposed. Residual networks and dense skip connections improve feature propagation within JSCC encoders and decoders and are known to stabilize training under significant channel distortion [49], [50]. Attention-based models, including both self attention and cross-attention mechanisms, have been introduced to better capture long-range correlations in images [51], [52], [53], [54]. Transformer-based JSCC encoders and decoders utilize global receptive fields to improve reconstruction of complex structures and textures [25], [55], [56].

Deep learning-based JSCC replaces the conventional pipeline of separate source coding and channel coding with a single end-to-end trainable encoder-channel-decoder system. It has also expanded to richer communication settings where the physical layer departs from the single-link AWGN or flat-fading assumptions. Several studies investigate JSCC for MIMO systems, where the encoder and decoder must account for the spatial channel matrix, inter-stream interference,

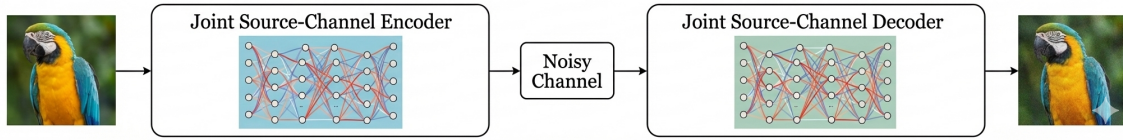


Figure 2.3: Components of the deep JSCC for a typical point-to-point image transmission system

and antenna diversity or multiplexing effects [57], [58], [59], [60]. Other works study OFDM-based JSCC, where differentiable base-band processing blocks are incorporated into the training loop to handle frequency-selective fading and practical transmitter impairments [61]. Robustness under imperfect CSI has also been investigated, since outdated or noisy channel estimates can degrade reconstruction quality [62]. These studies show that deep JSCC can be extended beyond simplified AWGN channels when physical-layer structure is included in the learning framework.

Others exploit channel output feedback links to refine the transmitted representation and improve end-to-end quality under precise block-length constraints, including schemes that use feedback to support variable-length transmission policies and reduce average delay [7], [63]. In addition, lightweight recurrent or state-space backbones have been introduced to reduce complexity while injecting CSI through dedicated embeddings, enabling a single model to adapt more effectively across varying channel conditions [64]. More broadly, refinement mechanisms for time-varying channels and SNR-aware decoding strategies have been proposed to adjust the effective encoding and decoding behaviour online by using instantaneous channel information or decoder-side SNR inference [65]. Surveys on semantic communication further present extensions to multi-user scenarios, relay-based communication, and heterogeneous devices with different SNR budgets [38]. These developments collectively show that deep JSCC is no longer limited to simplified channels but can be systematically adapted by embedding physical-layer structure, channel awareness, and feedback into the end-to-end learning framework. Throughout this thesis, the wireless channel is assumed to follow either an AWGN model or a slow flat Rayleigh fading model. These channel models are widely adopted in the deep JSCC literature because they isolate the effects of additive noise and large-scale fading while remaining computationally tractable for end-to-end training and fair comparison with existing methods. Under these assumptions, reconstruction quality can be evaluated directly using metrics such as PSNR and SSIM.

At the same time, many deep JSCC extensions still rely on a fixed-size encoder bottleneck,

i.e., a constant-dimensional latent representation and therefore a fixed number of channel symbols per image [6], [8], [25], [66]. Also, assumptions about channel statistics at training time, which leaves open the need for unified designs that jointly address channel variability and rate flexibility within a single scalable architecture. In this thesis, the term latent representation refers to the intermediate continuous vector produced by the encoder before (or as) channel symbols are formed, i.e., the bottleneck of encoder output that carries the compressed information about the input through the channel. Many deep JSCC models use a fixed-size latent representation, meaning that the dimensionality of input (equivalently, the number of channel symbols per image) is predetermined by the network architecture and remains constant at inference time. As a result, the effective compression ratio and channel-use budget are fixed unless a different model (or an explicit rate-control mechanism) is used.

Despite these advances, several limitations remain prevalent in the literature. Considering studied reports, the following limitations were discovered:

- A significant limitation is that many deep JSCC architectures rely on dense, fully learned latent representations, while only a limited number of studies integrate CS as an explicit, trainable measurement stage within the end-to-end JSCC pipeline [6], [25], [66], [67], [68]. As a result, many existing methods do not exploit the structural benefits of learned projections, such as controllable measurement budgets and interpretable sampling, that are naturally compatible with bandwidth-limited transmission. This gap is especially visible when the channel-use budget is small and the encoder must decide what information to preserve under severe compression.
- Another limitation concerns rate adaptivity. Most deep JSCC models produce a fixed-dimensional latent representation, which implies a fixed number of channel symbols per image at inference time. While the literature includes approaches such as progressive transmission, layered designs, masking/puncturing strategies, or training multiple models at different rate points, these solutions typically increase operational overhead or provide only coarse rate flexibility. The existing literature still lacks widely adopted mechanisms for fine-grained rate control within a single model, particularly designs that can allocate different communication budgets to different spatial regions or samples based on content

importance [25].

- From another point of view, a shortcoming is that a large portion of the JSCC literature is still developed and evaluated primarily through reconstruction quality, even when the downstream application is inference. In such reconstruction-centric pipelines, the training objective is dominated by pixel-domain distortion terms (e.g., MSE or SSIM), which biases the learned representation toward preserving low-level visual detail that may not be necessary for tasks such as detection or recognition. Although task-oriented and semantic communication has gained attention, many existing task-oriented designs still inherit reconstruction-based intermediates or require recovering dense feature maps before task prediction. There remains a shortage of compressive JSCC frameworks that transmit task-relevant semantic representations and directly optimize task performance under bandwidth constraints and channel impairments without requiring full image reconstruction [9].

Across these limitations, recent papers consistently highlight that deep JSCC is moving toward more adaptive and context-aware communication frameworks. The literature shows increasing interest in learned measurement mechanisms, rate control within a unified model, and task-driven feature-domain transmission. However, open challenges related to principled compressive integration, practical rate adaptability, and task-aligned communication objectives continue to motivate active research in the field [9].

2.2 Compressive Representations for Communication

2.2.1 Compressed Sensing

Conventional signal acquisition is typically grounded in the Shannon-Nyquist sampling principle, which links the sampling rate required for faithful reconstruction to the signal bandwidth. Under the band-limited assumption, avoiding aliasing requires sampling at a rate at least twice the highest frequency component present in the signal. This principle is central to the design of analog-to-digital conversion and has historically encouraged sampling-first pipelines in which signals are acquired at high resolution and then compressed afterward. In imaging, this approach

translates into capturing dense pixel grids even when the underlying scene contains substantial spatial redundancy. The consequence is that a large portion of the acquired samples does not represent new information but must still be stored, processed, and potentially transmitted. In bandwidth- and energy-limited settings such as wireless visual communication, this redundancy becomes costly, since the acquisition stage produces a high-dimensional representation that must then be reduced by separate compression and protection modules, increasing latency and computational burden [69].

CS was introduced as a foundational alternative to this sampling-first paradigm [10], utilizing the observation that many natural signals have far fewer effective degrees of freedom than their ambient dimension suggests. The essential premise is that a signal may be efficiently described by a small number of significant coefficients in an appropriate transform domain, even if it appears dense in their original representation domain. A simple image-based example helps to clarify this point. Consider a 256×256 grayscale image (see Fig. 2.4), which contains 65,536 pixel values. While this is a high-dimensional vector in the pixel domain, its representation in a wavelet or discrete cosine transform domain is often highly compressible, because most coefficients corresponding to smooth regions and fine-scale textures are small, whereas a comparatively small subset of coefficients captures dominant structures such as edges, contours, and large-scale intensity variations. CS exploits this structure by acquiring a reduced set of informative linear measurements that preserve the content needed for recovery, thereby combining acquisition and dimensionality reduction within a single step. In communication-oriented settings, this viewpoint is particularly attractive because it offers a principled mechanism to reduce the number of values that must be conveyed over the channel while still retaining the possibility of reconstructing the source or extracting downstream information reliably. This perspective motivates CS as a natural building block for bandwidth-limited wireless transmission, where the goal is to transmit compact measurements that are informative, structured, and amenable to robust recovery or inference at the receiver.

Let $\mathbf{x} \in \mathbb{R}^n$ denote a signal that is sparse in a transform basis Ψ . The signal can be written as $\mathbf{x} = \Psi \mathbf{s}$, where $\mathbf{s} \in \mathbb{R}^n$ is the corresponding sparse coefficient vector. The CS acquisition process

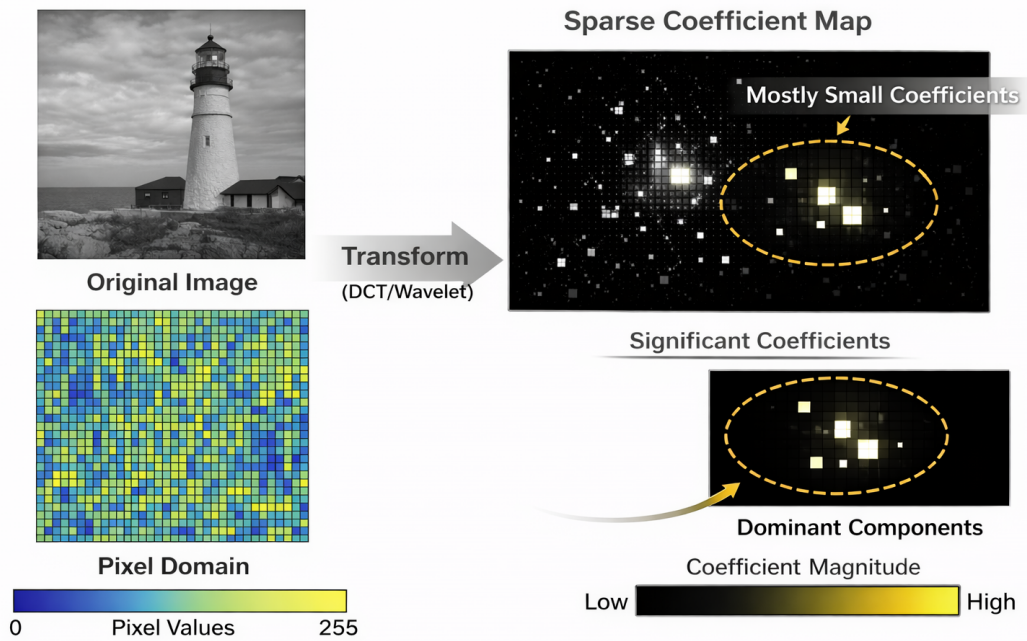


Figure 2.4: Example of a grayscale image (left) and its transform-domain coefficients (right), showing that most coefficients are near zero while a small subset carries the dominant structure.

produces measurements $\mathbf{y} \in \mathbb{R}^m$ through the linear projection shown in Fig. 2.5:

$$\mathbf{y} = \Phi \mathbf{x} = \Phi \Psi \mathbf{s}, \quad (2.1)$$

where $\Phi \in \mathbb{R}^{m \times n}$ with $m \ll n$ is the measurement (a.k.a. sensing) matrix that maps the high-dimensional signal into a lower-dimensional set of linear projections. In classical CS, Φ is often chosen to be *random* (or pseudo-random) because randomness provides a broadly applicable way to ensure that measurements are sufficiently spread across the signal coordinates, leading to incoherence with a wide class of sparsifying bases and enabling recovery guarantees with high probability. Common unstructured choices include i.i.d. Gaussian matrices with entries drawn from $\mathcal{N}(0, 1/m)$, Bernoulli or Rademacher matrices with entries in $\{\pm 1/\sqrt{m}\}$, and more generally sub-Gaussian ensembles, all of which are known to satisfy conditions such as incoherence and the restricted isometry property (RIP) for appropriate m [10].

In practical sensing and communication implementations, fully dense random matrices may be costly to store and apply, so structured measurement operators are often preferred.

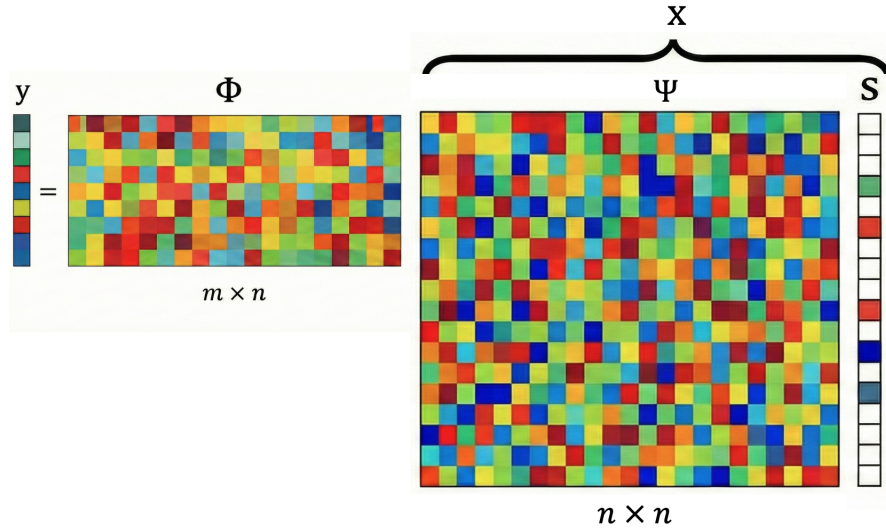


Figure 2.5: Schematic of the CS procedure.

Examples include partial Fourier and partial Hadamard matrices, which enable fast transform-based implementations, as well as random convolution and subsampled circulant operators, which require substantially less storage and support efficient matrix-vector multiplication through convolutional structure [70], [71]. Unlike dense Gaussian matrices that require storing all entries, a circulant sensing matrix is completely specified by a single generating vector and admits fast Fourier transform-based multiplication, reducing both memory and computational complexity [71]. These structured designs aim to retain the key benefits of randomness while improving computational efficiency and hardware feasibility. Under such conditions, the sparse representation can be reconstructed from y by solving:

$$\hat{s} = \arg \min_s \|s\|_1 \quad \text{subject to} \quad \|y - \Phi \Psi s\|_2 \leq \varepsilon, \quad (2.2)$$

where $\varepsilon \geq 0$ is an error tolerance that upper-bounds the measurement mismatch caused by noise and model imperfections (e.g., quantization or approximate sparsity). The estimate of the signal is given by $\hat{x} = \Psi \hat{s}$. These results have had a major theoretical impact in signal processing, imaging, and communications.

Although classical CS is mathematically elegant, several limitations arise when it is applied to real images and practical wireless communication systems. Natural images are not strictly sparse; their sparsity is only approximate and varies with edges, textures, and illumination

conditions [18]. A single fixed measurement matrix is rarely optimal for such diverse structures. Furthermore, reconstruction algorithms such as basis pursuit, iterative shrinkage-thresholding, and orthogonal matching pursuit rely on iterative solvers that are computationally expensive [13]. Their convergence behavior is sensitive to measurement noise and modeling inaccuracies. In wireless communication scenarios, where measurements are corrupted by channel noise, quantization, and fading, these issues become more pronounced [33]. The additive distortion introduced by the channel propagates through the iterative recovery process and often produces artefacts or unstable reconstructions. Reports on CS concepts and applications consistently emphasize that classical CS is not well suited for real-time, noise-sensitive environments unless combined with additional mechanisms that handle channel perturbations [72].

The emergence of deep learning has created a new paradigm known as deep compressive sensing. Deep CS replaces fixed linear sensing matrices and iterative reconstruction algorithms with learned operators and deep neural networks [34]. In this setting, the measurement process is parameterized as a trainable linear operator,

$$\mathbf{y} = b_{\theta}(\mathbf{x}) = \Phi_{\theta}\mathbf{x}, \quad (2.3)$$

where Φ_{θ} is jointly optimized with a reconstruction network c_{θ} . The reconstruction stage is typically implemented using convolutional autoencoders, residual networks, or transformer-based architectures [73], [74], [75], [76]. Training is performed end-to-end by minimizing:

$$\mathcal{L}(\theta) = \|\mathbf{x} - c_{\theta}(b_{\theta}(\mathbf{x}))\|_2^2, \quad (2.4)$$

or a perceptual loss suited to the task. This formulation allows the measurement matrix to adapt to natural image statistics, circumventing the need for hand-crafted sensing operators or explicit sparsity modeling.

Deep CS frameworks such as ReconNet [77], DR²Net [78], AMP-Net [79], ISTA-Net [80], and ADMM-Net [81] have demonstrated that learned sensing and reconstruction consistently outperform classical CS, especially at low sampling ratios. By replacing iterative optimization with feed-forward neural networks, these models achieve higher reconstruction fidelity while

significantly reducing inference latency. Survey papers on deep CS further emphasize that many neural architectures can be interpreted as unrolled or learned variants of classical optimization algorithms, while benefiting from a richer space of nonlinear transformations [13], [34]. As a result, deep CS has become an important research direction for image reconstruction tasks involving dimensionality reduction, and it has shifted the role of CS from a purely mathematical acquisition theory toward a flexible, learnable representation mechanism.

2.2.2 Compressive Sensing as a Tool for Communication Systems

The interpretation of CS as a learnable representation mechanism naturally extends its relevance beyond standalone reconstruction problems and into the design of communication systems. In wireless communication, the objective is not only to recover signals accurately, but also to reduce sampling, transmission, and processing overhead under bandwidth, latency, and energy constraints. From this perspective, CS provides a principled way to acquire compact, information-bearing measurements that can reduce signaling overhead and exploit inherent sparsity in channels, interference patterns, and transmitted data. Motivated by these advantages, a substantial body of work has explored the use of CS in modern communication pipelines [82], [83], [84]. In particular, Choi *et al.* survey how CS is applied across core wireless communication problems, including sparse channel estimation, interference cancellation, direction estimation, spectrum sensing, and sparse symbol detection, and discuss practical system-level considerations when deploying CS-based techniques [82]. Hayashi *et al.* provide a user-oriented overview of CS for communications, covering key recovery algorithms and sensing-matrix properties, and summarizing applications such as channel estimation, cognitive radio spectrum sensing, array processing, and multiple-access systems [83]. Gao *et al.* focus on next-generation wireless systems and argue that sparsity can be exploited to reduce overhead and hardware burden in scenarios including massive MIMO channel estimation and feedback, sparse signal detection in non-orthogonal multiple access, and sub-Nyquist wideband spectrum sensing [84].

Since deep JSCC frameworks require compact representations that are robust to channel noise and adaptable to bandwidth constraints, they naturally synergize with deep CS modules. Deep CS provides a learnable dimensionality reduction mechanism that can be optimized jointly

with a communication encoder. Instead of treating CS as a separate source coding block, deep CS can be integrated directly into the JSCC encoder so that compression and robustness are learned together. Some reports on semantic communication describe a general shift toward unified end-to-end communication systems where internal compression, channel adaptation, and inference tasks are co-optimized within a single neural architecture [85], [86], [87], [88].

The literature includes several attempts to integrate CS into communication pipelines. Block-based compressive sensing applies fixed or learned sensing matrices to image patches, followed by a JSCC encoder that maps the compressed measurements to channel symbols [20]. This reduces the spatial resolution early in the pipeline and improves bandwidth efficiency. Other works use global learned sampling matrices that directly compress the entire image into a lower-dimensional embedding that is transmitted through a deep communications encoder [21]. Studies on semantic image transmission also identify emerging hybrid models where CS based sampling is combined with deep JSCC for both reconstruction and classification tasks [89], [90].

However, despite the success of deep CS, several limitations persist in its integration with communication systems. A major limitation is the use of fixed sampling ratios. Most deep CS models adopt a single compression rate that remains constant during inference [91], [92]. This limits the ability of the system to adapt to variations in channel bandwidth or SNR. Existing rate-adaptive CS techniques often rely on multiple sensing networks or handcrafted decision policies, which increase complexity and memory usage [93], [94]. Another limitation is that existing hybrid CS-JSCC systems treat the sensing module as external to the end-to-end communication model. Training the sensing matrix separately from the decoder prevents the model from fully exploiting joint optimization [95]. Deep CS research also highlights challenges in robustness, noting that compressed representations may still be sensitive to noise and that embedding manifolds can be distorted by channel perturbations.

Current research trends show a movement toward unified compressive JSCC architectures that integrate sampling, representation learning, and channel mapping in a coordinated manner [67], [68], [96]. These models attempt to learn sensing operators that preserve channel robust features, develop content-adaptive or rate-adaptive sampling mechanisms, and incorporate channel awareness directly into the compressive encoding stage. Surveys on CS and semantic

communication suggest that such approaches will be important for future communication systems where bandwidth efficiency, robustness, and task-specific relevance must all be achieved simultaneously.

In the context of this thesis, CS is not used as an isolated compression method but as an integral component of a deep learning-based encoder within the JSCC framework. In all designed JSCC frameworks in the next chapters, a trainable CS module is employed to compress the input prior to channel mapping, thereby reducing redundancy and improving transmission efficiency. Rather than acting only as a dimensionality-reduction layer inside the encoder, CS offers several properties that are particularly attractive for JSCC and are not naturally provided by conventional entropy-coded compression pipelines. First, CS provides an explicit and interpretable rate control mechanism through the number of measurements m , which allows the transmitter to adjust the number of channel symbols in a fine-grained manner without redesigning a discrete bit allocation and without relying on fragile variable-length bitstreams. Second, widely used random or pseudo-random sensing operators distribute source information across measurements in a near-uniform manner, a property often referred to as the democracy of random projections, which makes the representation more tolerant to the loss or corruption of a subset of measurements and leads to graceful quality degradation when channel conditions deteriorate. Third, the linear structure of CS measurements aligns naturally with analog or real-valued transmission and can be integrated into end-to-end learned JSCC models with low encoder complexity, while still enabling strong reconstruction quality at low sampling ratios. These properties motivate treating CS as an internal and trainable module inside JSCC, so that the learned measurements are not only compact but also optimized for robustness under channel noise, fading, and bandwidth variation.

2.3 Adaptive JSCC Frameworks

As deep JSCC has evolved, a broad body of work has emerged to overcome the rigidity of early fixed-rate designs and to enable communication systems that can respond to fluctuating channel quality and bandwidth availability. Modern wireless environments frequently experience variations in SNR, temporal changes in available spectrum, and heterogeneous device constraints.

These practical factors motivate JSCC architectures that can adjust their behaviour dynamically. Two major research directions address this need: rate-adaptive JSCC and SNR-adaptive JSCC. Both areas are highlighted in recent surveys on deep learning-based adaptive communication [8], [9], [97], and together form the foundation of current research on flexible and robust semantic communication.

Rate adaptation has long been recognized as essential for bandwidth-limited links, particularly when channel resources vary unpredictably. The earliest rate-adaptive systems relied on progressive or layered transmission [98], [99], [100]. In these schemes, the encoder transmits a base representation and optional enhancement layers, allowing the receiver to reconstruct the source at multiple qualities depending on the number of layers received. While conceptually simple and grounded in scalable source-coding principles, layered approaches require multiple encoder passes, additional signalling, and provide only coarse control over the actual transmission rate. As a result, they are not well suited for real-time, learning-driven JSCC.

To achieve more fine-grained adaptation, several studies have explored variable-length latent representations and multi-model selection strategies. In these methods, the encoder produces a latent vector whose length depends on the desired rate, or a controller selects one among multiple JSCC networks trained at different compression ratios [101], [102], [103]. These designs improve flexibility across bandwidth regimes. However, they scale poorly with the number of supported rates because storing and switching between multiple networks increases memory and computational overhead. They also treat rate control mainly as a discrete selection problem rather than as a fully learnable component of the communication pipeline.

A more recent direction incorporates learning-based rate controllers into the encoder, enabling the network to decide which parts of the representation should be transmitted under a given bandwidth constraint. Controllers based on attention, feature gating, and differentiable approximations of discrete decisions have been proposed, offering improved granularity and adaptivity [104]. However, these mechanisms often add architectural complexity or rely on heuristic importance measures, and many lack a principled connection to compressive sensing or latent sparsity. Surveys such as [9] observe that the literature still lacks unified designs capable of rate adaptation without relying on multiple parallel models or handcrafted selection rules.

In parallel with these developments, SNR-adaptive JSCC has been widely studied as a way to mitigate the degradation that occurs when a model trained at a specific SNR operates under mismatched conditions. Several architectures integrate the channel SNR directly into the encoding or decoding process [66], [105]. Some concatenate the SNR to the input or latent vectors, while others use feature-modulation mechanisms such as feature-wise linear modulation (FiLM) layers or adaptive BatchNorm to adjust activations based on the estimated noise level [106]. These strategies significantly improve robustness because they allow the representation to adapt its statistical properties to the expected noise regime.

More advanced SNR-adaptive designs modulate both the encoder and decoder, incorporate channel-attention mechanisms, or learn to reshape the latent representation depending on the noise variance [107]. These methods generalize well across channel conditions and alleviate the need to train different models for different SNR levels. Nonetheless, they retain a key limitation: the latent dimensionality remains fixed, and therefore the transmission rate does not change even when channel bandwidth varies. Surveys including [9] note that this disconnect between SNR adaptivity and rate adaptivity remains a central barrier to deployment, since real-world systems must adapt to both noise levels and available channel resources simultaneously.

In addition to SNR variation, the type of channel impairment is also important. Much of the deep JSCC literature evaluates robustness primarily under AWGN, while practical wireless systems may experience multiplicative fading, non-Gaussian impulsive noise, or structured interference [8]. These impairments affect transmitted latent representations in different ways, so robustness to SNR variation alone does not fully characterize performance under diverse channel conditions. This motivates the use of both AWGN and Rayleigh fading evaluations in this thesis [6], [108].

Hybrid adaptive frameworks have begun to appear, combining SNR awareness with dynamic feature pruning or latent gating. For instance, some works use sparsification rules, entropy-based pruning, or differentiable switching mechanisms to reduce transmission cost when channel conditions improve [109], [110]. Others integrate transformer-based architectures whose attention patterns inherently adjust representational capacity across spatial patches or MIMO subchannels [23], [60], [111]. While these models improve robustness and interpretability, they continue to

produce fixed-size latent vectors and therefore cannot achieve true rate adaptivity.

To summarize, the current landscape of adaptive JSCC research shows a clear progression toward models that allocate communication resources based on both the content being transmitted and the channel environment. However, existing approaches tend to provide only partial adaptivity: layered systems offer coarse rate control, multi-model methods introduce high overhead, learning-based controllers add complexity, and SNR-adaptive models operate at fixed dimensionality. This underscores the need for unified frameworks that can simultaneously adjust sampling density, latent dimensionality, and feature modulation within a single end-to-end trainable pipeline capable of generalizing across diverse channel characteristics. The adaptive compressive JSCC architectures introduced in later chapters are motivated by these exact architectural gaps, seeking to combine fine-grained rate control with robust SNR-conditioned encoding in a principled and scalable manner under both additive and multiplicative channel distortions.

2.4 Semantic and Task-Oriented JSCC Frameworks

Semantic and task-oriented communication constitute another major research direction. Instead of aiming for pixel-based reconstruction, semantic communication focuses on conveying the information needed to execute a downstream task such as classification, detection, captioning, or speech understanding [36]. In these systems, the communication encoder, the decoder, and the task network are often trained jointly. Here, the task network refers to the downstream inference model (e.g., a classifier, detector, or captioning model) that maps the received semantic representation to the final task output. The objective functions may combine task accuracy, signal distortion, and semantic consistency [35]. A number of works demonstrate that semantic features can be transmitted with significantly fewer channel uses than required for pixel-accurate reconstruction. For image classification, learned semantic communication frameworks show that classification accuracy can be preserved even when channel bandwidth is reduced by large factors [112], [113], [114]. For object detection, systems transmit feature maps or attention-guided representations that preserve information relevant to bounding box regression and class predictions [115], [116]. Surveys on semantic communication highlight additional benefits,

including reduced sensitivity to channel disturbances, improved generalization to unseen channel conditions, and support for multi-task inference [35], [117], [118].

Feature domain or embedding domain JSCC extends these concepts further by transmitting deep features extracted by a pretrained task network (e.g., a CNN or Transformer feature extractor) rather than raw pixels [119], [120]. In feature domain communication, the encoder uses a convolutional or transformer network to produce embeddings that represent semantic content, which are then transmitted through a channel model [121]. Feature domain JSCC aligns well with recognition [122], retrieval [123], and detection [124] tasks, where such embeddings already serve as input to downstream decision networks. Feature domain transmission significantly reduces bandwidth and simplifies receiver processing because the receiver no longer needs to perform full image reconstruction. However, it also introduces important challenges. Deep embeddings tend to lie on structured low-dimensional manifolds, and compression or channel noise can shift them away from these manifolds. Such distortions may have a disproportionate impact on downstream tasks, which are often highly sensitive to embedding perturbations. Surveys on semantic communications emphasize the need for robust decoding strategies, such as denoising networks, distribution-aware feature restoration, or generative refinement models, to ensure consistency with the downstream task [117], [118].

Another trend in the literature involves integrating compression or dimensionality reduction into semantic and feature domain communication frameworks [33], [86], [125]. Some works use linear projections to reduce feature dimensionality, while others embed autoencoder style bottlenecks within the communication system. Surveys on CS and deep learning observe that compressive sampling provides a structured way to control dimensionality and may offer advantages for bandwidth-limited semantic transmission [9]. Nevertheless, many existing systems rely on fixed projection matrices or static bottleneck dimensions [32]. Rate adaptive compression, which allows the size of the transmitted representations to vary based on content complexity or channel quality, is still relatively rare. Some recent efforts introduce adaptive bit allocation [126], attention-driven prioritization [127], or discrete rate selection mechanisms [128], but fully learned, continuous rate control is still largely unexplored.

Current surveys also highlight several gaps in semantic and feature-domain JSCC. A large

portion of existing work focuses on high-level classification tasks, while fewer studies address tasks that require spatial localization, such as face or object detection [8], [9], [32]. Even fewer models operate entirely in the embedding domain for identity recognition or fine-grained retrieval. Another gap is the lack of unified frameworks that jointly consider compression, robustness, channel awareness, and semantic task performance. Many architectures still use heuristic or modular designs in which compression and semantic encoding are trained separately, limiting the benefit of end-to-end optimization. These challenges motivate further research into rate-adaptive, semantic, and feature-based communication systems that can operate reliably under channel variability and bandwidth constraints.

2.5 Evaluation Datasets and Performance Metrics

The evaluation of deep JSCC, CS-based frameworks, and task-oriented semantic communication requires datasets and metrics that reflect both reconstruction quality and downstream inference performance. In the wider literature, two broad categories of benchmarks are commonly used. The first category focuses on pixel-level image reconstruction, where the receiver aims to recover a visually faithful version of the transmitted image. The second category focuses on task-oriented inference, where the quality of communication is measured by the success of a downstream machine learning task rather than by image fidelity alone. This section reviews the datasets and performance metrics used in this thesis, together with related benchmarks and measures that are commonly used in the wider JSCC and semantic communication literature.

2.5.1 Evaluation Datasets

Standard Image Reconstruction Benchmarks

Image reconstruction benchmarks are used to evaluate how well a communication system preserves pixel-level and structural image information under bandwidth and channel constraints. Commonly used datasets include small-scale natural image datasets for fast prototyping, large-scale datasets for training, and fixed test sets for reproducible evaluation.

- **CIFAR-10:** CIFAR-10 contains 60,000 RGB images of size 32×32 pixels, evenly dis-

tributed across 10 semantic classes [129]. In deep JSCC and compressive JSCC studies, CIFAR-10 is widely used as a compact benchmark for testing encoder-decoder architectures, latent representations, channel robustness, and training behaviour under limited bandwidth and low SNR conditions.

- **ImageNet:** ImageNet contains approximately 1.2 million training images across 1,000 object categories [130]. It provides a large and diverse natural image distribution and is commonly used for training deep image transmission models on randomly cropped patches, often of size 224×224 . Its diversity helps reduce overfitting to small datasets and enables evaluation of models under more realistic natural-image statistics.
- **Kodak:** The Kodak dataset contains 24 uncompressed colour images and is widely used as a standard test benchmark for image compression and reconstruction [131]. Although it is small, it includes natural scenes with rich textures, smooth regions, edges, and illumination variations. Since it is normally used only for testing, it provides a useful benchmark for evaluating generalization to unseen high-quality images.
- **Other reconstruction datasets in the literature:** In addition to the datasets used in this thesis, the wider image restoration and JSCC literature often uses datasets such as BSD100 [66], Set5 and Set14 [64], Urban100 [105], DIV2K [132], and MS-COCO for evaluating reconstruction, super-resolution, perceptual quality, and generalization [8]. These datasets are not the primary focus of this thesis, but they represent common alternatives for future comparison and extension.

Downstream Task-Oriented and Facial Datasets

Task-oriented semantic communication requires datasets where the communication objective is linked to a downstream task such as detection, classification, verification, or identification. In such cases, a visually perfect reconstruction is not necessary; instead, the transmitted representation must preserve task-relevant semantic information.

- **WIDER Face:** WIDER Face is a large-scale face detection benchmark containing 32,203 images and 393,703 annotated faces with significant variation in scale, pose, occlusion,

illumination, and background clutter [133]. It is suitable for evaluating task-oriented JSCC systems where the receiver must preserve spatially localized information for bounding-box prediction under channel noise.

- **ImageNet-1K:** ImageNet-1K is also widely used as a classification benchmark with 1,000 object categories [130]. In task-oriented communication, it can be used to evaluate whether a transmitted semantic representation preserves class-discriminative information without requiring full image reconstruction.
- **Labeled Faces in the Wild (LFW):** LFW contains more than 13,000 facial images collected from unconstrained web environments [134]. It is commonly evaluated using 6,000 verification pairs under a ten-fold protocol. In feature-domain communication, LFW tests whether the transmitted and reconstructed identity embeddings preserve the angular similarity relationships required for face verification.
- **MegaFace:** MegaFace is a large-scale face recognition benchmark where probe images are matched against a gallery containing up to one million distractor identities [135]. It evaluates the scalability and robustness of identity embeddings under large-scale inter-class competition, making it suitable for testing semantic communication systems designed for face identification.
- **Large-scale facial training datasets:** Robust face recognition and detection systems often rely on large-scale training corpora such as CASIA-WebFace [136], VGGFace2 [137], MS-Celeb-1M [138], and synthetic datasets such as FRCSyn [139]. These datasets provide diverse identities, pose variations, illumination conditions, and demographic variability, helping train feature extractors that remain stable under compression and channel distortion.
- **Other task-oriented datasets in the literature:** Beyond the datasets considered in this thesis, task-oriented semantic communication studies may use datasets such as MS-COCO for object detection and captioning [140], Pascal VOC for detection and segmentation [141], Cityscapes for autonomous-driving scene understanding [142], ADE20K for semantic

segmentation [143], and KITTI for perception in intelligent transportation systems [144]. These datasets are relevant for future extensions of task-oriented JSCC beyond face-related and classification tasks.

2.5.2 Performance and Communication Metrics

Pixel-Level Reconstruction Metrics

For reconstruction-oriented JSCC systems, the receiver output $\hat{\mathbf{x}}$ is compared with the original source image $\mathbf{x}$. The most common metrics are Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM).

The MSE between an image $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$ is defined as

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2, \quad (2.5)$$

where N is the total number of pixels or image elements. MSE measures average pixel-wise distortion, but it does not always correlate well with perceived visual quality.

PSNR is calculated from the MSE as:

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{Peak}^2}{\text{MSE}} \right) \quad \text{dB}, \quad (2.6)$$

where Peak denotes the maximum possible pixel value, typically 255 for 8-bit images or 1 for normalized images. PSNR is widely used in image compression and JSCC because it is simple, reproducible, and allows direct numerical comparison across methods. However, it mainly reflects pixel-level fidelity and may not fully capture perceptual quality.

SSIM measures perceived structural similarity by comparing luminance, contrast, and structural information between $\mathbf{x}$ and $\hat{\mathbf{x}}$ [145]. For two image patches x and $\hat{x}$, SSIM is defined as:

$$\text{SSIM}(x, \hat{x}) = \frac{(2\mu_x \mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)}, \quad (2.7)$$

where μ_x and $\mu_{\hat{x}}$ are local means, σ_x^2 and $\sigma_{\hat{x}}^2$ are local variances, and $\sigma_{x\hat{x}}$ is the local covariance between the original and reconstructed patches. The constants C_1 and C_2 stabilize the division

when the denominator is small. SSIM ranges from 0 to 1, where a value closer to 1 indicates higher structural similarity. Compared with PSNR, SSIM better reflects structural preservation, such as edges and local contrast, and is therefore frequently used together with PSNR in image reconstruction studies.

Although PSNR and SSIM are standard in JSCC literature, they do not fully represent human perceptual quality. Two images may have similar PSNR or SSIM values but differ significantly in perceptual sharpness, texture realism, or semantic plausibility. For this reason, perceptual metrics are increasingly used in image restoration, compression, and learned communication systems. One widely used perceptual metric is Learned Perceptual Image Patch Similarity (LPIPS) [146]. LPIPS compares images in the feature space of a pretrained deep neural network rather than directly in the pixel domain. Given feature activations $\phi_l(\cdot)$ from layer l of a pretrained network, LPIPS can be written generally as:

$$\text{LPIPS}(x, \hat{x}) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|\mathbf{w}_l \odot (\hat{\phi}_l(x)_{hw} - \hat{\phi}_l(\hat{x})_{hw})\|_2^2, \quad (2.8)$$

where $\hat{\phi}_l(\cdot)$ denotes channel-normalized feature activations at layer l , H_l and W_l are the spatial dimensions of that feature map, $\mathbf{w}_l$ is a learned channel-wise weighting vector, and $\odot$ denotes element-wise multiplication. Lower LPIPS values indicate greater perceptual similarity. LPIPS is especially useful when evaluating perceptual image quality, but it is computationally heavier than PSNR and SSIM and depends on the chosen pretrained feature extractor. Therefore, in many JSCC studies, LPIPS is used as a complementary metric rather than a replacement for PSNR and SSIM.

Other perceptual or distributional metrics also appear in the wider literature, including Fréchet Inception Distance (FID), feature similarity metrics, and no-reference image quality measures [8]. These metrics are particularly relevant for generative reconstruction and semantic communication systems, but their interpretation depends strongly on the evaluation setting.

Semantic Inference and Downstream Task Metrics

For task-oriented communication, the goal is not to reconstruct the input image but to preserve the information needed for a downstream task. Therefore, evaluation must use task-specific

metrics. For detection tasks, precision and recall are commonly defined as:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad (2.9)$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively. The F1-score is the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (2.10)$$

In wireless detection tasks, F1-score measures whether channel distortion affects both localization and classification reliability. For bounding-box based detection, intersection over union (IoU), generalized IoU (GIoU), and mean average precision (mAP) are also widely used in the broader detection literature [122].

For face verification, a common metric is the True Accept Rate at a fixed False Accept Rate, denoted as TAR@FAR. The false accept rate measures the proportion of impostor pairs incorrectly accepted as matches, while the true accept rate measures the proportion of genuine pairs correctly accepted [122]. Reporting TAR at a strict FAR, such as $FAR = 10^{-3}$, evaluates whether the transmitted embedding remains discriminative under a low false-match tolerance. For identification, Rank-1 accuracy is commonly used. It measures the percentage of probe samples for which the correct identity obtains the highest similarity score among all gallery candidates [112]. This metric is particularly important for large-scale identity retrieval and face identification benchmarks such as MegaFace. For classification tasks, top-1 and top-5 accuracy are standard metrics. Top-1 accuracy measures whether the predicted class with the highest probability matches the ground truth label, while top-5 accuracy measures whether the correct label appears among the five most confident predictions [113]. These metrics are commonly used for ImageNet-1K and other large-scale classification benchmarks.

Communication Efficiency and Resource Metrics

In JSCC and semantic communication, performance must be interpreted together with communication cost. A method that achieves high reconstruction or task accuracy by transmitting a

much larger representation may not be efficient. Therefore, bandwidth and resource metrics are essential. The bandwidth compression ratio is commonly defined as:

$$R = \frac{k}{n}, \quad (2.11)$$

where k is the number of transmitted complex channel symbols and $n = H \times W \times C$ is the number of source symbols in the input image. A smaller value of R indicates a more compressed transmission. In rate-adaptive systems, R may vary across images or blocks, so the average effective bandwidth compression ratio should also be reported.

For feature-domain communication, channel uses per pixel (CuPP) is used to normalize the transmitted feature dimension by the input image size:

$$\rho = \frac{d_{\text{ch}}}{H' \times W'}, \quad (2.12)$$

where d_{ch} is the number of transmitted feature or channel dimensions, and $H' \times W'$ is the spatial size of the processed input region. CuPP is useful for comparing feature-domain methods with image-domain methods, although care must be taken to state whether the numerator counts real dimensions, complex symbols, or coded digital symbols.

Other communication-related measures include bits per pixel (bpp), channel uses per source symbol, average selected rate, side-information overhead, and coding rate in digital baselines. In adaptive schemes, rate-control headers or block-wise rate indices should be counted or discussed, especially when the side information is required for correct decoding.

Finally, practical deployment also depends on computational cost. End-to-end latency measures the processing time required for encoding, transmission-related mapping, decoding, reconstruction, or task inference. Parameter count and floating-point operations, or multiply-accumulate operations (MACs), are also useful for estimating memory and computational complexity. These measures are particularly important when models are intended for edge devices, mobile platforms, or real-time wireless perception systems.

Deep Compressive JSCC Frameworks for Wireless Image Transmission

3.1 Introduction and Related Works

The previous chapters have outlined the motivation for moving beyond separation-based wireless image transmission and have reviewed the main developments in deep JSCC, CS, and task-oriented communication. This chapter focuses on proposing deep compressive JSCC methodologies for wireless image transmission under various channel conditions. In these architectures, a learned CS-based module is embedded inside the JSCC framework and trained jointly with the encoder and decoder so that dimensionality reduction and channel robustness are optimized together.

Classical deep JSCC models map images directly to channel symbols using convolutional or transformer-based encoder-decoder pairs [6], [25], [51]. These systems have demonstrated clear advantages over separation-based schemes, including graceful degradation of reconstruction quality (e.g., PSNR/SSIM) as SNR decreases, robustness to channel mismatch (e.g., when the test SNR or fading statistics differ from those used in training), and strong performance under strict bandwidth compression ratios (i.e., when the number of channel uses k is much smaller than the number of source samples n , yielding a small k/n). However, most deep JSCC designs rely on latent feature tensors whose size is fixed by the network architecture [8]. This fixed latent

dimensionality introduces two practical issues in bandwidth-limited operation: (i) when k is small, the encoder may still spend representational capacity on visually redundant content, and (ii) adjusting the compression level to a new bandwidth budget typically requires redesigning the architecture or training and storing separate models for different k/n operating points.

These considerations motivate JSCC architectures to include an explicit and communication-oriented rate knob that can be matched to the available channel uses. CS provides a useful interface for this purpose because it represents compression through a measurement vector of length mv that is obtained via real-valued linear projections. When embedded inside an end-to-end learned JSCC pipeline, the measurement dimension mv directly controls how many values are produced before channel mapping, making the rate control mechanism explicit and interpretable. In addition, CS measurements are analog-valued, so channel impairments appear as additive perturbations on the measurements rather than bitstream failures, supporting smooth degradation under worsening channel conditions. It should be emphasized that CS does not automatically provide rate adaptivity by itself; rather, it becomes rate-flexible when the system is designed to vary mv (or select among multiple measurement dimensions) according to the bandwidth budget. Finally, CS operators can be implemented with low complexity using random or structured projection matrices (e.g., partial orthogonal transforms or subsampled Hadamard/Fourier-type operators), while modern deep CS further improves performance by learning both the sensing operator and the reconstruction network from data [10], [13], [77], [80]. Together, these properties make CS an attractive, rate-controllable dimensionality-reduction module that can be integrated with deep JSCC to better align compression and robustness with bandwidth-limited wireless image transmission.

Early attempts to combine CS with communication systems were based on a modular approach in which CS was used as an external compression tool and was followed by analog or hybrid digital-analog transmission schemes such as SparseCast [15]. While these systems benefit from the linear structure of CS and exhibit graceful SNR degradation, they do not use end-to-end learning and do not treat CS as part of a deep JSCC encoder. More recent studies have begun to embed compressive sampling directly into communication architectures [16], [18], [19]. Previous CS based JSCC methods apply sensing matrices to image patches and then

feed the resulting measurements into a JSCC encoder-decoder pair [20], [21]. Global sampling variants utilize a learnable linear layer to compress the full image into a low-dimensional latent vector, which is then transmitted through a neural channel encoder [147]. Results in these works indicate that internal compressive modules can reduce channel usage and improve performance compared with pure deep JSCC or CS-based schemes, especially at low bandwidth compression ratios and low to medium SNRs.

Despite these advances, the literature still lacks a systematic study of deep compressive JSCC architectures that treat the CS modules as an integral part of the JSCC encoder and that are evaluated across a wide range of SNR and bandwidth conditions. Many existing designs use loosely coupled training where the compression and decoding stages are not jointly optimized [8]. In addition, the impact of the sensing module on reconstruction quality, robustness, and computational complexity has not been analyzed in detail across multiple architectures.

This chapter addresses these gaps by introducing a set of deep compressive JSCC architectures for wireless image transmission that are developed in a progressive manner. The proposed designs begin with a convolutional JSCC pipeline, named BCS-JSCC, that embeds a deep CS-based sensing and reconstruction module, and demonstrate the resulting gains over standard deep JSCC baselines. Then, an enhanced variant of BCS-JSCC named DCS-JSCC is developed that strengthens the encoder/decoder design and refines the training objective. Finally, these methods are consolidated into a more comprehensive framework, entitled as CS-JSCC, with improved architecture and a broader experimental analysis.

To provide a clear and structured presentation, the remainder of this chapter is organized as follows. Section 3.2 introduces the common system model and notation used throughout the chapter. Section 3.3 presents the proposed architecture of BCS-JSCC and its results. Section 3.4 describes the design of DCS-JSCC and corresponding evaluation. Section 3.5 introduces the final fixed-rate deep compressive JSCC framework, CS-JSCC, and reports its experimental findings. Section 3.6 compares the proposed frameworks, and Section 3.7 summarizes the main insights and motivates the rate-adaptive architectures presented in Chapter 4.

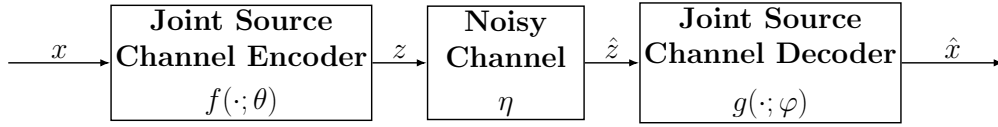


Figure 3.1: Components of the system model for JSCC frameworks.

3.2 System Model

The considered pipeline in this chapter is a point-to-point wireless image transmission system illustrated in Fig. 3.1. An input image of size $H \times W \times C$ (height, width, and number of color channels) is represented by a vector $\mathbf{x} \in \mathbb{R}^n$, where $n = H \times W \times C$ and $\mathbb{R}$ denotes the set of real numbers. The joint source-channel encoder implements an encoding function $f(\cdot; \theta) : \mathbb{R}^n \rightarrow \mathbb{C}^k$, which maps the real-valued image vector $\mathbf{x}$ to a complex-valued vector $\mathbf{z} \in \mathbb{C}^k$:

$$\mathbf{z} = f(\mathbf{x}; \theta) \in \mathbb{C}^k \quad (3.1)$$

Here, k represents the number of symbols transmitted over the wireless channel, θ denotes the set of trainable parameters of the encoder, and $\mathbb{C}$ is the set of complex numbers. In other words, the encoder compresses the n -dimensional real vector into k complex channel inputs with $k \ll n$. The channel input $\mathbf{z}$ satisfies the average transmit power constraint:

$$\frac{1}{k} \mathbb{E}[\mathbf{z} \mathbf{z}^*] \leq P \quad (3.2)$$

where $\mathbf{z}^*$ denotes the complex conjugate transpose of $\mathbf{z}$, and P specifies the average transmit power.

The channel is modeled as a function $\eta : \mathbb{C}^k \rightarrow \mathbb{C}^k$. In this thesis, two widely adopted scenarios are considered: AWGN and slow Rayleigh fading. For the AWGN case, the received signal $\hat{\mathbf{z}} \in \mathbb{C}^k$ is given by

$$\hat{\mathbf{z}} = \eta_n(\mathbf{z}) = \mathbf{z} + \mathbf{w} \quad (3.3)$$

where $\mathbf{w} \in \mathbb{C}^k$ is a vector of independent and identically distributed (i.i.d.) samples from the circularly symmetric complex Gaussian distribution $\mathcal{CN}(0, \sigma^2 \mathbf{I})$. Here, σ^2 represents the

average noise power.

For the slow fading case, the standard Rayleigh fading model is adopted. The channel effect is modeled as a multiplicative gain applied to the transmitted signal:

$$\eta_h(\mathbf{z}) = h\mathbf{z} \quad (3.4)$$

where $h \sim \mathcal{CN}(0, H_c)$ is a complex Gaussian random variable representing the channel coefficient. Combining both fading and Gaussian noise effects yields

$$\eta(\mathbf{z}) = \eta_n(\eta_h(\mathbf{z})) = h\mathbf{z} + \mathbf{w} \quad (3.5)$$

At the receiver, the joint source-channel decoder is defined by a mapping $g(\cdot; \varphi) : \mathbb{C}^k \rightarrow \mathbb{R}^n$, which reconstructs an estimate $\hat{\mathbf{x}}$ of the original image:

$$\hat{\mathbf{x}} = g(\hat{\mathbf{z}}; \varphi) = g(\eta(f(\mathbf{x}; \theta)); \varphi) \quad (3.6)$$

where φ represents the decoder parameters and $\hat{\mathbf{x}} \in \mathbb{R}^n$ is the recovered image. Throughout this thesis, it is assumed that the decoder has access to perfect CSI. Both the encoder $f(\cdot; \theta)$ and decoder $g(\cdot; \varphi)$ are implemented using deep neural networks, as described in the following sections.

3.3 BCS-JSCC: Block Compressive Sensing based JSCC for Wireless Image Transmission

In this section, a compressive JSCC algorithm, with a fixed sampling rate, named BCS-JSCC, is described. This proposed framework utilizes deep block-based CS (BCS) modules integrated within the end-to-end framework. The introduction of BCS-based sensing module allows the JSCC encoder to utilize the properties of CS to enhance the performance of the deep learning based JSCC scheme. Moreover, application of BCS concepts in the proposed JSCC decoder using a deep learning based reconstruction module has resulted in improved image reconstruction

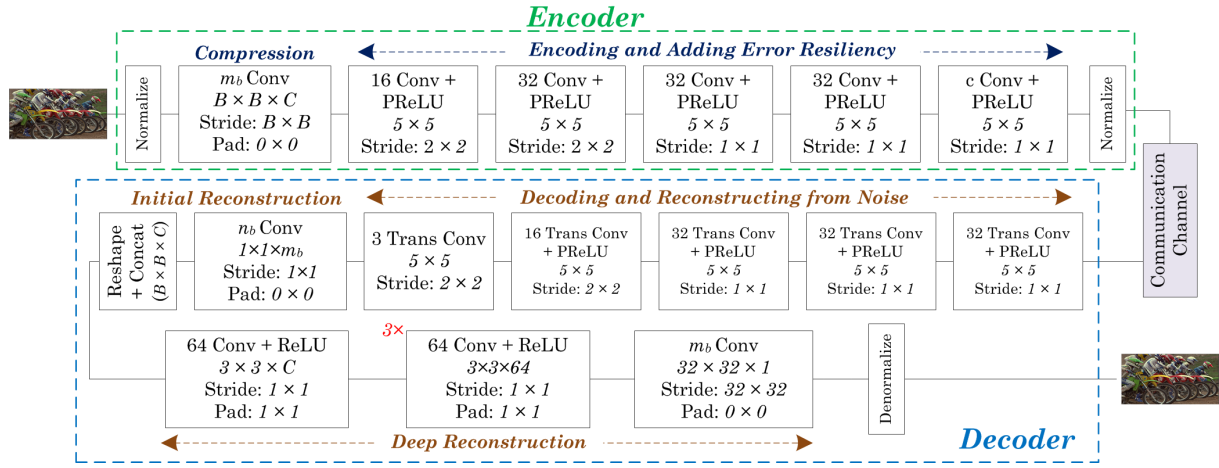


Figure 3.2: Architecture of the proposed BCS-JSCC framework

for various wireless image transmission scenarios. In the following subsections, the proposed architecture is presented, and then its performance is evaluated in comparison with well-known baselines.

3.3.1 Model Architecture

Encoder Design

The structure of the BCS-JSCC encoder and decoder, along with their internal components, is illustrated in Fig. 3.2. The encoder begins with a BCS sampling module followed by several processing blocks that together perform image compression and introduce robustness against channel noise. Because the decoder generally does not have prior knowledge of the input statistics, the first step is to normalize the input image by dividing pixel values by the maximum intensity level, that is 255, which restricts the dynamic range to $[0, 1]$. The normalized image is then passed to the sampling layer, which extracts compressed measurements using a BCS procedure [148].

The BCS sampling module operates as described in [73]. The input image is partitioned into N_b non-overlapping blocks of size $B \times B \times C$, where B denotes the spatial block size and C denotes the number of color channels. Let $j \in \{1, \dots, N_b\}$ index the blocks. Each block is vectorized into a length- n_b vector with $n_b = CB^2$, denoted by $\mathbf{x}_j \in \mathbb{R}^{n_b}$. A compressive sampling stage acquires a reduced set of m_b measurements from each block, corresponding to a sampling

ratio m_b/n_b . This operation is represented by a sampling matrix $\Phi_B \in \mathbb{R}^{m_b \times n_b}$ that maps the vectorized block $\mathbf{x}_j$ to its measurement vector $\mathbf{y}_j \in \mathbb{R}^{m_b}$ via $\mathbf{y}_j = \Phi_B \mathbf{x}_j$. Each measurement is an inner product between $\mathbf{x}_j$ and one row of Φ_B . When a row is reshaped into a $B \times B \times C$ tensor, it can be interpreted as a convolutional kernel applied to the corresponding image block, which motivates implementing the sampling stage using a convolutional layer. To preserve the block structure and avoid overlap between adjacent blocks, the convolution is performed with stride B in both spatial dimensions, and no bias terms or activation functions are used. In this implementation, the convolutional layer uses m_b filters (output channels). For the full image, each filter produces one spatial feature map whose entries correspond to that measurement index across all blocks; equivalently, each block contributes an m_b -dimensional measurement vector. The sampling kernels (i.e., the entries of Φ_B) are learned jointly with the remaining encoder and decoder parameters in an end-to-end manner during training. As shown in Fig. 3.2, the CS sampling layer is implemented as a convolution whose number of output channels equals the number of measurements per block, $m_b = \left\lfloor \frac{m_b}{n_b} n_b \right\rfloor = \left\lfloor \frac{m_b}{n_b} CB^2 \right\rfloor$ so the m_b convolution entry in the compression block follows directly from the chosen sampling ratio m_b/n_b and the block size $B \times B \times C$.

After the sampling stage, the compressed data flows through a sequence of convolutional layers (stated as Conv. in Fig. 3.2), each followed by a Parametric Rectified Linear Unit (PReLU) activation and a normalization layer [6]. These convolutional layers extract discriminative features from the compressed representation to form the channel input vector. In this design, PReLU is selected over standard ReLU or LeakyReLU to prevent critical information loss within the highly constrained communication bottleneck. Standard ReLU maps all negative values to zero, which introduces a severe bottleneck that discards valuable structural and phase details prior to transmission. By contrast, PReLU retains these negative features by utilizing a learnable slope parameter. Unlike LeakyReLU, which relies on a fixed, hand-crafted negative slope, PReLU dynamically updates this parameter during end-to-end training. This allows the encoder to adaptively capture complex dependencies in the source data and fully utilize both the positive and negative quadrants of the continuous channel signal space.

Let $\tilde{\mathbf{z}} \in \mathbb{C}^k$ denote the unnormalized complex-valued encoder output (i.e., the output of the

last convolutional layer before enforcing the power constraint). The final convolutional output is normalized to satisfy the transmission power constraint according to

$$z = \sqrt{kP} \frac{\tilde{z}}{\sqrt{\tilde{z}^* \tilde{z}}}, \quad (3.7)$$

where $\tilde{z}^*$ is the complex conjugate transpose of $\tilde{z}$, ensuring that the channel input z satisfies the average transmitted power limit P . The output of this normalization layer is transmitted through the wireless channel by mapping the real and imaginary components of z to the in-phase and quadrature components of the modulated signal.

To enable end-to-end training, the communication channel is implemented as a non-trainable layer that applies the channel model in Eq. (3.5) to the transmitted symbols, thereby introducing random corruption corresponding to the wireless medium [6].

Decoder Design

The receiver employs a joint source-channel decoder that reconstructs the transmitted information from the noisy channel output. The decoder first applies a sequence of transposed convolutional layers (stated as Trans. Conv. in Fig. 3.2) with PReLU activations to invert the encoder operations and to produce an estimate of the original latent representation corresponding to each image block. Once the CS measurements have been recovered, the decoder proceeds to reconstruct the full image. The reconstruction process consists of two stages: an initial reconstruction network and a deep reconstruction network [148]. Similar to the sampling stage, the initial reconstruction is implemented via a convolutional layer with a kernel size and stride configured to match the block structure. Specifically, the initial reconstruction uses CB^2 convolutional filters of size $1 \times 1 \times m_b$ to reconstruct each block. A combination layer then aggregates the reconstructed vectors: for each block, the $1 \times 1 \times CB^2$ output vector is reshaped into a $B \times B \times C$ tensor, and the resulting blocks are placed back into their original spatial locations and concatenated to form the initial reconstructed image. Since no activation functions are used in this stage, the initial reconstruction acts as a linear recovery process.

The deep reconstruction stage refines this initial estimate by performing nonlinear enhancement. In the BCS-JSCC decoder, this is achieved by a deep reconstruction sub-network [148]

consisting of m layers. With the exception of the first and last layers, all layers share the same configuration, employing d filters of size $f \times f \times d$, where each filter processes a spatial region of size $f \times f$ across d feature maps. The first layer receives the initial reconstruction and therefore uses d filters of size $f \times f \times 3$. The final layer generates the reconstructed image using three filters of size $f \times f \times d$. In the experiments, the number of filters and filter size are set to $d = 64$ and $f = 3$, respectively. A ReLU activation follows each convolutional layer in the deep reconstruction network. ReLU is used here because the module operates after the initial linear reconstruction and mainly learns non-negative residual feature responses associated with edges, textures, and local artefact correction. Compared with the communication bottleneck in the encoder, preserving signed channel features is less critical at this stage, while ReLU provides a simple and stable nonlinearity with low computational cost. This nonlinear stage significantly enhances the visual quality by correcting distortions introduced during sampling and by the channel.

3.3.2 Loss Function

The proposed encoder and decoder networks are jointly optimized in an end-to-end manner. Given the input image, the goal is to obtain a highly compressed encoded representation with the encoder, and then recover the original image from its noisy version with the decoder network. Since the encoder, decoder and communication channel form an end-to-end network, they can be trained jointly. Following most of deep learning-based methods in this field, the MSE is adopted as the cost function of the proposed BCS-JSCC network. The optimization objective is represented as:

$$\min_{\theta, \varphi} \frac{1}{N} \sum_{i=1}^N \|g(\eta(f(\mathbf{x}_i; \theta)); \varphi) - \mathbf{x}_i\|_2^2 \quad (3.8)$$

where θ and φ denote the trainable parameters of the encoder $f(\cdot; \theta)$ and decoder $g(\cdot; \varphi)$ networks, respectively, and $g(\eta(f(\mathbf{x}_i; \theta)); \varphi)$ is the reconstructed output corresponding to input $\mathbf{x}_i$. Also, N represents the number of training samples in the dataset. It should be noted that the encoder and decoder are trained jointly end-to-end, but they form two separate modules at inference time (transmitter and receiver).

3.3.3 Results and Discussions

The proposed model is implemented in Tensorflow and optimized using the Adam algorithm. The bandwidth compression ratio k/n , defined as (2.11), is changed from 0.05 to 0.45. Also, the channel SNR, defined as:

$$SNR = 10 \log_{10} \frac{P}{\sigma^2} (dB) \quad (3.9)$$

is varied during different experiments. The performance of the algorithm is quantified in terms of PSNR of the reconstructed images as defined in (2.6).

Proposed BCS-JSCC architecture is trained on both CIFAR-10 [129] and ImageNet [130] datasets and the results are compared with state-of-the-art deep learning-based JSCC methods, namely Deep JSCC [6] and ADJSCC [51].

Evaluation on CIFAR-10 Dataset

The CIFAR-10 dataset, as explained in section 2.5 is used to assess the performance of the proposed BCS-JSCC framework under a variety of channel conditions. Evaluation is conducted on a separate set of 10,000 test images that are not used during training. The learning rate is initially set to 10^{-3} and reduced to 10^{-4} after 500,000 iterations. This iteration point was selected empirically by monitoring the training loss and validation PSNR: in the early stage, the model still benefits from a larger learning rate because the encoder, decoder, and BCS reconstruction modules are jointly adapting to the channel. After approximately 500,000 iterations, the loss reduction becomes slower and the validation PSNR begins to saturate, so the learning rate is reduced to refine the parameters more stably. Mini-batches of size 64 are employed, and training proceeds until no further improvement is observed in the validation performance.

In all CIFAR-10 experiments, the block size and number of color channels are set to $B = 8$ and $C = 3$, respectively. Also, the parameter c in the encoder's final convolutional layer determines the dimensionality of the channel input. Unless stated otherwise, a fixed sampling ratio of 0.25 is utilized. The test images are never used to tune hyperparameters, and each test sample is transmitted 10 times to average out the randomness introduced by the channel. All experiments are conducted under an AWGN model with varying SNR values.

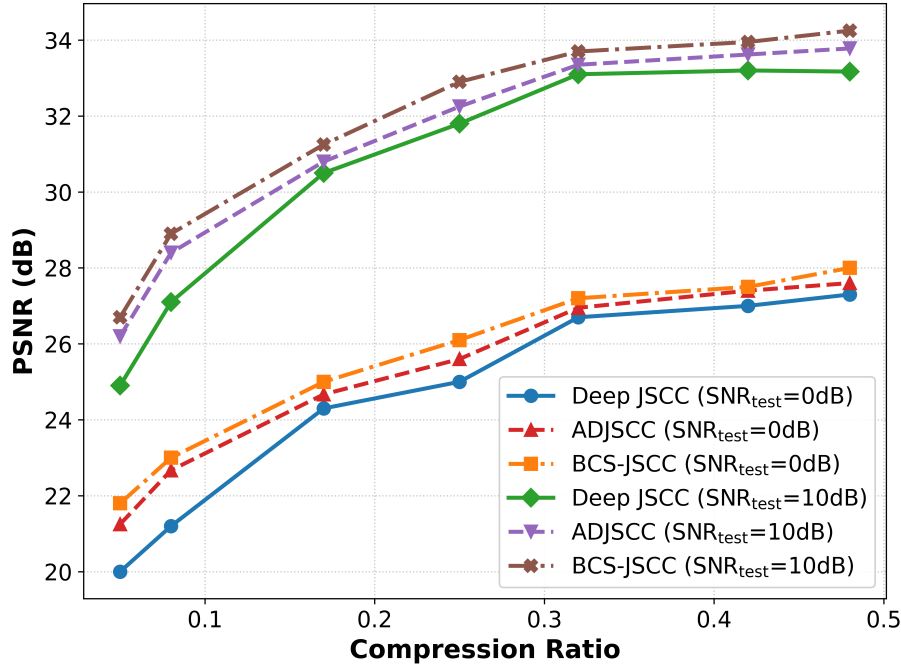


Figure 3.3: Performance of the proposed BCS-JSCC versus varying compression ratios over an AWGN channel on CIFAR-10 dataset

Fig. 3.3 shows the reconstruction PSNR as a function of the compression ratio for two representative channel conditions, $\text{SNR}_{\text{test}} \in \{0, 10\}$ dB, where each model is trained and evaluated at the same SNR. Several observations follow. First, BCS-JSCC consistently achieves the highest PSNR across the full range of compression ratios, indicating that integrating learned block-based sampling within the JSCC pipeline improves fidelity under bandwidth constraints. The gain is most visible at low compression ratios (i.e., the most bandwidth-limited regime), where all methods degrade but BCS-JSCC retains a clearer margin. Second, for the two SNR settings shown, the performance gap between BCS-JSCC and the baselines is significant at $\text{SNR}_{\text{test}} = 0$ dB, suggesting that the proposed compressive module is beneficial under harsh channel conditions as well. A probable explanation is that, in Deep JSCC and ADJSCC, additive channel noise perturbs a dense learned latent representation, and the nonlinear decoder can map these latent perturbations into structured pixel-domain artefacts, especially when the latent is highly compressed. In contrast, BCS-JSCC transmits an explicit set of learned linear measurements and performs a linear initial inversion followed by a refinement network, which provides a more stable reconstruction starting point and enables the decoder to suppress measurement-domain perturbations more effectively.

Fig. 3.4 further examines the behaviour of the proposed model under SNR mismatch for a fixed compression ratio of $1/6$. Each curve corresponds to a model trained at a particular SNR, denoted as SNR_{train} , and then tested across a wide range of channel conditions, represented by SNR_{test} . This setup highlights how well the learned model generalizes beyond the conditions seen during training. The results in Fig. 3.4 reveal several key findings:

- BCS-JSCC achieves higher PSNR than Deep JSCC in every tested scenario. The improvement is substantial at both low and high SNR values, indicating that the learned sampling and reconstruction mechanisms are effective at suppressing noise and preserving structural information.
- Similar to ADJSCC, the proposed method exhibits smooth and predictable performance degradation as SNR_{test} deviates from SNR_{train} . This behaviour indicates robustness to channel variations, which is critical for real-world wireless systems.
- When trained at a higher operating point, BCS-JSCC can also outperform ADJSCC in the moderate-to-high test SNR regime. In particular, the model trained at $SNR_{\text{train}} = 20$ dB attains a clear advantage over ADJSCC at higher SNR_{test} values. This suggests that the internal compressive representation remains effective in favourable channel conditions, i.e., as the channel becomes cleaner, the learned sampling-reconstruction pipeline continues to translate the improved reliability into higher PSNR rather than plateauing early.
- Deep JSCC exhibits pronounced saturation when trained at low SNR (e.g., $SNR_{\text{train}} = 4$ dB), where the curve flattens quickly and gains from improving SNR_{test} become limited. Training Deep JSCC at higher SNR (e.g., 13 or 20 dB) alleviates this effect, but its high- SNR_{test} performance still remains below the corresponding BCS-JSCC variants, which continue to improve and achieve the best PSNR.

In addition to PSNR, LPIPS is reported to provide a complementary perceptual-quality evaluation. While PSNR measures pixel-wise reconstruction fidelity, it may not fully reflect visual sharpness, texture consistency, and perceptual similarity. This limitation has also been highlighted in recent JSCC studies, where LPIPS is used to evaluate perceptual image quality more effectively [8], [146]. Fig. 3.5 shows the LPIPS performance of BCS-JSCC and the baseline

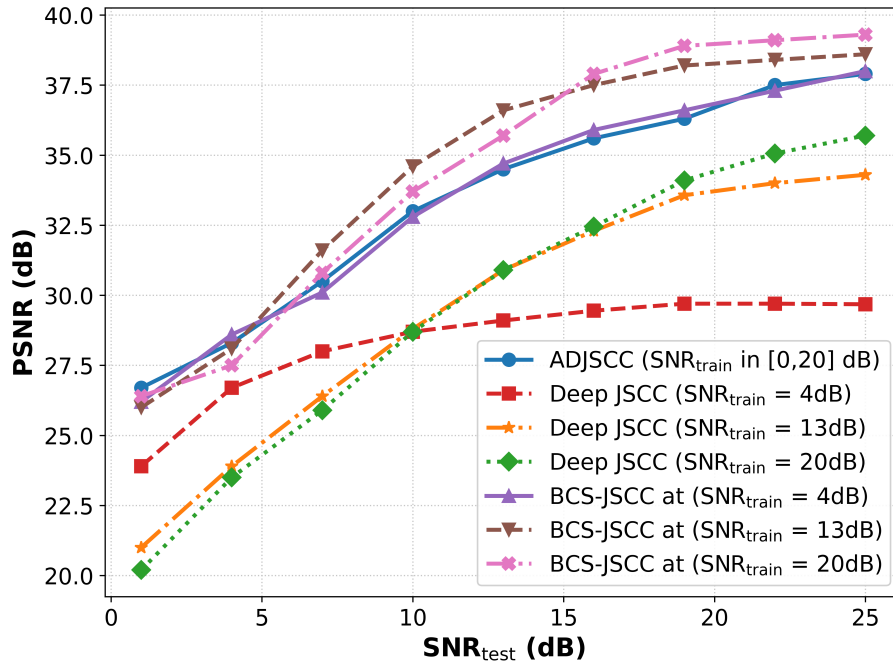


Figure 3.4: Performance of BCS-JSCC and baseline methods as PSNR versus channel SNRs over an AWGN channel on CIFAR-10 dataset (compression ratio=1/6)

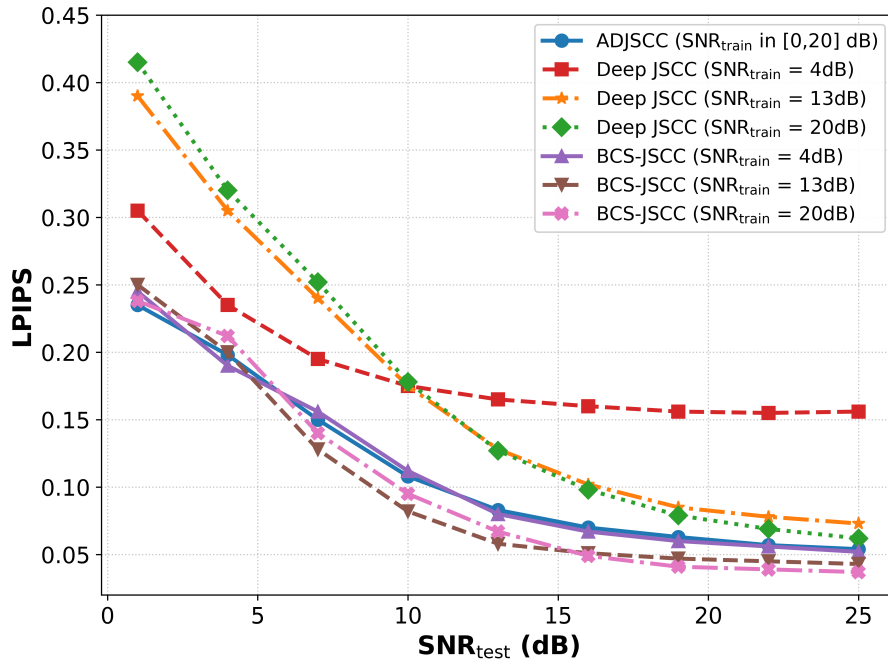


Figure 3.5: LPIPS comparison of BCS-JSCC and baseline methods over the AWGN channel methods under different test SNR values. Since lower LPIPS values indicate better perceptual similarity, the results show that BCS-JSCC achieves improved perceptual quality, especially in the medium-to-high SNR region where its PSNR advantage is also more evident.

Evaluation on Kodak Dataset

The proposed method is further evaluated on higher resolution images to demonstrate its effectiveness beyond low-resolution datasets. For this purpose, the architecture is trained on the ImageNet dataset [130], which is explained in section 2.5. During training, random crops of size 224×224 are extracted from the images and fed to the network in batches of 32 samples. In these experiments, the parameters are set to $B = 32$ and $C = 3$. A learning rate of 10^{-4} is used, and training proceeds until convergence. Unless stated otherwise, a fixed sampling ratio of 0.25 is utilized. For ImageNet, separate models are trained at different fixed SNR_{train} operating points, and each trained model is then evaluated across a range of SNR_{test} values to study mismatch behaviour.

For final performance evaluation, the Kodak dataset [131] is employed, as defined in section 2.5. Each Kodak image is transmitted 100 times to account for randomness in the channel, and the results are averaged across these trials. The evaluation is conducted under an AWGN channel model. Fig. 3.6 compares the PSNR of the proposed BCS-JSCC against Deep JSCC and ADJSCC for a compression ratio of $1/6$. The results clearly show that the proposed scheme surpasses both Deep JSCC and ADJSCC, capturing important visual structures in the compressed images and producing higher-quality reconstructions. The consistently superior PSNR across different SNR values highlights the method's robustness to channel noise and its ability to mitigate distortion. Overall, the technique achieves better image fidelity, even at increased compression levels and in noisy environments.

To further support the perceptual evaluation, Fig. 3.7 presents the LPIPS comparison for the Kodak dataset setting. The LPIPS metric measures the distance between original and reconstructed images in a learned deep-feature space, and therefore provides information that is not fully captured by PSNR or SSIM. The observed trend is consistent with the PSNR results: as the test SNR increases, the perceptual distance decreases for all methods. However, BCS-JSCC generally obtains lower LPIPS values than the baseline methods, indicating that the proposed compressive JSCC structure preserves perceptually relevant image content more effectively under the considered channel conditions.

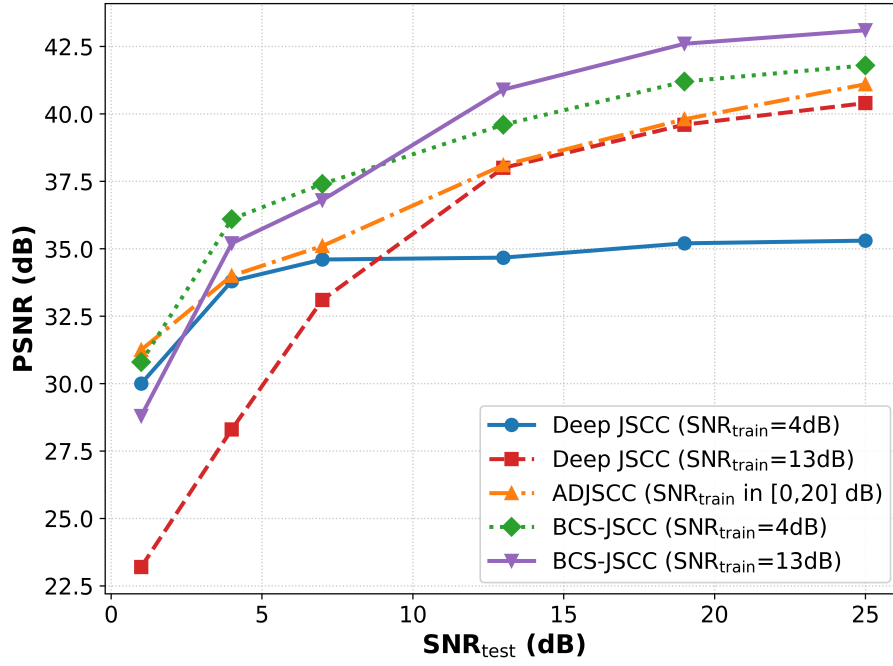


Figure 3.6: Performance of BCS-JSCC and baseline methods as PSNR versus channel SNRs over an AWGN channel on Kodak dataset (compression ratio=1/6)

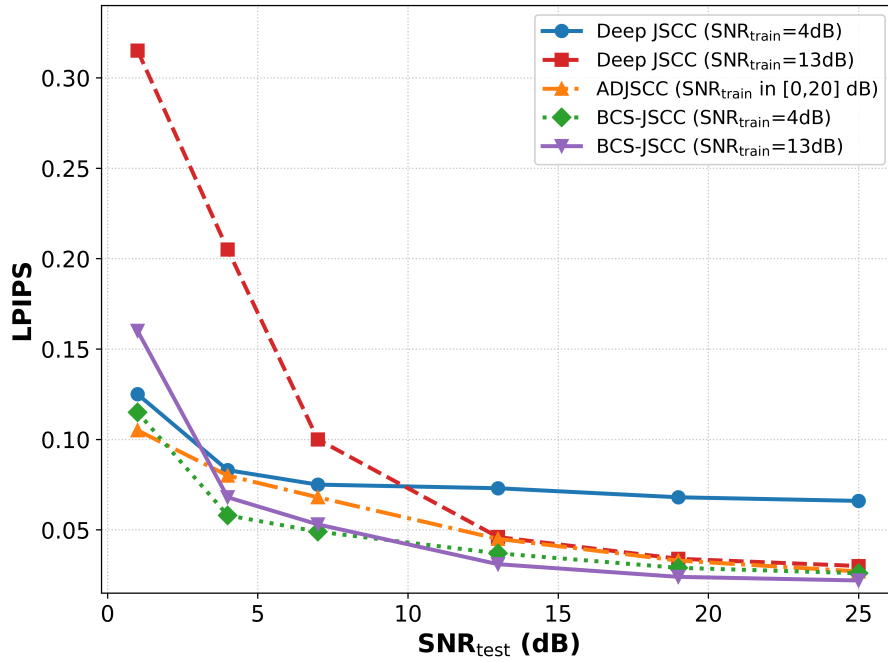


Figure 3.7: LPIPS comparison of BCS-JSCC and baseline methods under different test SNR values.

Training Loss Convergence

To further examine the optimization behaviour of BCS-JSCC, Fig. 3.8 shows the normalized training and validation MSE loss curves. The loss decreases rapidly during the early training

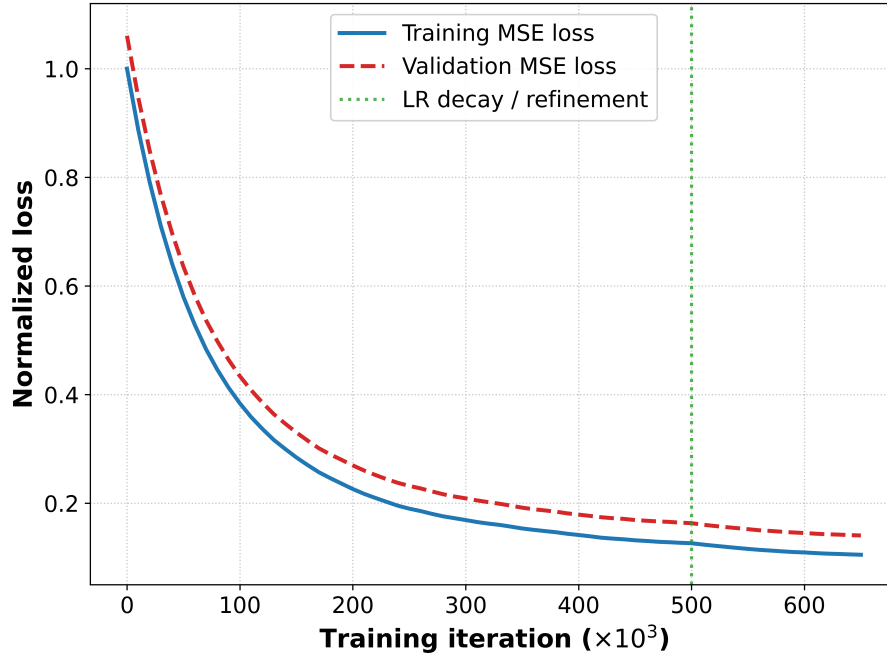


Figure 3.8: Normalized training and validation MSE loss convergence of BCS-JSCC.

stage and then gradually approaches a stable non-zero plateau. This indicates that the encoder, decoder, and BCS reconstruction modules are optimized stably under the considered channel setting. The learning-rate reduction at 500k iterations is used mainly for final refinement, since most of the loss reduction has already occurred before this point.

Visual Comparison

Fig. 3.9 presents a visual comparison of reconstructed images produced by Deep JSCC, ADJSCC, and the proposed BCS-JSCC model across three SNR conditions (2 dB, 10 dB, and 20 dB). All models are trained on the ImageNet dataset and evaluated over an AWGN channel. For each reconstruction, the corresponding PSNR and SSIM metrics are included, and each image contains a magnified region of interest to highlight differences in local detail. At 2 dB, Deep JSCC exhibits heavy noise across the entire scene. The reconstructed lighthouse, sky, and rock textures all suffer from pronounced graininess, and the enlarged patch reveals severe structural degradation. ADJSCC mitigates part of this noise, producing smoother regions, yet the restored textures remain coarse and lack consistency. In contrast, BCS-JSCC delivers a significantly cleaner reconstruction. The lighthouse edges appear sharper, colour transitions are smoother, and the rock surfaces maintain more natural structure. The zoomed-in patch shows that BCS-JSCC



Figure 3.9: Example of reconstructed image produced by the proposed BCS-JSCC and the baseline methods with compression ratio of 1/6, $\text{SNR}_{\text{train}}$ of 1 dB and SNR_{test} values of 2, 10 and 20 dB

suppresses channel noise much more effectively than the baseline methods. These qualitative differences are reflected in the notably higher PSNR and SSIM values.

At 10 dB, the performance gap becomes more evident. Deep JSCC still introduces noticeable noise, especially in the textured areas such as the rocks and water. ADJSCC provides improved stability but continues to produce blotchy patterns and loses fine-grained details. The BCS-JSCC output, however, captures both global structure and local detail with much higher fidelity. The lighthouse surface appears smoother, the roof textures are more consistent, and the overall contrast is better preserved. In the enlarged region, the difference in texture clarity is clear: BCS-JSCC maintains sharper boundaries and removes noise without oversmoothing. These results demonstrate that BCS-JSCC is more effective at extracting and reconstructing essential features even under moderate SNR conditions. At 20 dB, all methods improve, but BCS-JSCC continues to deliver the closest approximation to the original image. Deep JSCC reaches a performance plateau, still showing minor noise artefacts and local smoothing. ADJSCC achieves higher

SSIM and a cleaner appearance than Deep JSCC, but fine edges and textures remain slightly distorted. The BCS-JSCC result shows the highest perceptual quality, with well-preserved edges on the lighthouse, crisp texture on rocky surfaces, and accurate colour reproduction across the scene. The magnified region demonstrates that BCS-JSCC retains intricate structural details with minimal noise contamination. Its SSIM value is the highest in this comparison, confirming the superior preservation of perceptual similarity.

3.4 DCS-JSCC: Deep Compressive JSCC with Normalization and Refinement

In this section, an enhanced fixed-rate deep compressive JSCC framework, termed DCS-JSCC, is introduced. This model enhances the BCS-JSCC design by integrating Generalized Divisive Normalization (GDN) and Inverse GDN (IGDN) layers into the encoder and decoder, enabling more effective non-linear feature transformation under compressive representations. In addition, the deep reconstruction module is strengthened through a residual learning structure that facilitates deeper feature extraction and stabilizes training across noisy channel conditions. The loss function is also refined to provide stronger supervision for reconstruction, leading to improved image recovery at a fixed sampling rate. The following subsections describe the architecture of DCS-JSCC in detail and evaluate its performance against the baseline methods.

3.4.1 Model Architecture

Fig. 3.10 illustrates an end-to-end architecture of the encoder and decoder networks of the proposed DCS-JSCC. This model comprises a sampling network integrated with an encoding network to form the encoder, and a decoding network integrated with both the initial and deep reconstruction networks to form the decoder, as described below.

Encoder Design

The encoder shown in Fig. 3.10 comprises a BCS sampling network, followed by an array of processing blocks, which encode the image by further compressing it and adding resilience

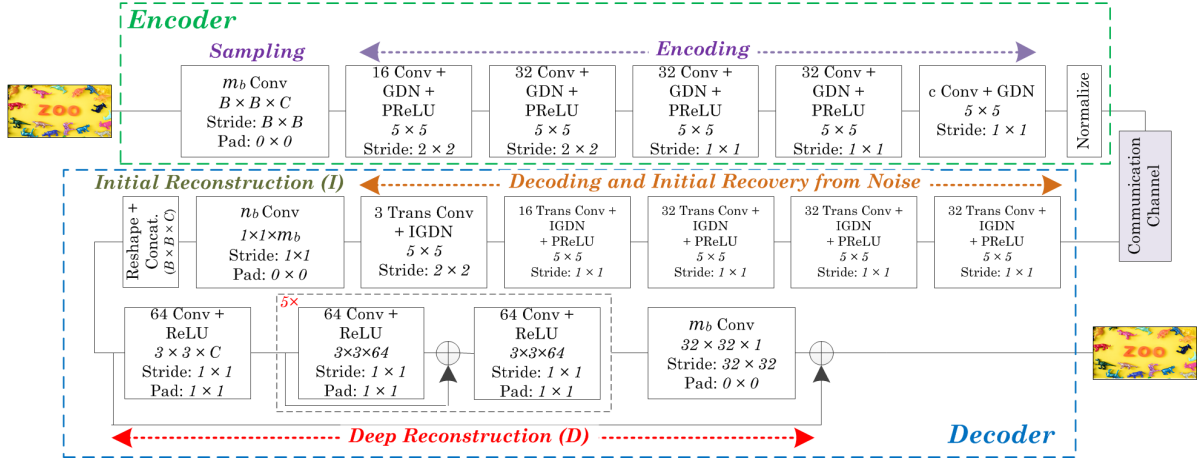


Figure 3.10: Architecture of the proposed DCS-JSCC framework

to noise. The CS sampling module inside the encoder is similar to BCS-JSCC, which is a convolutional layer with a stride of $B \times B$ and no biases or activation functions are used. Following the sampling network, the data flow proceeds through a series of convolutional layers, PReLU activation functions, and a GDN layer which constitute the encoding structure. These layers, except for the power normalization layer, are organized into five modules. Each of the first four modules comprises a convolution layer, a GDN layer, and a PReLU layer, while the fifth module consists of only a convolution layer and a GDN layer. Like BCS-JSCC, the final step within the encoder is the normalization process as in Eq. (3.7). After encoding, the joint source-channel coded sequence is transmitted over the communication channel which is included in the architecture as a non-trainable layer, represented by the transfer function in Eq. (3.5).

Using GDN in the encoder is beneficial as it is a non-linear transformation widely used in deep image compression networks [149]. Mathematically, let $v_i(m, n)$ represent the intermediate feature activation of the i -th channel at spatial coordinates (m, n) fed into the GDN layer from the preceding convolutional layer. The normalized output activation $\tilde{v}_i(m, n)$ generated by this transformation is explicitly calculated as:

$$\tilde{v}_i(m, n) = \frac{v_i(m, n)}{\sqrt{\beta_i + \sum_{j=1}^{C'} \gamma_{ij} (v_j(m, n))^2}} \quad (3.10)$$

where C' denotes the total number of channels in the current intermediate layer, β_i is a learned channel-specific bias, and γ_{ij} represents the learned cross-channel weighting parameters that

govern how the energy of channel j suppresses channel i . To guarantee mathematical stability within the end-to-end training pipeline and prevent division by zero or imaginary values in the denominator, both β_i and γ_{ij} are strictly constrained to be non-negative. This calculation performs a form of learned gain control inspired by biological visual systems and has been shown to approximate natural image statistics more effectively than conventional pointwise nonlinearities. Unlike pointwise activations such as ReLU or PReLU, GDN applies a channel-wise normalization via Eq. (3.10) that suppresses redundant or strongly correlated activations, producing a more decorrelated latent representation. In the context of compressive JSCC, this property is particularly beneficial after the compressive sampling stage, where the network must transform a highly compressed measurement vector into a compact and noise-resilient latent representation. By normalizing each feature in proportion to the aggregate energy of all surrounding features, GDN helps stabilize feature amplitudes and reduces the influence of localized high-frequency artefacts that are amplified by both compression and channel noise.

Decoder Design

As shown in Fig. 3.10 and similar to BCS-JSCC, the designed DCS-JSCC decoder has the following consecutive stages: decoding and initial recovery from noise, initial reconstruction and deep reconstruction. In the first stage, the decoder maps the symbols which are corrupted and compressed complex-valued ones to an estimation of the original channel input. In other words, it reverses the encoding stage operations of the encoder. This process involves passing the received corrupted coded inputs through a sequence of transposed convolutional layers with PReLU activation functions and IGDN. The inverse operation, IGDN, is used in the decoder to approximately invert the normalization applied in the encoder. The IGDN restores the amplitude profile of feature responses in a data-driven manner, enabling the decoder to reconstruct image structures more faithfully from noisy channel outputs. It should be noted that incorporating GDN and IGDN in DCS-JSCC enhances the network's ability to model statistical dependencies in the compressed measurements, leading to improved feature decorrelation, better noise resilience, and higher reconstruction quality relative to the BCS-JSCC baseline.

The reconstruction network comprises initial I and deep reconstruction networks D , ensuring

accurate recovery of the original image from the CS encoded measurements. For the initial reconstruction stage, similar to BCS-JSCC, a convolutional layer is employed with a specific kernel size and stride, as shown in Fig. 3.10. Subsequently, a combination layer is utilized, comprising a reshape function and a concatenation function, to obtain the initial reconstructed image. As there is no activation layer in the initial reconstruction network, it functions as a linear signal reconstruction network.

Following the initial reconstruction stage, the DCS-JSCC decoder employs an enhanced deep reconstruction network that significantly strengthens the non-linear recovery process compared with the design used in BCS-JSCC. While the BCS-JSCC architecture relies on a relatively shallow convolutional sub-network with uniform convolutional layers and ReLU activations, the reconstruction module in DCS-JSCC is substantially more expressive. It is built on a residual learning strategy that introduces deeper feature hierarchies, stronger non-linear modelling capacity, and improved gradient flow. The deep reconstruction pipeline consists of three major operations: feature extraction, non-linear mapping, and feature aggregation. The feature extraction block begins with a convolutional layer followed by a ReLU activation to convert the initial reconstructed output into a set of high-dimensional feature maps. These features form the input to the non-linear mapping stage, which constitutes the primary difference between DCS-JSCC and the earlier BCS-JSCC model. Instead of using identical convolutional layers repeated across the network, DCS-JSCC introduces a series of residual blocks, each containing multiple convolutional layers interleaved with non-linear activations and skip connections. This residual structure improves representational depth without degrading trainability and expands the effective receptive field, allowing the model to capture more complex spatial dependencies that arise from both compression and channel distortion. The final component of the deep reconstruction module is the feature aggregation operation, in which the high-dimensional features are collapsed through a final convolutional layer to synthesize the reconstructed image. A skip connection is introduced between the initial reconstruction and the output of the deep reconstruction network to accelerate convergence and to stabilize training. This skip pathway allows the network to learn residual refinement (i.e., an additive error term) rather than the full reconstruction, which is known to improve optimisation efficiency and reconstruction fidelity.

Compared with the deep reconstruction sub-network used in BCS-JSCC, which applies uniform convolutional layers with ReLU activations and no internal skip connections, the DCS-JSCC reconstruction network provides several advantages. It improves the ability of the decoder to correct structured artefacts that arise from block-based compressive sampling, it utilizes deeper non-linear transformations enabled by the residual architecture, and it exhibits enhanced robustness to channel noise due to better gradient propagation and more expressive feature modelling. These improvements lead to more accurate image recovery across a broad range of channel conditions.

Loss Function

The proposed encoder and decoder networks are optimized jointly in an end-to-end manner. Given an input image, the encoder $f(\cdot; \theta)$ produces the channel input $\mathbf{z}$, which is transmitted over the channel $\eta(\cdot)$. At the receiver, the decoder is composed of three modules: a channel decoding network $g(\cdot; \phi)$, an initial reconstruction module $I(\cdot; \psi)$, and a deep reconstruction module $D(\cdot; \chi)$. For convenience, the full decoder parameter set denoted as $\varphi \triangleq \{\phi, \psi, \chi\}$. The MSE is adopted as the cost function of the DCS-JSCC model, and two objectives are minimized, corresponding to the initial and final reconstructions.

The loss for the initial reconstruction is:

$$L_{\text{int}}(\theta, \phi, \psi) = \frac{1}{N} \sum_{i=1}^N \|I(g(\eta(f(\mathbf{x}_i; \theta))); \phi); \psi) - \mathbf{x}_i\|_2^2, \quad (3.11)$$

where N is the number of training samples. For the final reconstruction, the loss function is defined as

$$L_{\text{deep}}(\theta, \phi, \psi, \chi) = \frac{1}{N} \sum_{i=1}^N \|D(I(g(\eta(f(\mathbf{x}_i; \theta))); \phi); \psi); \chi) - \mathbf{x}_i\|_2^2. \quad (3.12)$$

Training is carried out by minimizing the sum of these two objectives with equal weights:

$$L_{\text{total}}(\theta, \phi, \psi, \chi) = L_{\text{int}}(\theta, \phi, \psi) + L_{\text{deep}}(\theta, \phi, \psi, \chi). \quad (3.13)$$

Compared with the loss formulation used in BCS-JSCC, the DCS-JSCC objective introduces an important refinement. In BCS-JSCC, training is performed using a single MSE loss computed

only on the final reconstructed image, which supervises the decoder output but provides no direct learning signal for intermediate representations. In contrast, the DCS-JSCC framework incorporates a two-stage loss that simultaneously optimizes both the initial reconstruction produced by the linear BCS module and the final reconstruction generated by the deep residual network. By introducing the intermediate loss term L_{int} , the encoder and decoder receive explicit supervision at an earlier stage of the pipeline, allowing the network to correct coarse structural errors before the non-linear refinement stage. This dual-objective formulation improves gradient flow, stabilizes training of the deeper DCS-JSCC architecture, and leads to more accurate reconstructions, particularly in challenging channel conditions. As a result, the DCS-JSCC model benefits from a stronger and more informative optimization signal than the single-stage MSE objective used in BCS-JSCC.

3.4.2 Results and Discussions

The proposed DCS-JSCC model is implemented using TensorFlow and optimized utilizing the Adam algorithm. The compression ratio is varied across experiments. Additionally, the channel SNR is adjusted during different trials. The performance of the algorithm is evaluated based on the PSNR of the reconstructed images. Similar to BCS-JSCC, the proposed DCS-JSCC architecture is trained on CIFAR-10 and ImageNet datasets. Then, the proposed model is tested on CIFAR-10 and Kodak datasets, and the results are compared with Deep JSCC [6] and ADJSCC [51].

Evaluation on CIFAR-10 Dataset

The evaluation settings for DCS-JSCC follow those used for BCS-JSCC. The method is tested on a separate set of 10,000 CIFAR-10 images that are completely distinct from the training samples. The training process begins with a learning rate of 10^{-3} , which is decreased to 10^{-4} after 500,000 iterations. The same schedule as BCS-JSCC is used to ensure a fair comparison and to isolate the effect of the architectural changes introduced in DCS-JSCC. The value of 500,000 iterations was chosen because the training loss and validation PSNR showed rapid improvement before this point and then entered a slower refinement stage. Reducing the learning

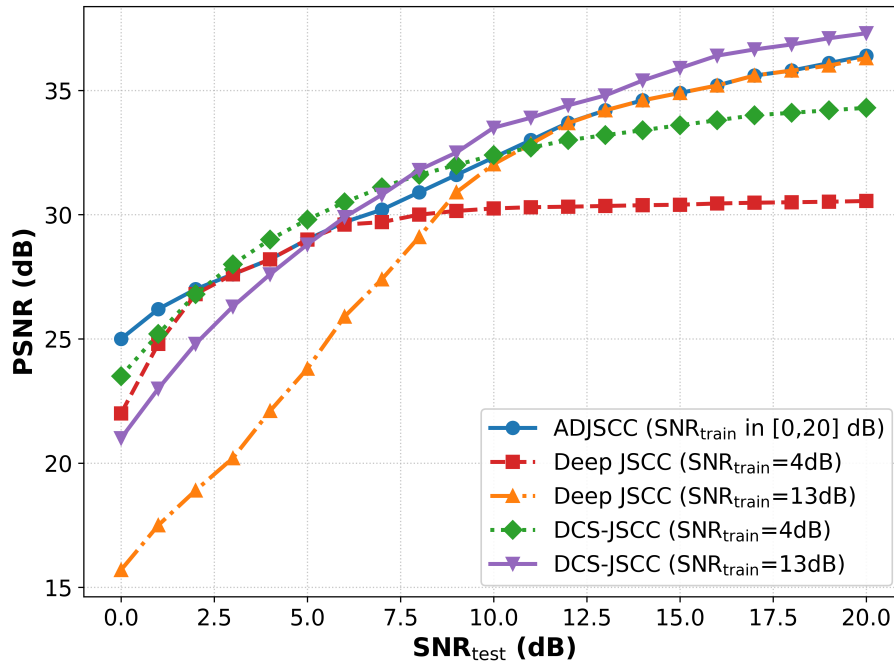


Figure 3.11: Performance of DCS-JSCC and baseline methods as PSNR versus channel SNRs over an AWGN channel on CIFAR-10 dataset (compression ratio=1/6)

rate at this stage improves convergence stability, especially because DCS-JSCC contains a deeper residual reconstruction module and a two-stage loss. Mini-batches of size 64 are employed, and training proceeds until no additional improvement is observed on the validation dataset.

For the CIFAR-10 experiments, the block size is set to $B = 8$, and the parameter c in the encoder's final convolutional layer determines the dimensionality of the channel input. It is also important to emphasize that the test images are never used for hyperparameter tuning. Fig. 3.11 presents a detailed comparison of PSNR performance across a wide range of channel SNR values for the proposed DCS-JSCC model, Deep JSCC trained at two fixed SNR points (4 dB and 13 dB), and the ADJSCC baseline trained over a SNR interval of [0,20] dB. All models operate at the same compression ratio of 1/6 to ensure a fair comparison. Each curve corresponds to a model trained at a specific $\text{SNR}_{\text{train}}$, and the performance is evaluated over multiple SNR_{test} values to examine robustness under mismatched channel conditions.

The Fig. 3.11 which is the results of PSNR vs. SNR_{test} , reveals following key behaviors:

- First, DCS-JSCC demonstrates clear improvements over Deep JSCC for both training SNR settings. When trained at 4 dB, DCS-JSCC consistently achieves higher PSNR across the entire testing range, reflecting its enhanced reconstruction capability and stronger

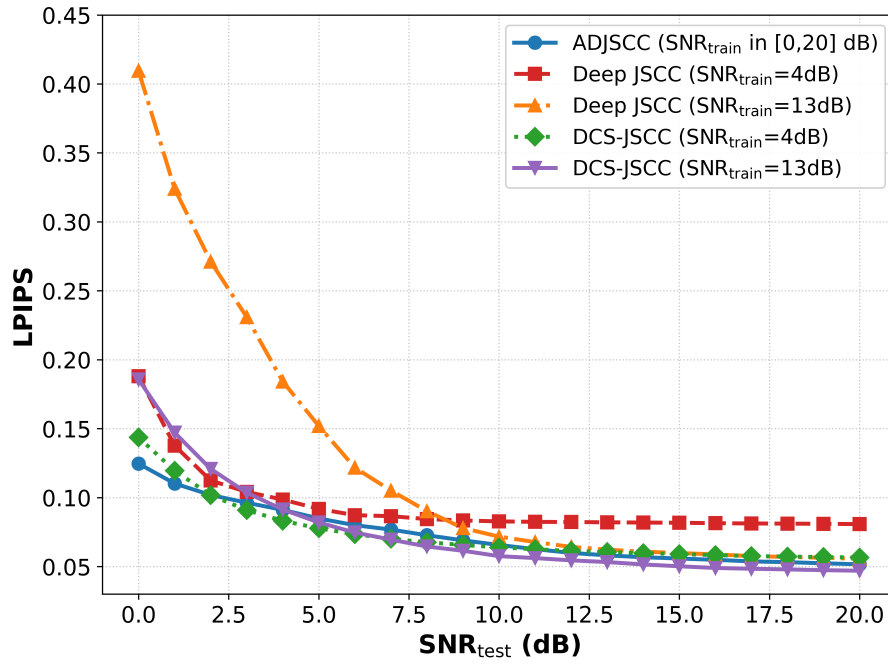


Figure 3.12: LPIPS comparison of DCS-JSCC and baseline methods over the AWGN channel on the CIFAR-10 dataset.

resilience to noise. When trained at 13 dB, DCS-JSCC continues to outperform Deep JSCC, particularly in the medium to low SNR_{test} region, where the gap becomes more noticeable.

- Second, comparing DCS-JSCC with ADJSCC highlights the advantages of the proposed architecture. Although ADJSCC is trained over a broad SNR range and thus provides relatively stable performance across different channel conditions, DCS-JSCC surpasses it in almost all regions. The improvement becomes particularly noticeable for $\text{SNR}_{\text{test}} \geq 8\text{dB}$, where DCS-JSCC (trained at both 4 dB and 13 dB) yields significantly higher PSNR. For instance, the model trained at 13 dB provides up to 1.2 dB improvement over ADJSCC at higher SNR levels.
- The curves also illustrate that DCS-JSCC features smooth performance transitions as SNR_{test} varies, confirming the inherent generalization ability of these architectures under mismatched channel conditions.

So, it can be stated that the results validate that the proposed DCS-JSCC framework not only achieves superior reconstruction quality compared with the Deep JSCC baseline but also exhibits

stronger robustness and adaptability relative to ADJSCC across the full operational SNR range.

In addition to PSNR, LPIPS is included as a complementary perceptual-quality metric for the CIFAR-10 evaluation. PSNR measures pixel-wise reconstruction fidelity, but it does not fully describe perceptual sharpness, texture consistency, or visual naturalness. Recent JSCC studies have therefore used LPIPS to evaluate perceptual similarity in the feature space of a pretrained neural network. Fig. 3.12 shows the LPIPS comparison of DCS-JSCC and the baseline methods over the AWGN channel. The LPIPS values decrease as the test SNR improves, which is consistent with the PSNR trend. DCS-JSCC generally obtains lower LPIPS than the corresponding Deep JSCC baselines, indicating that the improved normalization and residual reconstruction modules help preserve perceptually relevant image content more effectively.

Evaluation on Kodak Dataset

The proposed DCS-JSCC is further evaluated on higher resolution images by training it on the ImageNet dataset [130]. During training, each image is randomly cropped to produce patches of size 224×224 , which are then fed into the network in batches of 32 samples. For these experiments, the block size is set to $B = 32$. Training is performed with a fixed learning rate of 10^{-4} and continues until the model converges. Also, separate models are trained at different fixed SNR_{train} operating points, and each trained model is then evaluated across a range of SNR_{test} values to study mismatch behaviour. The dataset is partitioned into training and validation sets using a 9 : 1 split. For final performance evaluation, the Kodak dataset [131] is used. Each test image is transmitted 100 times through independently generated channel realizations, and performance is averaged to account for channel randomness. All evaluations are carried out under an AWGN channel model.

Fig. 3.13 presents the PSNR performance on the Kodak dataset for a compression ratio of $1/6$, comparing DCS-JSCC against Deep JSCC and ADJSCC. The results show that DCS-JSCC achieves the highest PSNR for most tested SNR_{test} values, and it consistently outperforms the baselines; at $SNR_{\text{test}} = 0$ dB, ADJSCC attains a slightly higher PSNR than the DCS-JSCC models. This improvement highlights the effectiveness of the learned compressive sampling and enhanced reconstruction architecture, which allows the model to preserve fine image structures

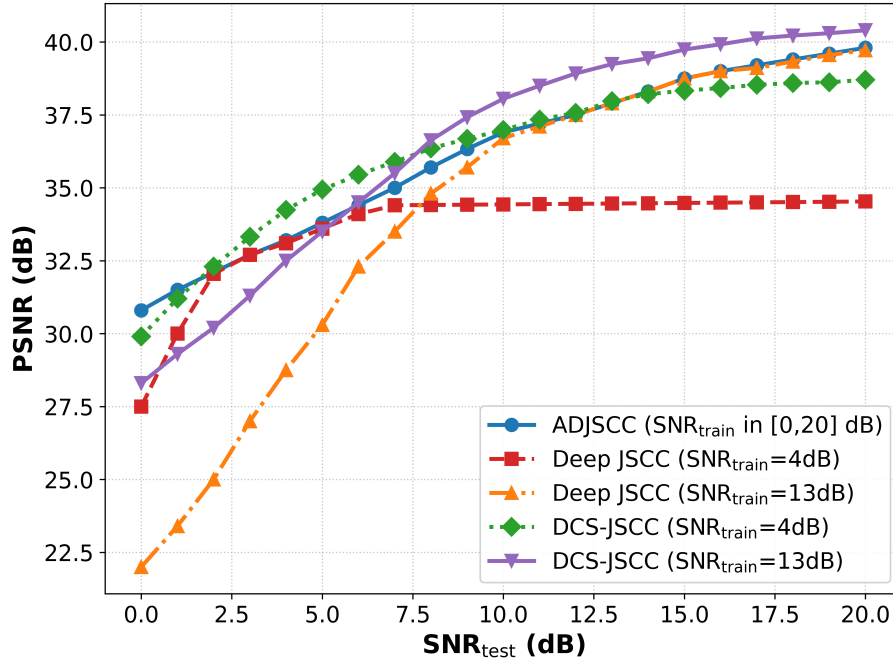


Figure 3.13: Performance of DCS-JSCC and baseline methods as PSNR versus channel SNRs over an AWGN channel on Kodak dataset (compression ratio=1/6)

more effectively than the baseline approaches. The trends observed in Fig. 3.13 can be interpreted as follows. At low SNR_{test} values, all methods experience noticeable degradation, yet DCS-JSCC maintains a clear advantage by stabilizing reconstruction quality and mitigating noise-induced distortions. In the mid-SNR regime (approximately 5 to 12 dB), DCS-JSCC exhibits a steeper performance increase compared with both Deep JSCC and ADJSCC, reflecting a more efficient use of channel information and stronger robustness to mismatched SNR conditions. At high SNR levels, DCS-JSCC achieves the highest PSNR at the upper end of the tested range, outperforming ADJSCC by a significant margin and substantially surpassing the Deep JSCC model trained at 4 dB. This suggests that the combination of GDN-based normalization, improved nonlinear reconstruction, and enhanced feature extraction enables more faithful recovery of high-frequency details.

The perceptual evaluation is further extended to the Kodak dataset, as shown in Fig. 3.14. Compared with CIFAR-10, Kodak contains higher-resolution images with richer textures and local structures, making perceptual evaluation particularly useful. The results show that LPIPS decreases with increasing test SNR for all methods, while methods that saturate in PSNR also show a slower reduction in LPIPS. DCS-JSCC achieves lower LPIPS than Deep JSCC across most SNR values and remains competitive with or better than ADJSCC, especially when trained

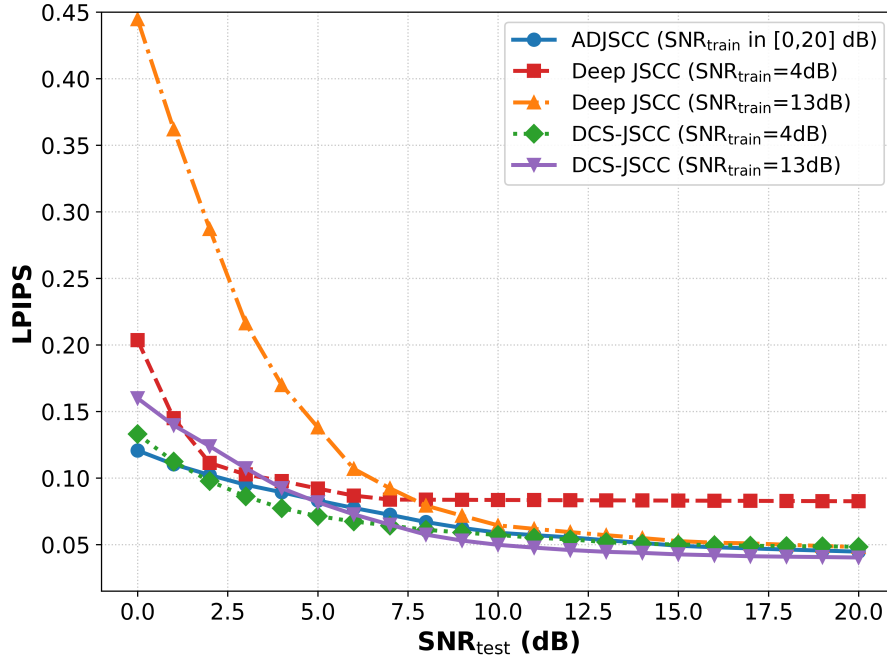


Figure 3.14: LPIPS comparison of DCS-JSCC and baseline methods over the AWGN channel on the Kodak dataset. Lower LPIPS indicates better perceptual image quality.

at the higher SNR operating point. This confirms that the proposed residual refinement and GDN/IGDN-based feature normalization improve not only pixel-level reconstruction fidelity but also perceptual similarity.

Ablation Study on DCS-JSCC Structural Components

To isolate and quantify the individual performance contributions of the architectural modifications introduced in DCS-JSCC over the BCS-JSCC framework, an ablation study is conducted. The evaluation was performed on the Kodak dataset at a bandwidth compression ratio of $R = 1/6$ under two distinct channel noise conditions: a harsh regime ($\text{SNR}_{\text{test}} = 4$ dB) and a moderate regime ($\text{SNR}_{\text{test}} = 13$ dB). All variants were trained at a fixed operating point of $\text{SNR}_{\text{train}} = 4$ dB to ensure a mathematically consistent baseline. The baseline model (Configuration 1) corresponds to the original BCS-JSCC configuration, which utilizes a shallow, uniform convolutional reconstruction network, a single-stage final MSE loss function, and standard PReLU activations without advanced latent normalization. The framework is progressively built up through the following configurations:

- Configuration 2 (+ Residual Blocks): The shallow uniform reconstruction network is

Table 3.1: Ablation study isolating the performance impact of various configurations in DCS-JSCC (Kodak dataset, $R = 1/6$, $\text{SNR}_{\text{train}} = 4$ dB).

Config. #	Residual Blocks	Two-Stage Loss	GDN / IGDN	$\text{SNR}_{\text{test}} = 4$ dB		$\text{SNR}_{\text{test}} = 13$ dB	
				PSNR (dB)	SSIM	PSNR (dB)	SSIM
1	×	×	×	28.10	0.762	32.85	0.854
2	✓	×	×	28.95	0.791	33.60	0.871
3	✓	✓	×	29.40	0.808	34.15	0.885
4	✓	✓	✓	30.15	0.831	34.90	0.902

replaced with the deeper residual CNN architecture featuring internal skip connections, while keeping the single-stage loss and omitting GDN/IGDN layers.

- Configuration 3 (+ Two-Stage Loss): Building on Configuration 2, the intermediate loss function L_{int} is introduced to provide direct joint supervision on the coarse output of the initial linear reconstruction block.
- Configuration 4 (Full DCS-JSCC): The final architecture of DCS-JSCC as described in subsection 3.4.1.

The quantitative impacts of these structural layers on the reconstruction PSNR and SSIM are summarized in Table 3.1. As observed from the results in Table 3.1, upgrading to a deeper residual mapping strategy (Configuration 2) provides an immediate performance increment of +0.85 dB and +0.75 dB at low and moderate SNRs, respectively. This demonstrates the benefit of optimization via residual error tracks, which allows the decoder to correct spatial block artifacts more accurately without gradient degradation. The introduction of intermediate supervision via the dual-loss configuration (Configuration 3) provides an additional gain of approximately +0.45 dB to +0.55 dB. By forcing the network to optimize the linear initialization explicitly, the encoder learns a cleaner, more reliable intermediate projection space, which stabilizes optimization pathways across the subsequent nonlinear layers. Finally, the inclusion of GDN/IGDN layers (Configuration 4, representing the complete DCS-JSCC model) yields the largest single-step improvement, boosting the low-SNR metrics by +0.75 dB and pushing the high-SNR SSIM past the 0.90 threshold. This underscores that biological-style divisive gain control is highly effective for compressive communication tasks; by dampening highly correlated channel-wise feature components, it yields an intrinsically noise-resilient latent representation that preserves delicate

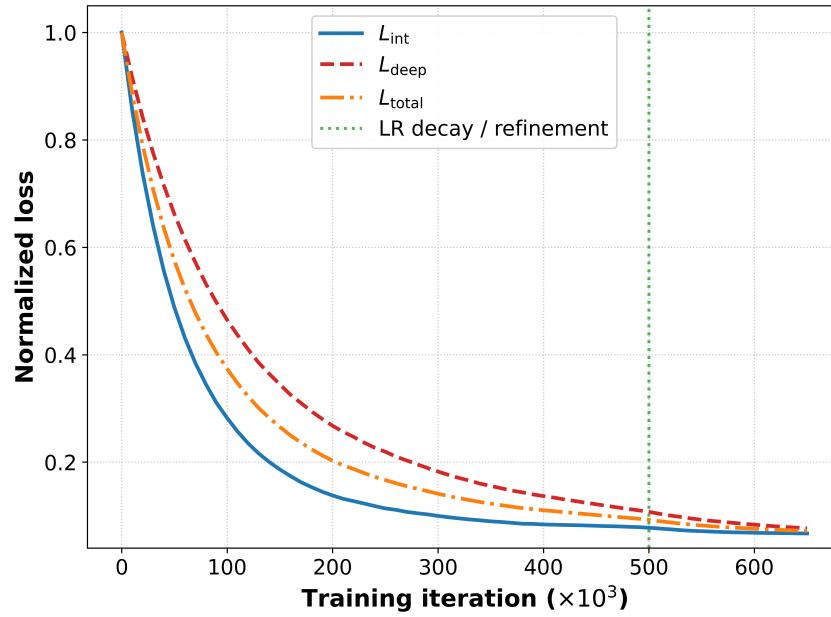


Figure 3.15: Normalized convergence of L_{int} , L_{deep} , and L_{total} in DCS-JSCC.

structural content even over impaired channels.

Training Loss Convergence

To verify the convergence behaviour of DCS-JSCC, Fig. 3.15 presents normalized curves for the intermediate reconstruction loss L_{int} , the deep reconstruction loss L_{deep} , and the total loss L_{total} . The intermediate loss decreases faster because the initial reconstruction module learns a coarse inverse mapping from the recovered measurements. The deep reconstruction loss decreases more gradually because the residual refinement network learns more detailed nonlinear corrections. The total loss follows a stable decreasing trend and approaches a non-zero plateau before the learning-rate reduction, while the reduced learning rate provides a final refinement stage.

Visual Comparison

Fig. 3.16 provides a qualitative assessment of reconstructed images obtained by Deep JSCC, ADJSCC, and the proposed DCS-JSCC across three different SNR levels (2 dB, 10 dB, and 20 dB). For each method, the results include both the reconstructed full image and a magnified region of interest, accompanied by the corresponding PSNR and SSIM values. These visual results clearly highlight the improvements achieved by DCS-JSCC in terms of noise suppression,

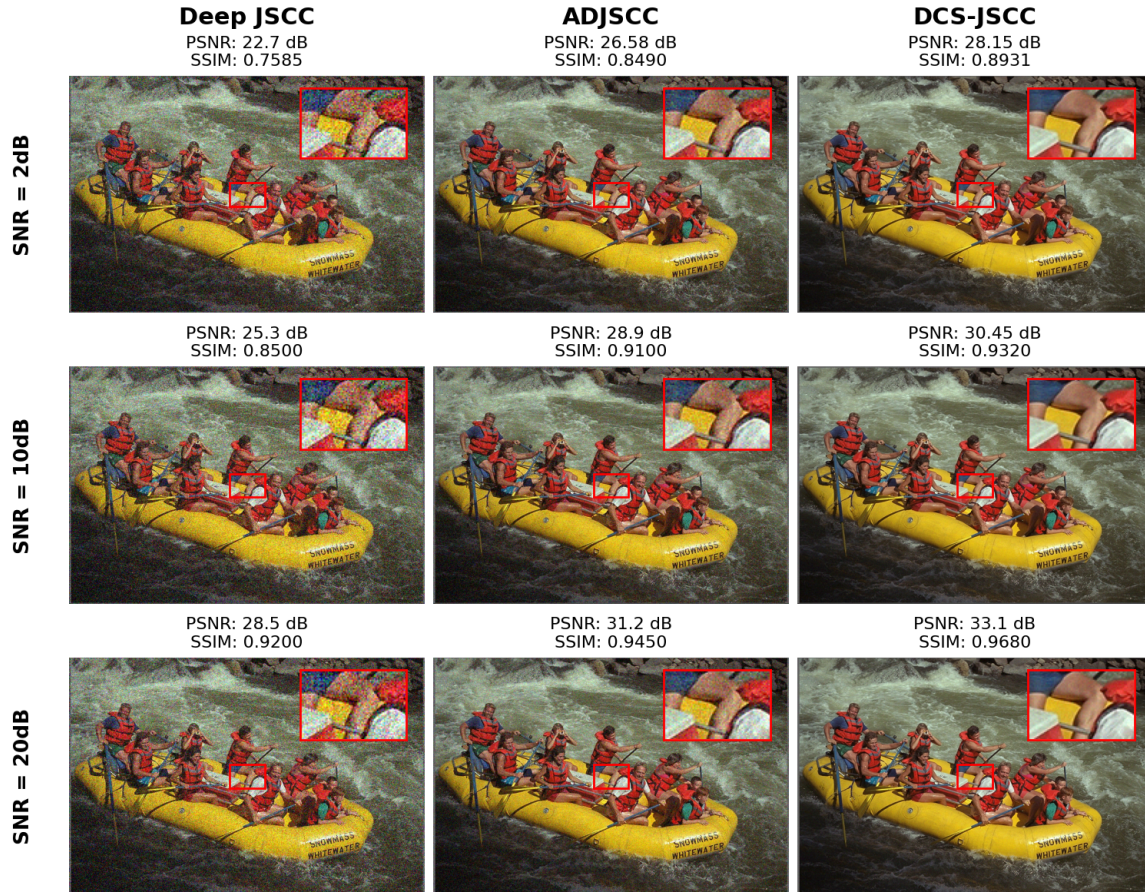


Figure 3.16: Example of reconstructed image produced by the proposed DCS-JSCC and the baseline methods with compression ratio of 1/6, $\text{SNR}_{\text{train}}$ of 1 dB and SNR_{test} values of 2, 10 and 20 dB

detail preservation, and overall perceptual quality. At the lowest SNR level of 2 dB, Deep JSCC exhibits substantial noise artefacts across the entire image, with severe graininess that obscures structural details. ADJSCC provides a noticeable improvement by reducing rough noise patterns, yet edges and textures remain distorted and lack consistency. In contrast, DCS-JSCC produces significantly cleaner reconstructions, preserving object boundaries and color transitions far more effectively. The zoomed-in regions confirm this difference: while Deep JSCC and ADJSCC still display speckled noise and texture degradation, DCS-JSCC recovers smoother surfaces and sharper contours, reflected in its higher PSNR and SSIM scores.

At moderate SNR (10 dB), all methods improve, but the gaps between them remain evident. Deep JSCC still suffers from visible noise, particularly in textured areas such as the water and raft surface. ADJSCC produces a more stable reconstruction, yet fine-grained textures appear smeared or blotchy in the enlarged region. DCS-JSCC clearly outperforms both baselines by

restoring more accurate textures and reducing channel-induced artefacts. The magnified patches show that DCS-JSCC maintains cleaner edges, smoother tonal transitions, and more faithful color representation. This aligns with the quantitative jump in PSNR and SSIM observed for DCS-JSCC in this SNR regime. At 20 dB, the performance of all methods improves further, but meaningful differences persist. Deep JSCC exhibits diminishing returns and continues to show visible smoothing and residual noise. ADJSCC generates a much clearer reconstruction but still fails to recover high-frequency details present in the original image. The reconstruction from DCS-JSCC is noticeably closer to the ground truth, demonstrating consistent restoration of fine textures and maintaining structural integrity in challenging regions such as clothing folds and facial contours. This is reflected in the significantly higher SSIM value for DCS-JSCC, which indicates better preservation of perceptual similarity.

3.5 CS-JSCC: Advanced Deep Compressive JSCC with Saliency Aware Sampling

In this section, another variant of the proposed fixed-rate deep compressive JSCC framework, referred to as CS-JSCC, is introduced. This model advances the JSCC design by incorporating a multi-channel BCS module that captures richer structural information than the single-channel BCS sampling used in BCS-JSCC and DCS-JSCC. The multi-channel sampling process enables CS-JSCC to probe the image with several complementary measurement operators, resulting in more expressive and robust compressed representations under strict bandwidth constraints. Beyond the enhanced sampling stage, CS-JSCC employs more sophisticated initial and deep reconstruction networks, combining improved block reconstruction with a powerful iterative refinement module inspired by unrolled denoising-based CS recovery. Furthermore, the loss function and training strategy are expanded to jointly supervise multi-channel sampling, initial reconstruction accuracy, and deep refinement quality, providing stronger end-to-end guidance than those used in BCS-JSCC and the two-loss structure of DCS-JSCC. The subsequent subsections detail the architecture of CS-JSCC and present its performance relative to baseline JSCC techniques.

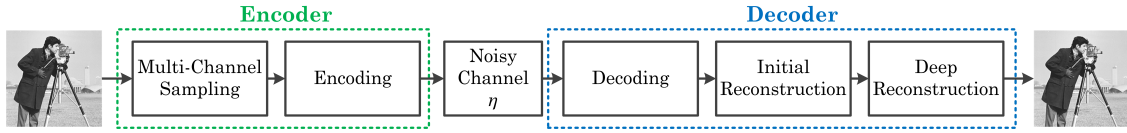


Figure 3.17: Architecture of the proposed CS-JSCC framework

3.5.1 Model Architecture

As illustrated in Fig. 3.17, the CS-JSCC framework comprises a joint source-channel encoder and decoder, forming a fully end-to-end trainable model. At the encoder side, a multi-channel BCS sampling module captures compressed measurements, followed by a sequence of layers that jointly perform image compression and enhance robustness to channel impairments. At the decoder side, the noisy received signal is first processed by a decoding module that estimates the channel input, after which an image reconstruction module restores and refines the recovered image blocks.

Encoder Design

As illustrated in Fig. 3.18, the CS-JSCC encoder is a deep neural network that jointly performs image compression and channel encoding in an end-to-end manner for robust wireless image transmission. The encoding process begins with an adaptive sampling stage, where each image block is sampled at one of several predefined rates. To achieve this, the multi-channel CS sampling network f_{samp} [150] is integrated into CS-JSCC model. This sampling module acts as a learnable sampling front end that captures essential visual information while adapting compression strength to local image content. Unlike state-of-the-art deep JSCC methods that apply a uniform fixed compression rate across the entire image [6], [25], [45], [51], [61], the multi-channel BCS framework enables content-adaptive block-wise sampling, allowing the encoder to allocate higher sampling rates to visually salient regions (e.g., edges and textures) and lower rates to smoother areas. For a given global target sampling rate SR , this mechanism effectively distributes the sampling budget across the image blocks according to perceptual importance, thereby enhancing overall transmission efficiency and reconstruction quality under varying SNR and bandwidth conditions.

The sampling network f_{samp} comprises s parallel channels, each corresponding to a distinct

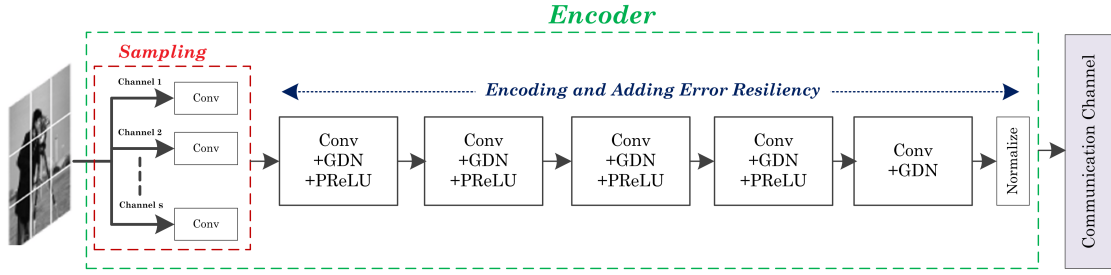


Figure 3.18: Proposed architecture for CS-JSCC encoder

sampling rate from a set $\mathcal{S}$ of predefined sampling rates. The sampling process operates on non-overlapping image blocks of size $B \times B$. The visual saliency of each block is first computed as described in [151]. The saliency values are then normalized to obtain block-specific saliency weights p_i , so that the average saliency weight across all blocks in an image is approximately 1. This ensures that the global sampling target rate SR is met on average. The saliency weight p_i indicates the relative importance of the i -th block within the image. Given a target sampling rate SR for the entire image, the continuous sampling rate s_i of the i -th block is determined as:

$$s_i = SR \cdot p_i \quad (3.14)$$

The continuous sampling rate s_i of the i -th image block is then quantized down to the nearest predefined sampling rate $\hat{s}_i \in \mathcal{S}$ supported by the model. For example, if the target global image sampling rate is $SR = 0.1$ and the set of predefined sampling rates is $\mathcal{S} = \{0.01, 0.03, 0.05, 0.10, 0.20, 0.30, 0.40\}$, a highly salient block with normalized saliency weight of $p_i = 1.9$, will be sampled at rate $\hat{s}_i = 0.1$, since $s_i = 0.1 \times 1.9 = 0.19$ according to Eq. (3.14). Conversely, for a smoother background block with $p_i = 0.4$, the resulting continuous sampling rate is $s_i = 0.1 \times 0.4 = 0.04$ which will be quantized to 0.03. This mapping ensures that each block is routed to one of the corresponding multi-channel sampling paths during training, enabling the model to learn adaptive, block-wise compression in an end-to-end manner, while not surpassing the target global image sampling rate SR . Fig. 3.19 provides a visual example of this process. As shown in this figure, blocks containing edges and textured regions obtain larger values of p_i and are therefore assigned higher sampling rates, whereas smooth background regions are mapped to lower sampling-rate channels. In this example, the image contains 196

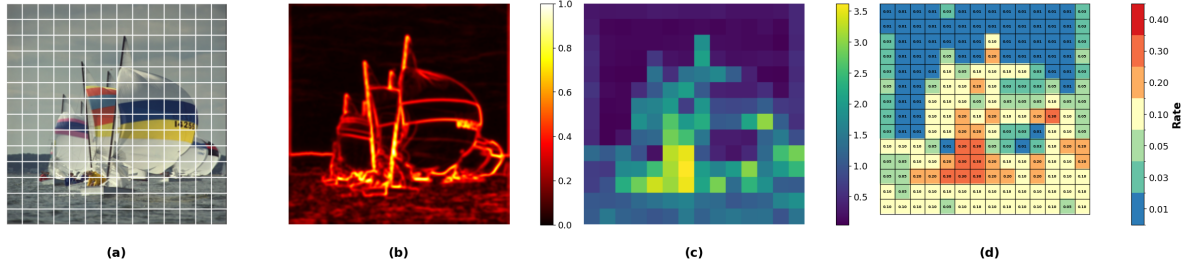


Figure 3.19: Visualization of the saliency-based sampling rate assignment mechanism in CS-JSCC: (a) block partitioning, (b) saliency map, (c) normalized block weights, and (d) corresponding block-wise sampling rate assignment.

blocks, with 63 blocks assigned to rate 0.01, 19 blocks to rate 0.03, 24 blocks to rate 0.05, 64 blocks to rate 0.10, 18 blocks to rate 0.20, and 8 blocks to rate 0.30. This confirms that the multi-channel sampling module distributes the sampling budget according to local visual importance.

Assuming j is the index of the sampling channel associated with the sampling rate $\hat{s}_i$, the image block $\mathbf{x}_i$ is routed to the j -th channel, which performs sampling via a convolutional layer without bias or activation:

$$\mathbf{y}_i = f_{\text{samp}}(\mathbf{x}_i, \mathbf{w}_j^{\text{samp}}; \boldsymbol{\theta}_{\text{samp}}) \quad (3.15)$$

where $\mathbf{w}_j^{\text{samp}}$ denotes m_j convolutional kernels of size $B \times B \times 1$ for the j -th channel, $\boldsymbol{\theta}_{\text{samp}}$ is the set of the multi-channel sampling network parameters, i.e., the set of s convolutional kernels $\{\mathbf{w}_j^{\text{samp}}\}_{j=1}^s$, and $\mathbf{y}_i$ is the vector of acquired measurements for the i -th image block. This operation is equivalent to applying a learned measurement matrix Φ_j in the CS formulation, with m_j corresponding to the number of acquired measurements. The selected channel index $j \in \{1, 2, \dots, s\}$ is transmitted to the receiver as side information, requiring only $\lceil \log_2 s \rceil$ bits per block. Since an incorrectly decoded index would cause the receiver to use the wrong reconstruction channel for the corresponding block, we assume that this side information is transmitted through a strongly protected control stream. In the proposed setting with $s = 7$, each block requires 3 index bits. Using a short high-redundancy forward error correction (FEC) code, this side information can be transmitted with practically error-free reliability. Therefore, the receiver is assumed to have reliable access to the correct block-wise channel indices.

All convolutional kernels across channels are jointly optimized with the encoder and decoder

during training, enabling the network to learn specialized sampling filters tailored to different compression levels. The number of channels s controls the granularity of adaptivity: larger s values offer finer control but increase model complexity and memory cost, while smaller s reduce flexibility. Although multiple kernel sets $\{w_j^{\text{samp}}\}_{j=1}^s$ must be stored, the memory footprint remains moderate and can be further reduced using model-compression techniques such as kernel pruning or low-rank factorization [152]. In summary, by utilizing the multi-channel BCS sampling framework within the CS-JSCC encoder, we enable content-aware, dynamically adaptive image compression that effectively balances compression efficiency, perceptual quality, and channel robustness across diverse transmission conditions.

Following the multi-channel sampling stage, the data is processed through a series of convolutional layers with PReLU activations and GDN layers, collectively forming the encoding structure $f_{\text{enc}}(\cdot; \theta_{\text{enc}})$. Except for the final power normalization stage, the encoder consists of five modules. Each of the first four modules includes a convolutional layer followed by a GDN layer and a PReLU activation, while the fifth module comprises only a convolutional layer and a GDN layer. The resulting feature maps are subsequently aggregated through a global average pooling layer, projected to a $2k$ -dimensional latent vector via a fully connected layer, and finally normalized through a power-normalization stage to produce the channel input symbols. This layered design progressively extracts representative features from the compressed image blocks, which are then aggregated to form the channel input symbols.

The PReLU activations are used in the encoder and channel-decoding layers because these parts operate on compressed continuous-valued representations before and after transmission. In this setting, negative feature responses may still carry useful structural and phase-related information, and forcing them to zero with standard ReLU can discard information in the communication bottleneck. PReLU avoids this by retaining negative responses through a learnable slope, which gives the network more flexibility than a fixed LeakyReLU. GDN layers, incorporated after convolutional layers, further stabilize feature statistics, reduce channel-wise redundancy, and improve nonlinear representation efficiency.

Within the CS-JSCC framework, GDN operates on the intermediate feature maps generated by the encoder. By normalizing and decorrelating local feature responses, GDN reduces redundancy

across channels and emphasizes perceptually significant structures such as edges and textures. This promotes a more compact and structured latent representation, enabling subsequent layers to allocate capacity more effectively for robust joint source-channel mapping under noisy channel conditions.

At the final stage of the encoder, the output of the last convolutional layer, denoted by $\tilde{\mathbf{z}} \in \mathbb{C}^k$, is normalized according to Eq. (3.7). Once normalized, the joint source-channel coded sequence is transmitted over the communication channel, with the real and imaginary components of $\mathbf{z}$ mapped directly onto the I and Q components of the signal. During transmission, the channel introduces random impairments that corrupt the transmitted symbols. To allow end-to-end optimization of the entire system, the channel is incorporated into the network architecture as a non-trainable layer, modeled using the transfer function described in the section 3.2.

Decoder Design

The CS-JSCC decoder, illustrated in Fig. 3.20, is designed to reconstruct the noisy, compressed signal received from the channel. Its operation proceeds in two main stages. First, the corrupted complex-valued inputs are mapped to an estimate of the original channel symbols. Next, a reconstruction network refines these estimates to recover the image blocks.

The initial stage $f_{\text{dec}}(\cdot; \theta_{\text{dec}})$ mirrors the encoding process of the transmitter. It begins by projecting the received channel symbols into a spatial feature representation via a fully connected layer. The resulting tensor is reshaped into a compact feature map, which serves as the input for the subsequent upsampling process. The upsampling is achieved through a sequence of five transposed convolutional layers (see Fig. 3.20). The first and last of these transposed convolutional layers are followed by an IGDN layer, while the intermediate transpose convolutional layers additionally include a PReLU activation layer after the IGDN layer. Together, these layers progressively transform the coded representations back into approximate image measurements. The transposed convolutional layers upsample the coded features, thereby increasing spatial resolution and mapping them back toward the original CS space. IGDN layers normalize the feature maps using learnable gain parameters, which improve training stability and convergence. The PReLU activations introduce nonlinearity, allowing the network to capture complex depen-

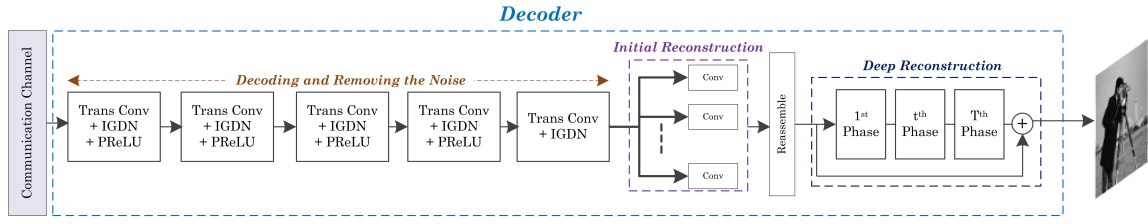


Figure 3.20: Proposed design for CS-JSCC decoder

dencies between the coded signal and the target image features. Collectively, these operations enhance the decoder's ability to mitigate channel distortions and produce a good approximation of the compressed image measurements.

The recovered CS measurements for each image block are subsequently used to reconstruct the original image in two stages: initial reconstruction, denoted as $f_{\text{init}}(\cdot; \theta_{\text{init}})$, and deep reconstruction, denoted as $f_{\text{deep}}(\cdot; \theta_{\text{dec}})$, as shown in Fig. 3.20. Both components are inspired by the multi-channel compressive reconstruction scheme proposed in [150], but are utilized here to operate in conjunction with the JSCC encoder-decoder and the channel model.

The initial reconstruction network f_{init} has s distinct input channels, matching the s sampling channels in f_{samp} . Assuming that the i -th image block $\mathbf{x}_i$ is routed to the j -th sampling channel at the encoder resulting in a measurement vector $\mathbf{y}_i$ (Eq. (3.15)), the approximate measurement vector $\hat{\mathbf{y}}_i$, generated after decoding and noise removal, is routed to the j -th channel of the initial reconstruction network at the decoder. Inside this channel, the recovery is performed via a linear mapping implemented via $3B^2 \times 1 \times 1 \times m_j$ convolutional kernels. The resulting from this convolution $3B^2$ output values are then spatially rearranged into a $B \times B \times 3$ structure, which is a coarse approximation of the original image block $\mathbf{x}_i$. Formally, this can be written as:

$$\hat{\mathbf{x}}_i = f_{\text{init}}(\hat{\mathbf{y}}_i, \mathbf{w}_j^{\text{init}}; \theta_{\text{init}}) \quad (3.16)$$

The initial reconstruction network f_{init} provides a coarse estimation of the image blocks from the recovered CS measurements at the receiver. Its primary purpose is to map the blockwise compressive measurements back into the spatial image domain, yielding an approximate reconstruction of the underlying RGB image. This coarse output consists of raw pixel intensities rather than feature representations, and therefore serves as an explicit image-domain initialization for the

subsequent refinement stage. Unlike the original multichannel BCS framework in [150], where reconstruction is performed from clean compressive measurements, our initial reconstruction module operates on measurements that have passed through both the encoding stage and a noisy communication channel. As a result, the estimate produced by f_{init} reflects the combined degradations of compression and wireless transmission. This estimate serves as a starting point for the deep reconstruction network, which performs iterative denoising and artefact removal to obtain the final high-quality reconstruction.

The deep reconstruction network, f_{deep} , is designed to iteratively enhance the initial estimate by progressively restoring spatial and perceptual details. It follows an unrolled iterative structure that mirrors classical CS reconstruction algorithms but uses learnable convolutional layers instead of explicit analytical operations. Each refinement stage consists of two steps: a blockwise approximation that enforces measurement consistency and a feature-based denoising step implemented using a compact DnCNN-inspired sub-network [153]. Within the denoising sub-network, ReLU activations are used because the module performs image-domain residual refinement, where a simple sparse nonlinearity is effective for suppressing local artefacts while keeping computational cost low. These denoising blocks are trained to suppress both compression artefacts and the distortions introduced by the wireless channel, enabling f_{deep} to learn an implicit noise model that adapts to the transmission conditions. The number of refinement stages determines the reconstruction depth, with each stage contributing incrementally to the overall image quality. For architectural details of the deep reconstruction network, interested readers are referred to [150].

In contrast to traditional deep CS approaches, the reconstruction in CS-JSCC is not trained in isolation. The parameters of f_{init} and f_{deep} are optimized jointly with those of the sampling network, the encoding and the decoding modules, through a channel model. This end-to-end learning strategy allows the entire system to co-adapt the sensing, encoding, and reconstruction stages for optimal performance under given channel conditions. As a result, the reconstruction network learns to recover visually faithful images even at low sampling rates and SNRs.

3.5.2 Loss Functions and Training Process

The proposed CS-JSCC framework is trained in two stages in an end-to-end manner to jointly optimize the sampling, encoding, decoding, and reconstruction components under realistic channel conditions. In the first stage, we optimize all the network parameters except for the parameters of the deep reconstruction module which is disabled at this stage. The sampling network f_{samp} generates channel-aware compressed measurements through learnable convolutional filters $\{w_j^{\text{samp}}\}_{j=1}^s$, and the saliency-based router allocates blocks to different sampling channels. The resulting multi-rate measurements are then passed to the encoding modules f_{enc} , which map them to complex-valued channel input symbols normalized to satisfy an average transmit-power constraint. After propagation through the wireless channel, the decoding modules f_{dec} produce an estimate of the compressed measurements at the receiver. These recovered measurements are subsequently fed into the initial reconstruction network f_{init} , which produces a coarse image-domain reconstruction. During this stage, the optimization objective is to minimize the discrepancy between this coarse reconstruction and the original image, so that the sampling front end, the encoding and decoding modules, and the initial reconstruction module learn to cooperate and provide a stable, channel-aware initialization for the subsequent deep reconstruction stage. The loss function for this stage is the MSE distortion between the coarse reconstruction produced at the receiver and the original image:

$$L_1 = \frac{1}{N} \sum_{n=1}^N \left\| \hat{\mathbf{x}}_0^n - \mathbf{x}^n \right\|_2^2, \quad (3.17)$$

where $\hat{\mathbf{x}}_0^n$ is defined as:

$$\begin{aligned} \hat{\mathbf{x}}_0^n &= f_{\text{init}}\left(f_{\text{dec}}(\hat{\mathbf{z}}; \theta_{\text{dec}}); \theta_{\text{init}}\right) \\ \hat{\mathbf{z}} &= \eta(\mathbf{z}) \\ \mathbf{z} &= f_{\text{enc}}\left(f_{\text{samp}}(\mathbf{x}^n; \theta_{\text{samp}}); \theta_{\text{enc}}\right) \end{aligned} \quad (3.18)$$

and is the output of the initial reconstruction network which serves as input to the deep reconstruction network. The parameters of both the sampling network and the initial reconstruction

network as well as the parameters of the encoding and decoding modules are learned at this first stage. Training this composite mapping end-to-end ensures that the sampling front end and the encoding and decoding modules co-adapt to the channel conditions and produce stable coarse reconstructions that will be further refined during the second stage of the training.

After this pre-training step, the sampling network is frozen while all remaining modules including the initial reconstruction network, the encoding and decoding modules, and the deep reconstruction network are jointly fine-tuned. During the second stage, the coarse reconstruction of the image blocks $\hat{\mathbf{x}}_0^n$ obtained from the initial reconstruction network is further refined through the deep reconstruction network. The optimization objective in this stage is to minimize a hybrid perceptual-fidelity loss that balances structure preservation and pixel-level accuracy:

$$L_2 = \frac{1}{N} \sum_{n=1}^N \left[(1 - \lambda)(1 - \text{SSIM}(\hat{\mathbf{x}}^n, \mathbf{x}^n)) + \lambda \|\hat{\mathbf{x}}^n - \mathbf{x}^n\|_1 \right], \quad (3.19)$$

where

$$\hat{\mathbf{x}}^n = f_{\text{dec}}(\hat{\mathbf{x}}_0^n; \theta_{\text{dec}}) \quad (3.20)$$

and $\hat{\mathbf{x}}_0^n$ is given in Eq. (3.18). At this training stage the parameters θ_{init} , θ_{enc} , θ_{dec} and θ_{deep} of the initial reconstruction network, the encoding and decoding modules and the deep reconstruction network, respectively, are learned jointly, while the parameters of the sampling network θ_{samp} remain fixed. The parameter $\lambda \in [0, 1]$ in the loss function L_2 controls the trade-off between perceptual quality and reconstruction accuracy. The SSIM term emphasizes structural consistency and perceptually meaningful details, while the ℓ^1 term enforces photometric accuracy and supplies non-saturating gradients even when structural similarity is high. Training these components jointly allows all receiver-side modules to adapt to the distortions introduced by the channel and the sampling process. Rather than attributing denoising to any single module, the entire receiver pipeline cooperatively refines the transmitted image, leading to high-fidelity reconstruction across a range of SNR conditions.

Also in proposed loss function, ℓ^1 loss is utilized instead of MSE in the second-stage reconstruction because MSE tends to over-penalize large local pixel errors, which often leads to

overly smooth reconstructions and loss of fine texture details. In contrast, the ℓ^1 loss is more robust to local outliers and better preserves edges and high-frequency image structures. In addition, the gradient magnitude of MSE decreases rapidly as the reconstruction approaches the target image, whereas the ℓ^1 loss maintains a more stable optimization signal during the later refinement stages. Empirically, it is observed that combining SSIM with ℓ^1 produces sharper visual reconstructions and improved PSNR and SSIM compared with SSIM+MSE under the same training conditions.

3.5.3 Results and Discussions

The proposed model is implemented in PyTorch and optimized using the Adam algorithm. The details of the model architecture are presented in Table 3.2. Each image is partitioned into non-overlapping blocks of size $B \times B$ as illustrated in Fig. 3.19. The proposed CS-JSCC architecture is trained on both the CIFAR-10 and ImageNet datasets. For CIFAR-10 images, size of blocks are set $B = 8$ which yields a 4×4 grid and 16 blocks per image. For ImageNet patches, $B = 16$ has been utilized which results in a 14×14 grid and 196 blocks per image. Unless stated otherwise, SR is set to $SR = 0.1$ and a multi-channel sampling network with $s = 7$ channels is considered, which provides a balanced trade-off between adaptivity and efficiency, as it is further discussed in the following.

Each channel is optimised for one of the sampling rates in the set:

$$\mathcal{S} = \{0.01, 0.03, 0.05, 0.10, 0.20, 0.30, 0.40\} \quad (3.21)$$

While $s = 7$ channels are used in the experiments to balance flexibility and complexity, the network architecture remains fully scalable and can accommodate additional predefined rates if finer rate control is desired. The deep reconstruction network is configured with $T = 6$ phases, each consisting of $l = 4$ convolutional layers, resulting in a total of 24 convolutional layers across the deep reconstruction module. The number of phases T was determined through empirical optimization to balance reconstruction fidelity against receiver-side computational complexity. While standalone unrolled CS architectures often implement a larger number of

Table 3.2: Layer configuration of CS-JSCC. Output sizes are given as Channels $\times$ Height $\times$ Width. B is the block size, K is the kernel size, S is the stride, and P is the padding.

#	Layer / Operation	K/S/P	Out Ch.	Output size
Sampling network (per channel j)				
$S1_j$	Conv (no bias/act)	$B/B/0$	m_j	$m_j \times 1 \times 1$
Encoder (feature extraction and symbol mapping)				
E1	Conv+GDN+PReLU	3/1/1	64	$64 \times B \times B$
E2	Conv+GDN+PReLU	5/2/2	128	$128 \times B/2 \times B/2$
E3	Conv+GDN+PReLU	5/2/2	256	$256 \times B/4 \times B/4$
E4	Conv+GDN+PReLU	3/1/1	256	$256 \times B/4 \times B/4$
E5	Conv+GDN	1/1/0	256	$256 \times B/4 \times B/4$
E6	GlobalAvgPool	–	256	256 (vector)
E7	Linear	–	$2k$	$2k$ (vector)
E8	PowerNorm (I/Q)	–	$2k$	$2k$ (vector)
Decoder (symbol expansion and up-sampling)				
D0	Linear (reshape)	–	$256 \times B/4 \times B/4$	$256 \times B/4 \times B/4$
D1	TransConv+IGDN	1/1/0	256	$256 \times B/4 \times B/4$
D2	TransConv+IGDN+PReLU	3/1/1	256	$256 \times B/4 \times B/4$
D3	TransConv+IGDN+PReLU	5/2/2	128	$128 \times B/2 \times B/2$
D4	TransConv+IGDN+PReLU	5/2/2	64	$64 \times B \times B$
D5	TransConv+IGDN	3/1/1	3	$3 \times B \times B$
Initial reconstruction (per channel j)				
$R0_j$	Conv (reshape to pixels)	1/1/0	$3B^2$	$3B^2 \times 1 \times 1$ ($\rightarrow 3 \times B \times B$)
Deep reconstruction (per phase, repeated T times)				
R1	Conv+ReLU	3/1/1	d	$d \times H \times W$
R2	Conv+ReLU	3/1/1	d	$d \times H \times W$
R3	Conv+ReLU	3/1/1	d	$d \times H \times W$
R4	Conv (residual)	3/1/1	3	$3 \times H \times W$

phases (e.g., $T = 10$ in [150]), our simulations under joint source-channel conditions revealed that reconstruction quality (PSNR/SSIM) strictly saturates beyond $T = 6$. Limiting the unrolled depth to 6 phases significantly curtails processing latency and memory footprint at the receiver, ensuring real-time suitability for edge intelligence deployments with no measurable degradation in final image accuracy. The parameters of both GDN and IGDN layers are initialized following the guidelines in [154], with $\beta = 0.1$ and $\gamma \sim \mathcal{N}(0, 0.1^2)$.

The compression ratio is defined as $R = k/n$, where k and n represent the channel bandwidth (number of symbols transmitted over the channel) and source bandwidth (number of source symbols), respectively. The channel is given by Eq. (3.9) and is varied across different experimental conditions. The reconstruction quality is evaluated using the PSNR as stated in Eq. (2.6).

The performance of the proposed CS-JSCC method is compared against state-of-the-art deep learning-based JSCC methods, namely Deep JSCC [6], ADJSCC [51], SWinJSCC [25] and NTSCC [155].

Evaluation on CIFAR-10 Dataset

The evaluation begin by assessing the performance of the proposed CS-JSCC scheme on the CIFAR-10 dataset [129]. For training, 50,000 images are used together with random channel realizations, while evaluation is performed on a separate test set of 10,000 images that are not seen during training. The training procedure employs an initial learning rate of 10^{-3} , which is reduced to 10^{-4} after 500k iterations. A mini-batch size of 64 is used, and training is continued until no further improvement is observed on the test set. During evaluation, each test image is transmitted 10 times to mitigate the randomness introduced by the channel and ensure statistically reliable results. The learning-rate schedule and number of training iterations were selected based on convergence behaviour. In the first training stage, the sampling network, encoder, decoder, and initial reconstruction module require sufficient iterations to learn a stable measurement and coarse reconstruction space. Empirically, most of the loss reduction occurs before approximately 500,000 iterations, after which the improvement in validation PSNR becomes marginal. Therefore, the learning rate is reduced at this point to support stable refinement. In the second stage, the sampling network is frozen and the remaining modules are trained until the hybrid SSIM and ℓ^1 loss reaches a stable plateau.

The proposed CS-JSCC algorithm is firstly evaluated under the AWGN channel model, where the transfer function $\eta_n(\cdot)$ is given in Eq. (3.3). In this setting, the channel SNR is varied by adjusting the noise variance σ^2 , and the performance of CS-JSCC is compared against the benchmark methods. To assess robustness, a study is done on how well the proposed scheme generalizes to channel conditions different from those used during training. Figs. 3.21 and 3.22 present the average PSNR of reconstructed images as a function of the test SNR for two bandwidth compression ratios. Unless otherwise stated, the proposed CS-JSCC model reported in the comparison figures is trained using randomly sampled SNR values in the range $[0, 20]$ dB. This setting follows the channel-adaptive evaluation protocol used by ADJSCC, SwinJSCC and

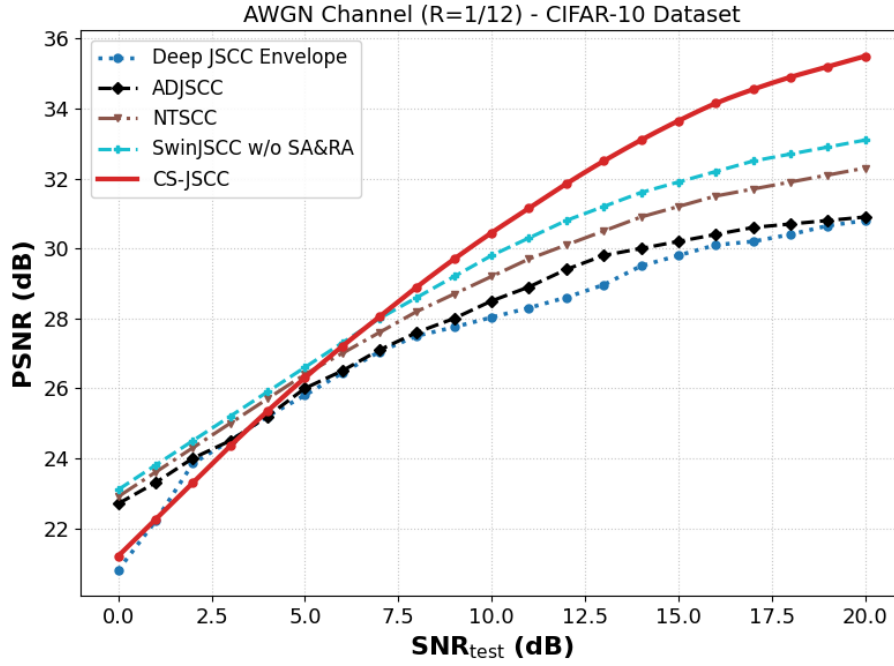


Figure 3.21: Performance of the proposed CS-JSCC versus varying compression ratios over an AWGN channel with $R = 1/12$ in CIFAR-10 dataset

NTSCC and allows a single CS-JSCC model to be evaluated across the full range of SNR_{test} values. For the fixed-SNR Deep JSCC baselines, the corresponding models trained at the specified $\text{SNR}_{\text{train}}$ values are reported.

As can be seen in Figs. 3.21 and 3.22, which correspond to bandwidth compression ratios $R = 1/12$ and $R = 1/6$, respectively, the proposed CS-JSCC model trained over $\text{SNR}_{\text{train}} \in [0, 20]$ dB achieves stable performance across the full range of test SNR values. Compared with the fixed-SNR Deep JSCC baselines, CS-JSCC provides consistently higher PSNR in both bandwidth settings, particularly in the medium-to-high SNR regime where fixed-SNR Deep JSCC models tend to saturate. Compared with ADJSCC, CS-JSCC shows comparable performance at low SNR and achieves clear gains as SNR_{test} increases. The comparison with the stronger recent baselines, NTSCC and SwinJSCC, further demonstrates the benefit of the proposed framework. At very low SNR values, SwinJSCC and NTSCC can achieve slightly higher PSNR due to their strong learned nonlinear representations. However, as the channel condition improves, CS-JSCC gradually surpasses both methods. This trend is observed for both $R = 1/12$ and $R = 1/6$, with the advantage becoming more evident in the medium-to-high SNR range. The results indicate that the structured CS measurement representation, combined with saliency-aware sampling and

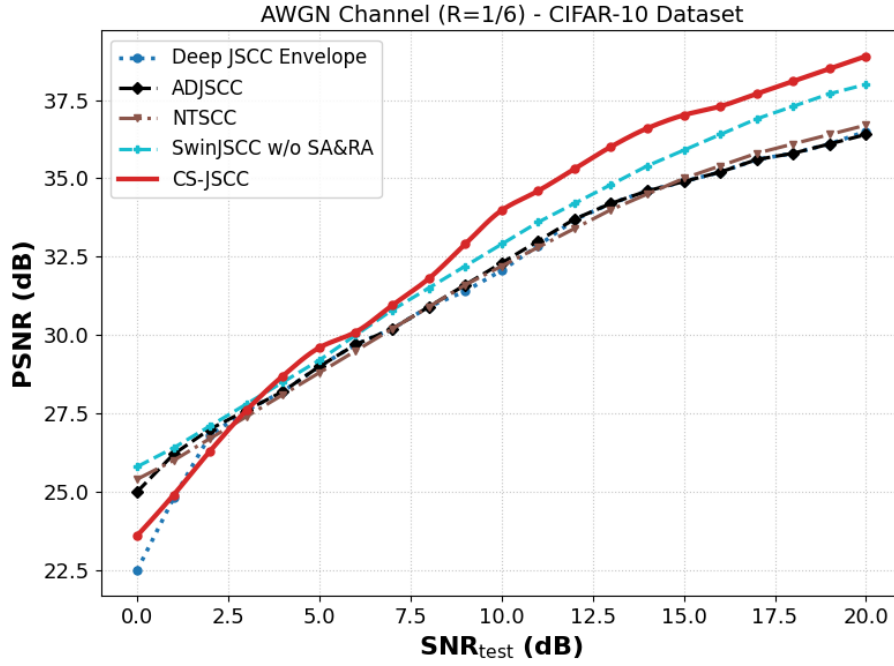


Figure 3.22: Performance of the proposed CS-JSCC versus varying compression ratios over an AWGN channel with $R = 1/6$ in CIFAR-10 dataset

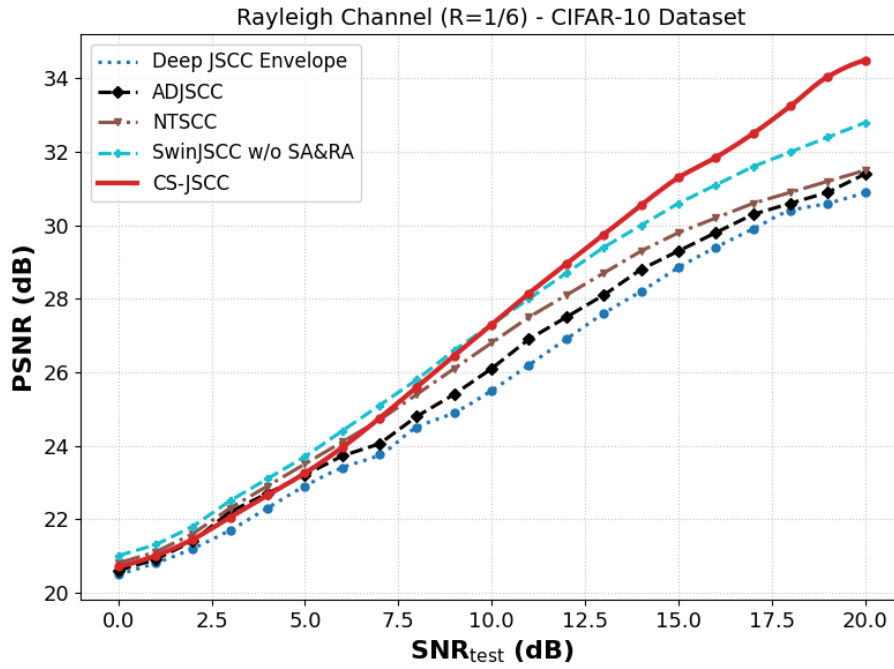


Figure 3.23: Performance of the proposed CS-JSCC versus varying compression ratios over a Rayleigh channel with $R = 1/6$ in CIFAR-10 dataset

two-stage reconstruction, is particularly effective when the channel is sufficiently reliable to preserve fine image details.

The performance of the proposed CS-JSCC scheme is also evaluated under a slow Rayleigh

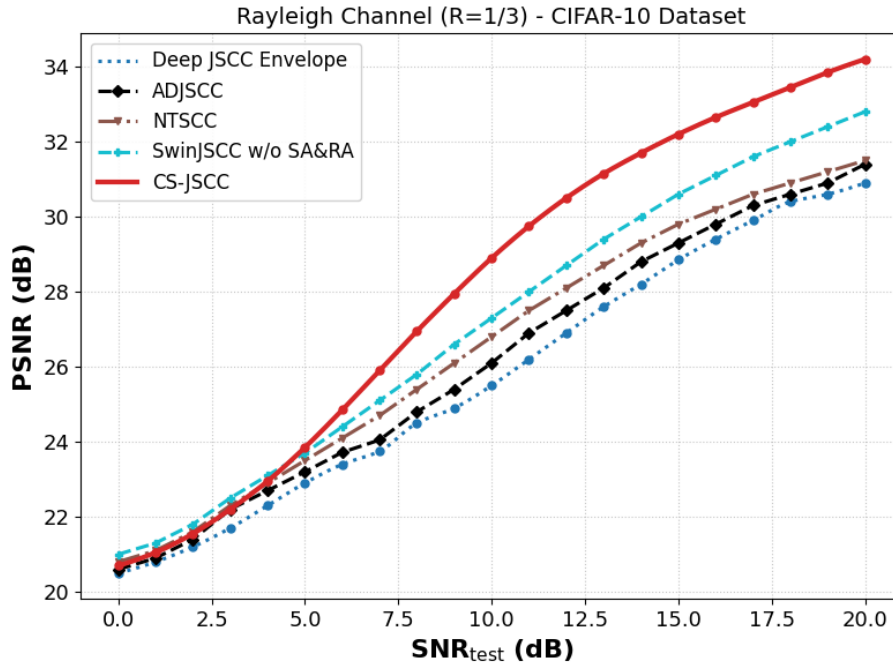


Figure 3.24: Performance of the proposed CS-JSCC versus varying compression ratios over an Rayleigh channel with $R = 1/3$ in CIFAR-10 dataset

fading channel. In this scenario, the channel gain h is modeled as a random variable drawn from a complex Gaussian distribution, $h \sim \mathcal{CN}(0, H_c)$, for each transmitted image. During the transmission of a single image, h remains constant, but it changes independently to a new realization for the subsequent image. In the experiments, H_c is set to $H_c = 1$ and vary the noise variance σ^2 to simulate different average SNR conditions. Figs. 3.23 and 3.24 demonstrate the robustness of CS-JSCC to variations in average channel SNR under slow Rayleigh fading. As expected, the overall performance decreases compared to the static AWGN channel due to the additional fading impairments. Nevertheless, the reconstructed images retain reasonably high quality, confirming that CS-JSCC maintains strong resilience in more challenging fading environments.

To complement the PSNR-based evaluation, LPIPS is also reported for the CIFAR-10 experiments over the AWGN channel. PSNR measures pixel-level reconstruction fidelity, but it does not fully represent perceptual similarity, especially when two reconstructed images have similar distortion values but different texture quality or visual sharpness. Recent JSCC studies therefore include LPIPS as an additional perceptual-quality metric, where lower values indicate greater perceptual similarity. Fig. 3.25 shows that the LPIPS values generally decrease as the

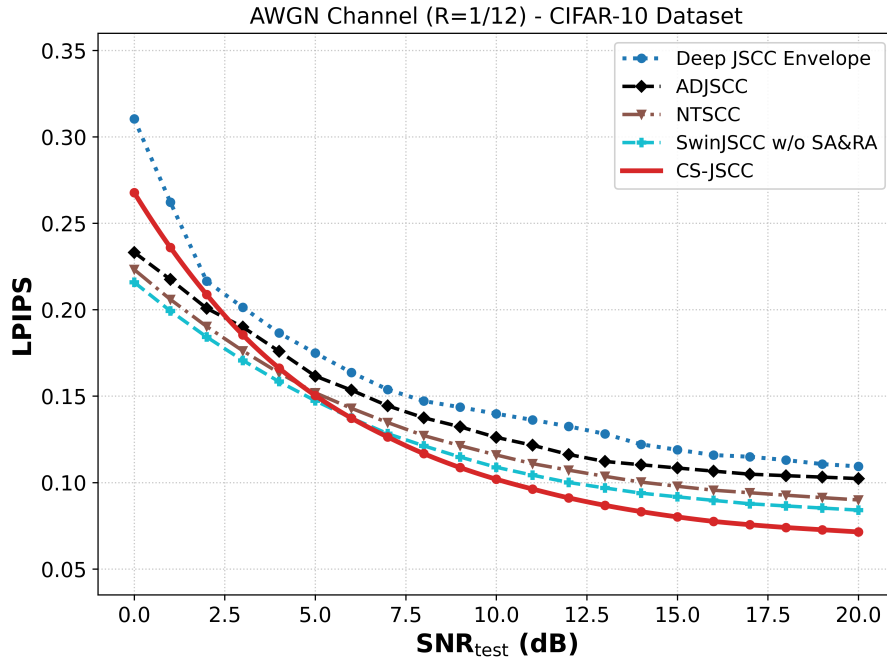


Figure 3.25: LPIPS comparison of CS-JSCC and baseline methods over the AWGN channel on the CIFAR-10 dataset at $R = 1/12$

test SNR increases. At low SNR, the perceptual quality of all methods is limited by severe channel noise, while at medium and high SNR values, CS-JSCC achieves a lower LPIPS than the baseline methods. This trend is consistent with the PSNR results and indicates that the proposed saliency-aware compressive JSCC design improves both pixel-level fidelity and perceptual image quality.

The LPIPS evaluation is further extended to the Rayleigh fading channel in Fig. 3.26. Compared with the AWGN case, the Rayleigh channel introduces stronger fading-related distortions, so the LPIPS values are generally higher for the same SNR range. Nevertheless, the same overall trend is observed: increasing test SNR reduces the perceptual distance for all methods. The Deep JSCC envelope and ADJSCC improve gradually but remain limited by their fixed or attention-based latent representations. NTSCC and SwinJSCC provide stronger perceptual quality than the basic Deep JSCC baseline, while CS-JSCC achieves the lowest LPIPS at medium-to-high SNR values. This supports the effectiveness of combining adaptive multi-channel CS sampling with deep reconstruction refinement under fading channels.

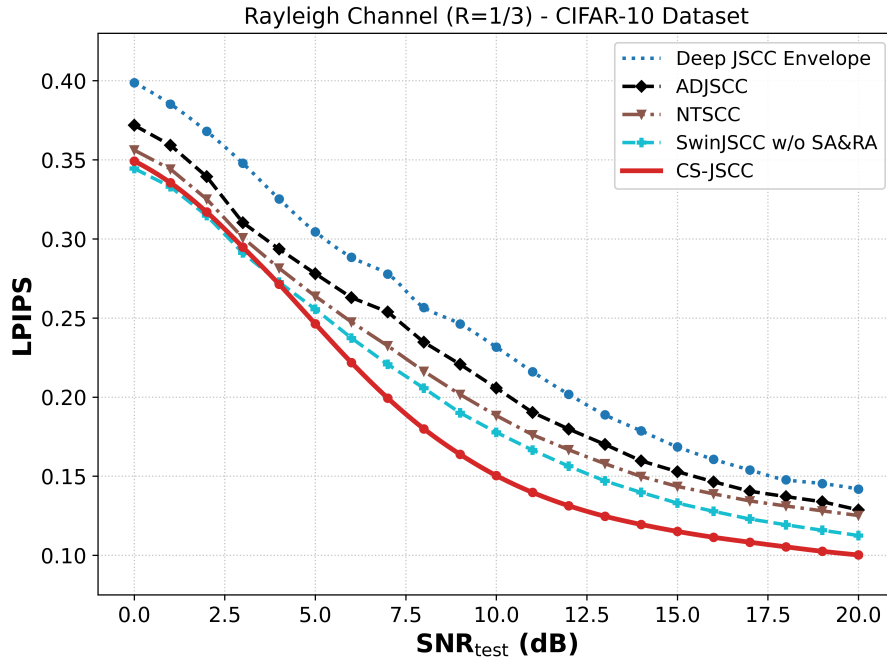


Figure 3.26: LPIPS comparison of CS-JSCC and baseline methods over the Rayleigh fading channel on the CIFAR-10 dataset at $R = 1/3$.

Evaluation on Kodak Dataset

The proposed CS-JSCC framework is further evaluated on higher-resolution images. To this end, the architecture is trained on the ImageNet dataset. During training, images are randomly cropped into patches of size 224×224 , and processed in mini-batches of 32 samples. The learning rate is initialized at 10^{-4} , and training is continued until convergence. A 9:1 split is used for training and validation. For final testing, the Kodak dataset is employed [131]. Each image in the Kodak dataset is transmitted 100 times to account for channel randomness, and the results are averaged over these realizations. The evaluation considers both AWGN and Rayleigh fading channels, providing a comprehensive assessment of CS-JSCC under different channel conditions.

The performance of the proposed CS-JSCC algorithm is evaluated over an AWGN channel using the Kodak dataset. Fig. 3.27 presents a comparison of the CS-JSCC scheme with benchmark approaches across different SNR levels. The results show that, although ADJSCC, NTSCC, and SwinJSCC achieve slightly higher PSNR at the very low-SNR points, CS-JSCC rapidly improves as the channel condition becomes more reliable and surpasses all compared methods from the low-to-moderate SNR region onward. The gain is especially clear at medium

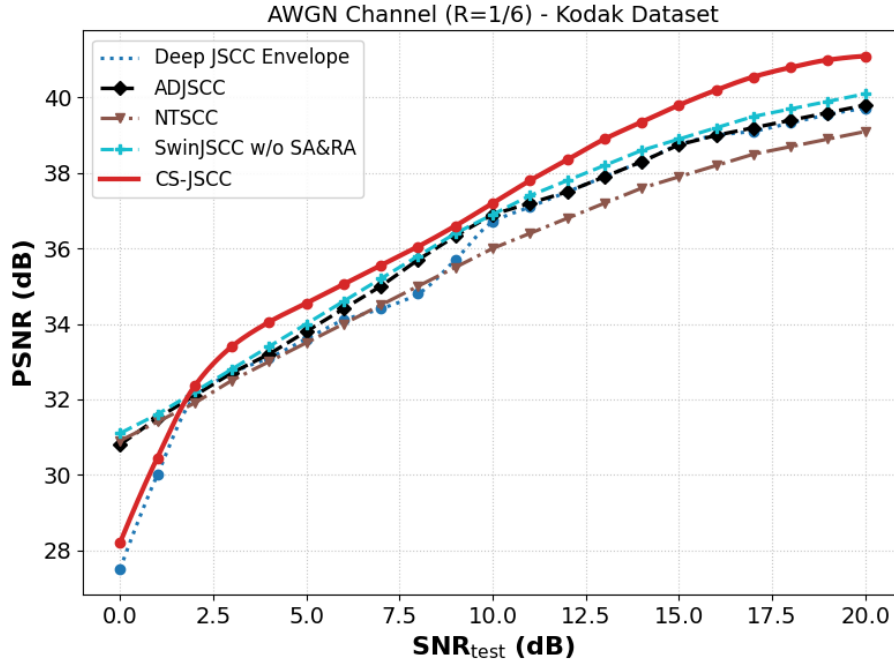


Figure 3.27: Performance of the proposed CS-JSCC versus varying compression ratios over an AWGN channel with $R = 1/6$ in Kodak dataset

and high SNR values, where CS-JSCC benefits from the structured CS measurements and the two-stage reconstruction process. The CS front-end enables efficient compression by acquiring compact measurements while preserving essential image information, and the learned encoder maps these measurements into robust channel symbols. At the receiver, the reconstruction network effectively suppresses channel-induced noise and refines fine visual details. In addition, the adaptive sampling strategy assigns higher sampling rates to visually important regions, further improving reconstruction quality under bandwidth-constrained transmission.

The performance of the proposed CS-JSCC scheme is also evaluated under a Rayleigh fading channel and compare it with Deep JSCC, ADJSCC, NTSCC, and SwinJSCC. Fig. 3.28 illustrates the results for the Kodak dataset at bandwidth compression ratio $R = 1/6$. The results show that CS-JSCC consistently outperforms Deep JSCC across the full SNR range, confirming its robustness against fading effects. Compared with ADJSCC, NTSCC, and SwinJSCC, CS-JSCC achieves comparable performance at very low SNR values and gradually provides higher PSNR as SNR_{test} increases. The improvement is particularly clear in the medium-to-high SNR regime, where the structured CS measurements and two-stage reconstruction network can more effectively exploit the improved channel quality. Unlike standard autoencoder-based JSCC models, the CS sampling stage produces compact measurement representations that spread image information

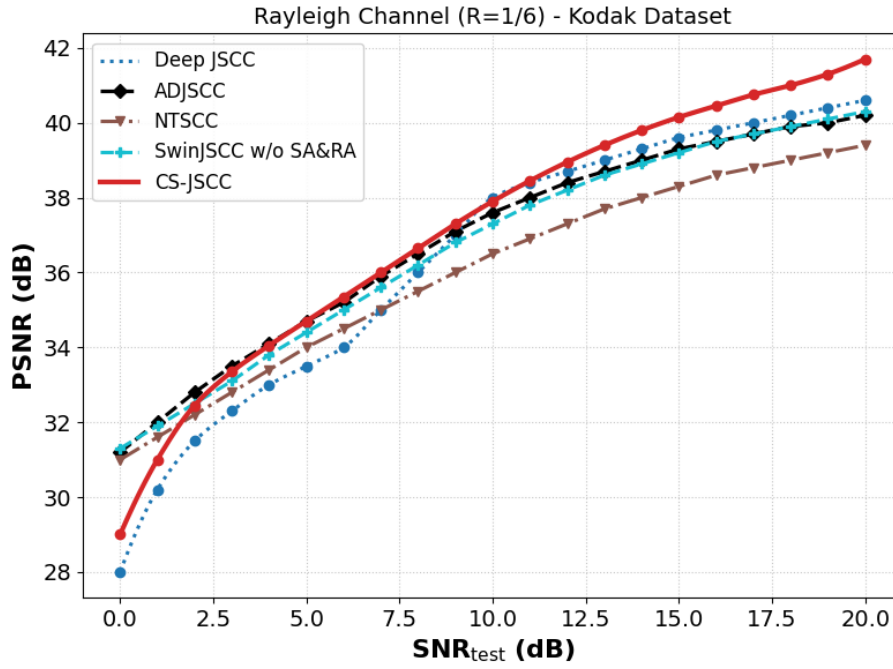


Figure 3.28: Performance of the proposed CS-JSCC versus varying compression ratios over an Rayleigh channel with $R = 1/6$ in Kodak dataset

across multiple coefficients before channel encoding. This provides the reconstruction network with a more informative and structured input for suppressing fading-induced distortion and recovering fine image details. The adaptive sampling mechanism further improves performance by allocating more measurements to visually important regions while compressing smoother regions more aggressively.

Fig. 3.29 reports the LPIPS comparison on the Kodak dataset over the Rayleigh fading channel. Since Kodak images contain richer textures and higher-resolution structures than CIFAR-10, LPIPS is useful for assessing whether the reconstructed images preserve perceptually relevant details beyond pixel-wise fidelity. The LPIPS values decrease as the test SNR increases, which is consistent with the improvement observed in PSNR. At low SNR, all methods are affected by fading distortion, but CS-JSCC achieves lower LPIPS at medium-to-high SNR values. This indicates that the proposed saliency-aware sampling and deep reconstruction refinement improve not only distortion-based reconstruction performance but also perceptual image quality under Rayleigh fading.

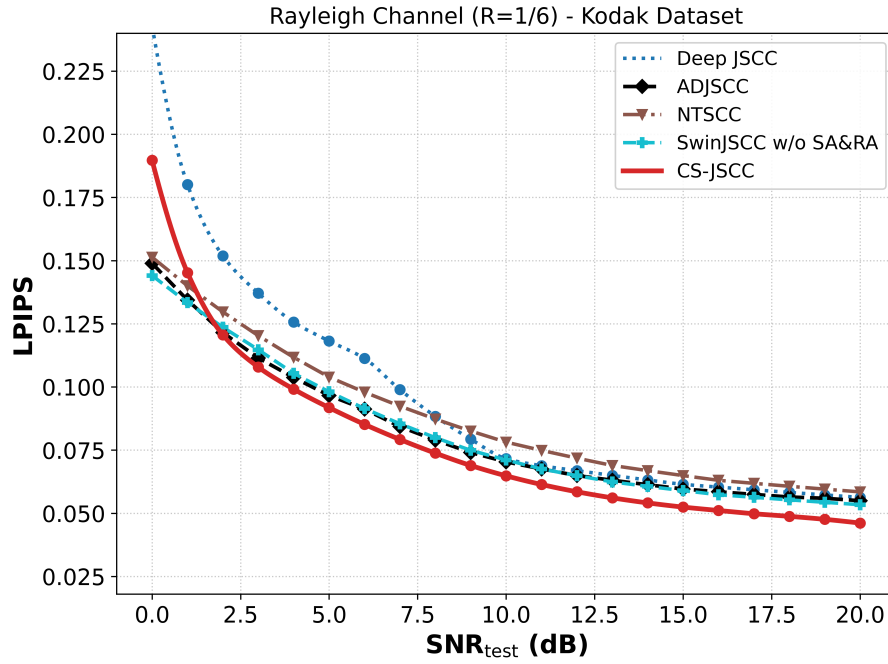


Figure 3.29: LPIPS comparison of CS-JSCC and baseline methods over the Rayleigh fading channel on the Kodak dataset at $R = 1/6$.

Impact of Global Sampling Rate SR

The impact of the global target sampling rate SR on the performance of the proposed CS-JSCC framework is evaluated as follows. In this study, the overall bandwidth compression ratio is fixed to $R = 1/6$, and the value of SR is varied to analyze its effect on the reconstruction quality. For small values of SR , the sampling module applies aggressive compression and produces a limited number of measurements, thereby restricting the amount of information available to the encoding module. In contrast, for larger values of SR , more measurements are retained, however, the encoding module must then apply stronger compression when mapping measurements to the k channel symbols in order to satisfy the fixed bandwidth compression ratio R .

The results, presented in Table 3.3, demonstrate the effect of varying SR on PSNR values for the Kodak dataset with $R = 1/6$, $SNR_{\text{train}} = 2$ dB and $SNR_{\text{test}} = 0$ dB. The analysis reveals that the highest PSNR (27.42 dB) is achieved when the global target sampling rate is set to $SR = 0.10$. This observation suggests that a moderately low sampling rate, where the sampling module already performs a significant amount of compression but still preserves sufficient information for the encoding module, results in optimal image reconstruction quality. For smaller values of SR (e.g., $SR = 0.05$), the CS stage becomes overly aggressive, leading to a noticeable loss of

Table 3.3: Effect of global target sampling rate SR on reconstruction quality (Kodak dataset, $R = 1/6$, $SNR_{\text{train}}=2$ dB, $SNR_{\text{test}}=0$ dB).

SR	0.05	0.075	0.10	0.125	0.15	0.175	0.20	0.25
PSNR (dB)	26.50	26.90	27.42	27.30	27.20	27.05	26.95	26.80

visual information and reduced PSNR. On the other hand, increasing SR beyond 0.10 yields only marginal gains or even slight performance degradation; the additional measurements must be compressed more strongly by the encoding module to satisfy the same bandwidth compression ratio, which can limit robustness to channel noise and partially offset the benefit of higher sampling density. Overall, these results highlight the importance of balancing compression between the sampling and encoding modules. While the proposed framework offers flexibility in selecting SR , excessively low or high values lead to suboptimal reconstruction quality.

Based on this study, $SR = 0.10$ was selected as a good operating point and is adopted in all experiments unless stated otherwise. Although this choice was validated for the Kodak dataset and the considered channel conditions, the same tuning procedure can be applied to other datasets, bandwidth compression ratios, and SNR regimes to determine optimal operating points for different application scenarios.

Ablation Study on the Loss Function Hyper-parameter

Fig. 3.30 illustrates the impact of the hyper-parameter λ on the reconstruction quality. The L_2 loss function, defined in Eq. (3.19), is a hybrid loss which combines the SSIM and the ℓ^1 norm. The value of λ controls the relative weighting between the perceptual similarity (SSIM) and the pixel-wise accuracy (ℓ^1 norm). A higher λ prioritizes pixel accuracy, while a lower λ promotes perceptual similarity. Analyzing the results for the Kodak dataset at a bandwidth compression ratio of $R = 1/6$, $SNR_{\text{train}} = 4$ dB and $SNR_{\text{test}} = 2$ dB, the model achieves its highest reconstruction PSNR value of 32.6 dB at $\lambda = 0.3$. This indicates that a balanced weighting between SSIM and ℓ^1 loss near $\lambda = 0.3$ yields the best reconstruction performance in terms of PSNR. When λ increases significantly beyond 0.3, the PSNR decreases, which indicates that relying too heavily on pixel-wise accuracy degrades the perceptual fidelity and the overall reconstruction performance. Hence, perceptual guidance from the SSIM term is essential for optimal performance. However, high perceptual similarity alone does not guarantee the best

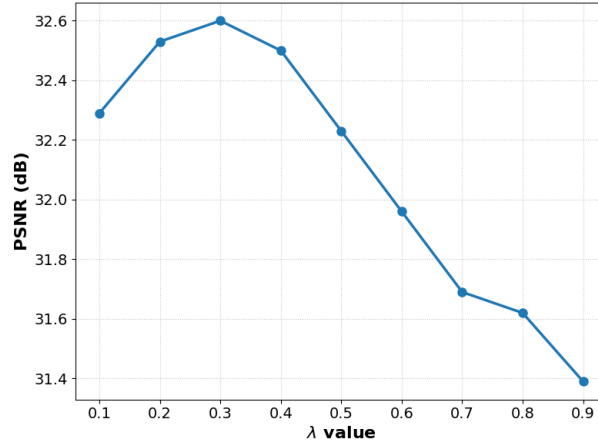


Figure 3.30: Comparison between hyper-parameter values of L_2 loss function (Kodak Dataset - $R = 1/6$ - $SNR_{\text{train}} = 4$ dB - $SNR_{\text{test}} = 2$ dB)

possible reconstruction performance and overemphasizing the SSIM term in the loss function should be avoided as suggested by the decreasing PSNR values when λ decreases beyond 0.3.

Ablation Study on the Number of Sampling Channels

To evaluate the effect of the number of channels in the multi-channel sampling network, an ablation study is conducted in which the number of sampling paths s was varied while keeping all other parameters fixed. To ensure a fair comparison, for each configuration of the sampling network with s channels, the associated set of predefined sampling rates $\mathcal{S}_s$ was chosen as a subset of the same set $\{0.01, 0.03, 0.05, 0.10, 0.20, 0.30, 0.40\}$. In particular, for $s = 1$, a single sampling rate $\mathcal{S}_1 = \{0.10\}$ is used; for $s = 3$ $\mathcal{S}_3 = \{0.05, 0.10, 0.20\}$ is used; for $s = 5$ $\mathcal{S}_5 = \{0.03, 0.05, 0.10, 0.20, 0.30\}$; is utilized and for $s = 7$ the full set $\mathcal{S}_7 = \{0.01, 0.03, 0.05, 0.10, 0.20, 0.30, 0.40\}$ is utilized. For the configurations with $s = 9$ and $s = 11$, the additional sampling rates refine this set, but the minimum and maximum sampling rates remain fixed at 0.01 and 0.40, respectively. Each configuration was trained and tested under identical conditions using CIFAR-10 with a global target sampling rate of $SR = 0.1$, corresponding to an average compression ratio of 10% across the image. The performance was evaluated in terms of PSNR and SSIM, while computational complexity was assessed through the number of trainable parameters (M) and inference time per image. The results are summarized in Table 3.4.

As seen in Table 3.4, increasing the number of channels from $s = 1$ to $s = 5$ leads to a clear improvement in reconstruction quality. The PSNR increases by nearly 2.3 dB compared to the

Table 3.4: Ablation study on the number of sampling channels (s) in the proposed CS-JSCC system. Results are averaged over the CIFAR-10 test set with global sampling rate $SR = 0.1$.

# Channels (s)	PSNR (dB)	SSIM	Parameters (M)	Inference Time (ms)
1	28.76	0.842	1.42	7.9
3	30.21	0.871	1.53	8.3
5	31.02	0.883	1.59	8.7
7	31.47	0.891	1.64	9.1
9	31.49	0.892	1.73	9.6
11	31.45	0.889	1.82	10.2

single-channel configuration (where all image blocks are sampled at the same rate equal to the global target rate SR), while SSIM improves from 0.842 to 0.883. This gain demonstrates the benefit of introducing multiple sampling paths, which provide finer granularity in rate allocation and better alignment with local image saliency. However, the improvements begin to saturate beyond $s = 7$, where additional channels contribute only marginal gains in PSNR and SSIM despite increasing parameter count and inference time. When $s > 9$, the model becomes slightly over-parameterized, offering no tangible quality benefit while increasing training complexity and memory footprint.

From a complexity perspective, the sampling module adds only a small overhead relative to the overall model, since each additional channel contributes a lightweight convolution layer with limited kernel size. The inference time grows almost linearly with s , but the total cost remains low compared to the deep encoder and decoder. Therefore, the seven-channel configuration achieves the most favorable trade-off between performance and computational efficiency. It provides sufficient adaptivity to image content without unnecessary redundancy or instability during training. These findings confirm that the multi-channel structure enhances the model’s ability to perform spatially adaptive sampling while maintaining low complexity. The chosen configuration of $s = 7$ channels offers an optimal balance between reconstruction accuracy and computational cost, justifying its adoption in the proposed CS-JSCC framework.

Component-Level Ablation Study

To further isolate the contribution of the main components of CS-JSCC, a component-level ablation study is conducted on CIFAR-10 under the AWGN channel. All variants are evaluated under the same setting with $SR = 0.1$, $R = 1/6$, and $SNR_{\text{test}} = 7$ dB. Three ablated configurations are considered. Variant A removes the explicit CS sampling and initial CS reconstruction stages

Table 3.5: Component-level ablation study on CIFAR-10 under AWGN channel with $SR = 0.1$, $R = 1/6$, and $SNR_{\text{test}} = 7$ dB.

Variant	Configuration	CS/Adaptive	f_{deep}	PSNR/SSIM
A	No explicit CS	No/No	Yes	29.84 / 0.864
B	Uniform CS, $s = 1$	Yes/No	Yes	28.76 / 0.842
C	Adaptive CS w/o f_{deep}	Yes/Yes	No	30.48 / 0.876
Full	Full CS-JSCC, $s = 7$	Yes/Yes	Yes	31.47 / 0.891

and replaces them with a direct learned JSCC bottleneck of comparable output dimensionality, while retaining the same deep reconstruction module. This variant tests whether the gain comes only from a learned compact representation and receiver-side refinement, rather than from the CS measurement structure. Variant B keeps the CS sampling structure but uses a single-channel uniform CS sampler with $s = 1$, so all image blocks are sampled at the same rate without adaptive block-wise allocation. Variant C retains the adaptive CS front-end but removes the deep reconstruction module f_{deep} .

As shown in Table 3.5, Variant A achieves lower performance than the full CS-JSCC model even though it retains the same deep reconstruction module. This shows that the improvement is not solely caused by receiver-side refinement. Variant B further shows that simply using a CS-style block sampler is not sufficient by itself, because uniform CS sampling without adaptive allocation gives the lowest PSNR among the ablated variants. This indicates that the benefit of the CS front-end becomes most effective when it is combined with the saliency-aware multi-channel sampling mechanism. Variant C removes only the deep reconstruction module and still achieves 30.48 dB, showing that the adaptive CS front-end and initial reconstruction already provide a strong reconstruction starting point. Adding f_{deep} improves the result further to 31.47 dB, confirming that the deep reconstruction network provides an additional refinement gain. Overall, the ablation confirms that the final improvement comes from the coordinated use of adaptive CS measurements, rate-aware initial reconstruction, and receiver-side deep refinement, rather than from one component alone.

Complexity and Latency Analysis

To provide a fairer comparison between the proposed CS-JSCC framework and the baseline methods, we further evaluate computational complexity and inference latency. For all methods,

Table 3.6: Complexity and latency comparison of different JSCC-based image transmission methods. Latency values are reported per image and averaged over the test set.

Method	Params (M)	MACs (G)	Enc. (ms)	Dec. (ms)	Total (ms)
Deep JSCC	0.76	0.42	1.5	1.8	3.3
ADJSCC	1.28	0.58	2.1	2.7	4.8
NTSCC	32.90	5.42	36.0	15.0	51.0
SwinJSCC	18.34	4.15	13.0	13.0	26.0
CS-JSCC	1.64	0.93	2.6	6.5	9.1

Multiply-Accumulate operations (MACs) and latency are reported for input images with resolution $224 \times 224 \times 3$. All inference-time measurements are conducted on the same hardware platform using an NVIDIA RTX 3090 GPU, and the reported latency values are averaged over the Kodak test dataset. The encoder latency includes source-to-channel-symbol mapping, while the decoder latency includes channel-symbol recovery and image reconstruction. For CS-JSCC, the decoder latency includes the channel decoding module, the initial reconstruction module, and the deep reconstruction network.

As shown in Table 3.6, Deep JSCC has the lowest computational cost, but this comes at the expense of reconstruction quality, especially in the medium-to-high SNR regime. ADJSCC introduces attention-based channel adaptation with a moderate increase in parameters and latency. NTSCC and SwinJSCC provide stronger representation capacity, but they require substantially larger models and higher computational cost. In particular, NTSCC relies on nonlinear transform coding and entropy-model-guided transmission, while SwinJSCC uses a Transformer backbone to increase model capacity.

The proposed CS-JSCC introduces additional latency compared with Deep JSCC and ADJSCC, mainly because of the two-stage reconstruction network at the receiver. However, its total inference time remains considerably lower than the more complex NTSCC and SwinJSCC baselines. The encoder remains lightweight because the multi-channel CS sampling front-end consists only of small convolutional sampling paths. Therefore, CS-JSCC offers a favourable performance-complexity trade-off. It improves reconstruction quality and structural fidelity while keeping the parameter count and latency within a practical range. This supports the design choice of using the seven-channel sampling configuration, which, as shown in Table 3.4, provides strong reconstruction performance with only a modest increase in inference time.

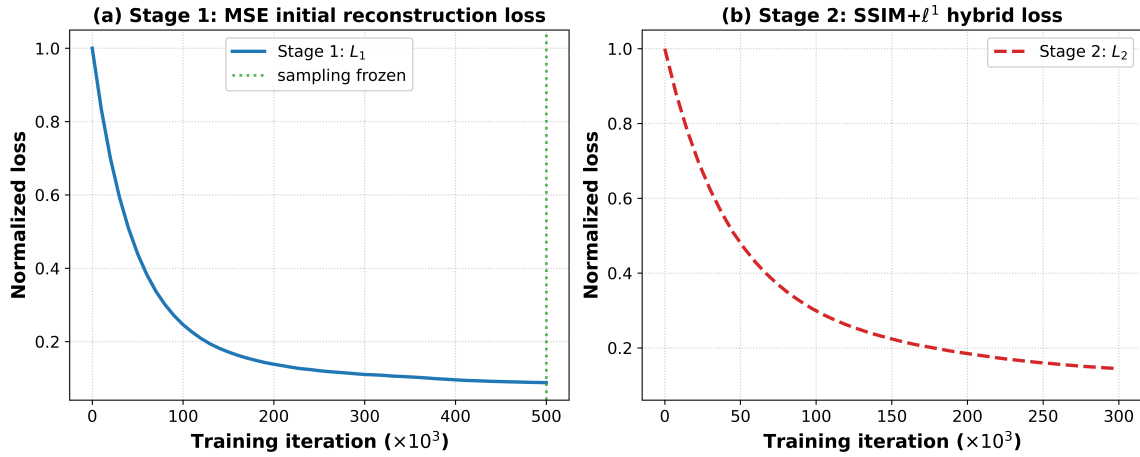


Figure 3.31: Normalized loss convergence of CS-JSCC: (a) Stage 1 MSE-based initial reconstruction loss L_1 , and (b) Stage 2 hybrid SSIM and ℓ^1 reconstruction loss L_2 .

Training Loss Convergence

Fig. 3.31 shows normalized loss convergence curves for the two training stages of CS-JSCC. In Stage 1, the MSE-based loss L_1 decreases rapidly as the sampling network, encoder, decoder, and initial reconstruction module learn a stable coarse reconstruction space. Once this stage reaches a near-plateau, the sampling network is frozen. In Stage 2, the hybrid SSIM and ℓ^1 loss L_2 decreases more gradually because the deep reconstruction module performs structure-preserving refinement. The two curves are shown separately because the two stages optimize different loss functions with different meanings and scales.

Visual Comparison

Fig. 3.32 provides a qualitative comparison of reconstructed images produced by the proposed CS-JSCC model and four baseline methods across different SNR conditions in an AWGN channel. For each method, both PSNR and SSIM values are reported. The visual examples show that reconstruction quality improves progressively from Deep JSCC and ADJSCC to the more recent SwinJSCC and NTSCC, with CS-JSCC consistently producing the clearest and most structurally faithful reconstructions. At low SNR (e.g., 2 dB), Deep JSCC exhibits severe noise amplification and textured artifacts, while ADJSCC reduces some of the channel distortion but still suffers from noticeable structural degradation. SwinJSCC and NTSCC further improve reconstruction quality, preserving more image content and reducing noise. Nevertheless, CS-JSCC retains

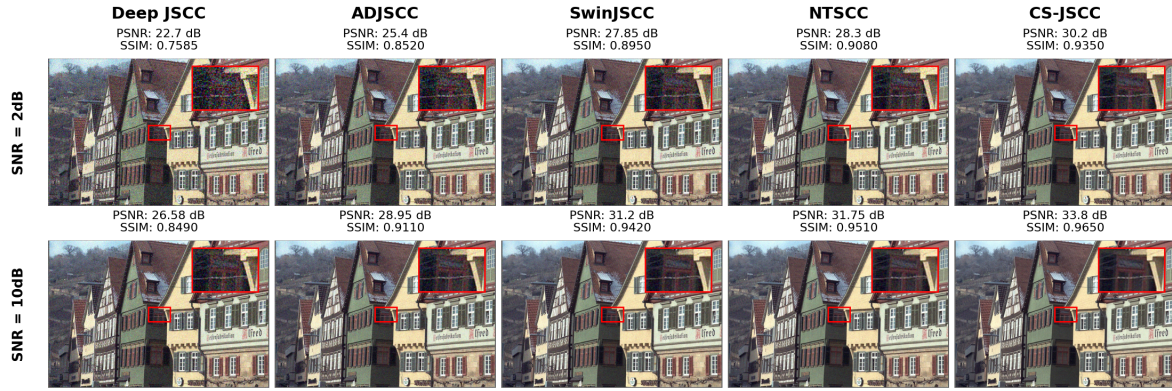


Figure 3.32: Examples of reconstructed images produced by the proposed CS-JSCC and the baseline methods for bandwidth compression ratio $R = 1/6$ and SNR values of 2 and 10 dB

finer structural details, particularly around window frames, roof boundaries, and other repetitive patterns, as highlighted in the enlarged regions.

As the SNR increases to 10 dB and 20 dB, all methods benefit from the improved channel conditions. However, differences remain visible in high-frequency regions. SwinJSCC and NTSCC generate sharper reconstructions than the earlier baselines, yet residual blur and slight texture degradation are still observable. In comparison, CS-JSCC produces cleaner reconstructions with sharper edges, better texture continuity, and an overall appearance that more closely resembles the ground truth.

From a quantitative perspective, CS-JSCC consistently achieves the highest PSNR and SSIM values across the evaluated SNR range. The improvement in SSIM is particularly important because it indicates better preservation of structural information and perceptual similarity. These results demonstrate that the proposed CS-based representation, together with adaptive sampling and deep reconstruction, provides measurable gains over both the earlier CNN-based JSCC methods and more recent transformer-based and nonlinear transform-based approaches.

3.6 Overview of the Proposed Fixed-Rate Compressive JSCC Frameworks

The proposed frameworks in this chapter share a common objective: to embed a CS-inspired sampling and reconstruction mechanism inside an end-to-end deep JSCC pipeline so that dimen-

sionality reduction and channel robustness are optimized jointly. However, the methods differ in where the main performance gains are obtained. BCS-JSCC establishes the core idea and provides a baseline design where block-wise learned sampling and two-stage reconstruction are trained together with the JSCC encoder and decoder. DCS-JSCC keeps the same sampling structure but strengthens the representation and reconstruction stages by introducing more effective normalization and a deeper refinement network, thereby improving robustness and reconstruction fidelity without increasing adaptivity overhead. CS-JSCC extends the design further by introducing controlled spatial adaptivity through multi-channel sampling, allowing the system to allocate measurements preferentially to informative regions while keeping a fixed average sampling budget. Since these models are introduced progressively, a compact cross-comparison helps us track which modules change, what capability is added at each stage, and what qualitative trade-offs arise in terms of complexity, adaptivity, and expected robustness.

Table 3.7 summarizes the differences at a level that complements the descriptions in Sections 3.3-3.5, and it also clarifies why the next chapter moves from fixed-rate designs toward explicit rate-adaptive mechanisms. Overall, the comparison highlights a clear progression: BCS-JSCC demonstrates the value of embedding learned sampling and reconstruction within JSCC, DCS-JSCC improves the stability and expressiveness of the fixed-rate pipeline, and CS-JSCC introduces principled spatial adaptivity to better match the measurement budget to image content. While all three frameworks operate at fixed average compression ratios, the final design suggests that further gains require mechanisms that adjust the effective rate more directly in response to bandwidth variability, which motivates the rate-adaptive architectures presented in Chapter 4.

Fig. 3.33 compares the envelope PSNR of BCS-JSCC, DCS-JSCC, and CS-JSCC on the Kodak dataset over an AWGN channel at a fixed compression ratio of 1/6. For each method, the envelope is computed by taking, at every SNR_{test} , the best PSNR achieved among the available training conditions, so that each framework is represented by a single best-case curve and the comparison remains uncluttered. The three envelopes follow the expected trend: reconstruction quality improves steadily as SNR_{test} increases, with diminishing gains at higher SNR values where the channel becomes less of a limiting factor. Across the full SNR range, CS-JSCC consistently yields the highest PSNR, indicating that its content-aware measurement allocation

Table 3.7: Qualitative comparison of BCS-JSCC, DCS-JSCC, and CS-JSCC.

What to look for	BCS-JSCC	DCS-JSCC	CS-JSCC
Encoder design			
Learned CS sampling integrated inside JSCC	✓	✓	✓
Sampling granularity	Block-wise	Block-wise	Block-wise
Sampling policy across blocks	Uniform	Uniform	Adaptive
Multi-channel sampling paths	×	×	✓
Block routing / saliency-based allocation	×	×	✓
Extra side information (per-block index)	×	×	✓
Rate-control handle in the design	Sampling ratio fixed	Sampling ratio fixed	Fixed avg. SR + block-wise selection
Latent shaping and channel robustness (JSCC core)			
CNN-based JSCC encoder/decoder	✓	✓	✓
GDN/IGDN normalization	×	✓	✓
Power normalization	✓	✓	✓
Reconstruction strategy (receiver)			
Initial reconstruction	✓	✓	✓
Deep reconstruction type	Shallow CNN	Residual CNN	Unrolled denoising-style
Designed to correct block artefacts	✓	✓	✓
Training supervision			
Single loss on final output	✓	×	×
Intermediate supervision (init + final)	×	✓	✓
Perceptual-fidelity loss (SSIM/ ℓ_1)	×	×	✓
Novelty			
Primary capability	Baseline compressive JSCC	Stronger latent + refinement	Spatially adaptive sampling under fixed budget

provides a clear advantage under a fixed average sampling budget. DCS-JSCC forms a strong middle curve, outperforming BCS-JSCC due to its more effective latent shaping and deeper refinement network, which better suppresses compression artefacts and residual channel noise. The relative gaps are most pronounced in the low-to-mid SNR region, where noise and limited channel reliability make the reconstruction task harder and architectural differences translate into larger fidelity gains. As SNR_{test} increases toward the high-SNR regime, all methods begin to plateau, and the performance differences narrow, reflecting that reconstruction quality is increasingly governed by the fixed compression constraint rather than channel impairment. Overall, the figure provides a compact performance-oriented complement to Table 3.7, confirming that the progressive design changes from BCS-JSCC to DCS-JSCC to CS-JSCC translate into a consistent improvement in reconstruction fidelity under the same rate budget.

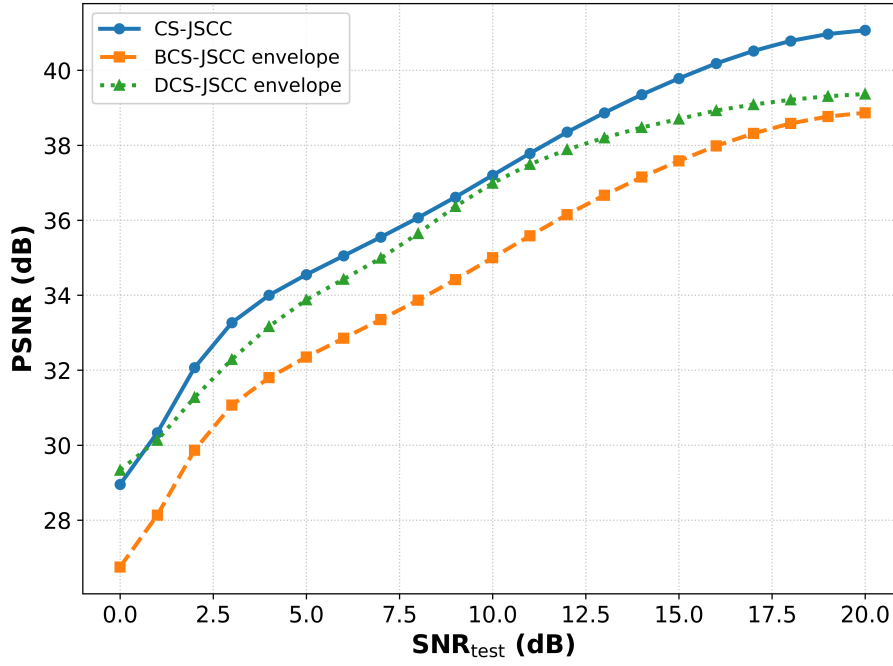


Figure 3.33: Envelope PSNR comparison of the suggested fixed-rate compressive JSCC frameworks on Kodak dataset over an AWGN channel at compression ratio $k/n = 1/6$.

3.7 Conclusive Remarks

This chapter has investigated a family of fixed-rate deep compressive JSCC frameworks for wireless image transmission, in which CS based modules are integrated as internal components of the encoder and decoder and optimized jointly with the communication system. Three architectures have been developed and evaluated in a progressive manner: BCS-JSCC, DCS-JSCC, and CS-JSCC:

- **BCS-JSCC:** The first framework, BCS-JSCC, establishes a baseline fixed-rate compressive JSCC design by integrating a block-based CS sampling module into the encoder and a two-stage reconstruction network into the decoder, enabling an end-to-end learned sensing and recovery process that improves bandwidth efficiency and noise robustness over Deep JSCC and ADJSCC.
- **DCS-JSCC:** Building on the foundation of BCS-JSCC, DCS-JSCC enhances the architecture through the use of GDN and IGDN layers for better latent normalization, a residual-based deep reconstruction network for stronger nonlinear refinement, and a dual-stage loss function that supervises both initial and final reconstructions, yielding consistent PSNR

and SSIM gains across SNR regimes.

- The most advanced design of this chapter, CS-JSCC, generalizes these ideas using a multi-channel adaptive BCS sampling module that allocates sampling rates according to visual saliency and a more sophisticated reconstruction pipeline combining improved blockwise recovery with an unrolled deep refinement network, trained with a hybrid SSIM and L1 objective. As a result, CS-JSCC achieves the highest reconstruction quality and strongest robustness among all proposed methods across AWGN and Rayleigh channels on CIFAR-10 and Kodak datasets.

Together, these designs demonstrate that appropriately structured compressive sampling and reconstruction can significantly enhance the performance and robustness of JSCC frameworks under bandwidth and SNR constraints. Beyond the individual performance gains, several general insights emerge from the comparative analysis in this chapter:

- First, integrating CS as a trainable component of the JSCC encoder enables more effective use of channel resources than treating compression and channel coding as separate stages.
- Second, the choice of normalization, reconstruction depth, and loss formulation has a direct impact on robustness to channel noise and on the ability to exploit improved SNR.
- Third, multi-channel and content adaptive sampling provide a practical mechanism for focusing the available measurements on informative regions, which improves reconstruction quality without increasing the average compression ratio.
- Finally, the experiments highlight the importance of jointly optimizing all components of the communication pipeline, from sampling to deep reconstruction, within a single end-to-end learning framework.

These findings validate the viability of deep compressive JSCC as a practical approach for bandwidth-constrained wireless image transmission and clarify the roles of sampling strategy, network architecture, and training objectives in shaping performance.

At the same time, all three frameworks operate at fixed average compression ratios, which limits their flexibility when channel bandwidth or application requirements vary over time. This

motivates the development of rate-adaptive designs that can adjust the sampling ratio and channel usage dynamically. Chapter 4 addresses this limitation by introducing two rate-adaptive JSCC architectures that extend the principles established in this chapter to support multiple operating rates within a single unified model.

Rate Adaptive Deep Compressive JSCC for Wireless Image Transmission

4.1 Introduction and Related Works

The fixed-rate deep compressive JSCC frameworks developed in Chapter 3 demonstrated that integrating learned CS modules into end-to-end JSCC pipelines can substantially improve reconstruction fidelity and robustness under bandwidth constraints. However, these architectures share an inherent limitation: the sampling ratio is fixed once the model is trained. In practice, wireless communication systems rarely operate under static conditions. Available bandwidth fluctuates due to various reasons, while SNR varies unpredictably over time and across users. As a result, practical deployments require rate-adaptive JSCC mechanisms capable of adjusting the number of transmitted symbols on demand, without retraining or maintaining multiple models. The objective of this chapter is to address this gap by developing deep compressive JSCC architectures that support rate control while retaining the robustness and efficiency established in Chapter 3.

Rate adaptation has long been a key requirement in wireless multimedia communication, and various approaches have been proposed to realize it. Early systems relied on progressive or layered transmission [33], [98], [99], [100], where a base layer captures coarse content and additional refinement layers are transmitted when extra bandwidth becomes available. Although

conceptually simple and inspired by scalable video/image coding, these schemes require multiple encoder passes, introduce design overhead, and provide only coarse granularity in rate adjustment. A parallel group of methods achieves rate flexibility using variable-length representations or multiple pre-trained JSCC models [101], [102], [103], [156], [157], [158], [159], where a controller selects among networks trained at different compression levels. While effective in stabilizing performance across bandwidth regimes, these methods incur heavy memory and computation costs, making them impractical for edge devices and do not exploit compressive structure within the latent representation.

Another line of research explores SNR-adaptive JSCC, conditioning the encoder or decoder on channel estimates [66], [105], [106], [107], [160], [161]. These designs use mechanisms such as feature-wise modulation, channel-adaptive attention, or explicit conditioning on noise variance to mitigate SNR mismatch. Although such models adapt well to varying noise levels, their latent dimensionality remains fixed. Consequently, they cannot adjust the actual transmission rate and therefore cannot satisfy bandwidth adaptivity requirements. Recent surveys, including [9], identify this lack of rate flexibility as a fundamental bottleneck in existing JSCC systems.

More advanced adaptive schemes utilize deeper learning-driven control policies. For instance, Gumbel-Softmax-based rate controllers learn to selectively activate groups of latent features depending on the source content and channel state, enabling multi-rate operation within a single model [162]. Similarly, entropy-aware feature pruning removes low-energy coefficients or patches to reduce transmission cost at high SNR, though at the expense of additional side information and complexity [163]. Other approaches utilize adaptive model selection frameworks, where multiple JSCC networks of varying complexity are executed based on available computational and channel resources [164], [165], [166]. Despite their adaptivity, these strategies remain orthogonal to CS and do not exploit the benefits of learned sampling as a dimension-reduction front end.

Recent advances in transformer-based JSCC introduce a different form of adaptivity tied to the self-attention mechanism, particularly in multiple-input and multiple-output (MIMO) systems [23], [60], [111]. Vision-transformer JSCC models use auxiliary channel-state heatmaps and attention to implicitly allocate representational capacity across spatial regions or sub-channels,

thereby improving robustness [167], [168]. Nevertheless, such architectures still output fixed-size latent vectors and thus cannot directly address rate flexibility.

Building on these observations, this chapter develops a family of rate-adaptive deep compressive JSCC frameworks that integrate learned compressive sampling with rate control in a single end-to-end trainable pipeline. The shared objective is to remove the fixed-latent bottleneck of conventional JSCC by enabling the transmitter to vary the number of channel symbols on demand, using channel state information (in the form of SNR feedback) together with source-dependent statistics, while preserving the robustness benefits of joint optimization. To this end, both designs employ learned sampling operators and content-aware rate selection to adjust the measurement density across images (and, when applicable, across blocks), and both rely on reconstruction networks that remain well-behaved over a range of sampling ratios without retraining. Despite this common goal, the two proposed frameworks differ in how rate adaptivity is realised and how tightly the components are coupled. The first framework, named RACS-JSCC, adopts a modular, blockwise strategy in which each block is routed to one of a small number of predefined sampling branches (e.g., low, medium, high), enabling simple discrete rate control with minimal architectural overhead. The second framework, titled URC-JSCC, provides a unified and expressive design by combining hierarchical learned sampling with a hybrid CNN-Transformer encoder-decoder backbone and an explicit rate regularisation term, allowing the operating point to be shaped directly through a distortion-rate trade-off during training. Together, these two designs establish a coherent progression in this chapter: from a lightweight multi-branch rate-adaptive extension of fixed-rate compressive JSCC to a fully unified architecture that jointly controls sampling, encoding, and reconstruction under variable bandwidth and SNR.

To provide a clear and structured presentation, the remainder of this chapter is organized as follows. Section 4.2 introduces the common system model and notation used across both suggested rate-adaptive JSCC frameworks. Section 4.3 describes the RACS-JSCC architecture as rate-adaptive compressive JSCC algorithm and its results. Section 4.4 presents the details of URC-JSCC design and corresponding evaluation. Section 4.5 discusses the comparison between the designed framework, and Section 4.6 summarizes the insights gained from these studies.

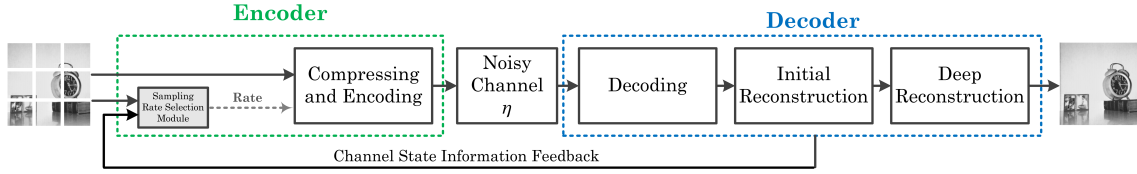


Figure 4.1: Components of the system model for rate-adaptive JSCC frameworks

4.2 System Model

The system model considered in this chapter is a point-to-point wireless image transmission system, as illustrated in Fig. 4.1. An input image of size $H \times W \times C$ (height, width, and number of colour channels) is represented by a vector $\mathbf{x} \in \mathbb{R}^n$, where $n = H \times W \times C$ and $\mathbb{R}$ denotes the set of real numbers. The joint source-channel encoder maps $\mathbf{x}$ to a complex-valued channel input vector and transmits it over a wireless channel. The receiver then reconstructs an estimate of the original image.

The encoder in this chapter consists of two components: (i) a sampling-rate selection network and (ii) a JSCC encoding network. The sampling-rate selection network determines the sampling configuration based on the available CSI and the input image statistics. In particular, CSI is estimated at the receiver and fed back in the form of the channel SNR. After selecting the sampling rate, the JSCC encoding network is represented by an encoding function $f(\cdot; \theta) : \mathbb{R}^n \rightarrow \mathbb{C}^k$, which maps the real-valued image vector $\mathbf{x}$ to a complex-valued channel input vector $\mathbf{z} \in \mathbb{C}^k$:

$$\mathbf{z} = f(\mathbf{x}; \theta) \in \mathbb{C}^k \quad (4.1)$$

where k denotes the number of transmitted channel symbols and θ denotes the set of trainable parameters of the encoder. In the proposed rate-adaptive schemes, the effective number of transmitted symbols may vary with the selected sampling configuration. In practice, a maximum tensor length $k_{\max}$ is set for minibatch processing, but the transmitter uses only the first $k \leq k_{\max}$ symbols selected by the sampling configuration. The remaining entries are zero-padded and ignored via a binary mask, so they are neither transmitted nor included in the power/rate accounting.

The wireless channel is modelled as a mapping $\eta : \mathbb{C}^k \rightarrow \mathbb{C}^k$. In this chapter, the AWGN

and slow Rayleigh fading models described in Eqs. (3.3), (3.4), and (3.5) are considered. The received signal is therefore given by

$$\hat{\mathbf{z}} = \eta(\mathbf{z}) = \eta(f(\mathbf{x}; \theta)) \in \mathbb{C}^k \quad (4.2)$$

At the receiver, the decoder is implemented as a two-stage reconstruction pipeline that is compatible with both rate-adaptive frameworks proposed in this chapter. Specifically, the receiver first applies a decoding network that maps the noisy channel symbols to an intermediate real-valued representation. This decoding function is defined as $g(\cdot; \varphi) : \mathbb{C}^k \rightarrow \mathbb{R}^m$:

$$\tilde{\mathbf{x}} = g(\hat{\mathbf{z}}; \varphi) = g(\eta(f(\mathbf{x}; \theta)); \varphi) \in \mathbb{R}^m \quad (4.3)$$

where φ denotes the parameters of the decoding network and m denotes the dimension of the intermediate representation. In the proposed rate-adaptive schemes, m depends on the selected sampling configuration and therefore may vary across images (and across blocks in blockwise rate selection).

Given $\tilde{\mathbf{x}}$, the receiver constructs an initial reconstruction of the image using an initial reconstruction network $h(\cdot; \psi) : \mathbb{R}^m \rightarrow \mathbb{R}^n$:

$$\hat{\mathbf{x}}_{\text{init}} = h(\tilde{\mathbf{x}}; \psi) \in \mathbb{R}^n \quad (4.4)$$

where ψ denotes the parameters of the initial reconstruction network. Finally, a deep reconstruction network refines $\hat{\mathbf{x}}_{\text{init}}$ to obtain the final reconstructed image. This refinement function is defined as $d(\cdot; \chi) : \mathbb{R}^n \rightarrow \mathbb{R}^n$:

$$\hat{\mathbf{x}} = d(\hat{\mathbf{x}}_{\text{init}}; \chi) \in \mathbb{R}^n \quad (4.5)$$

where χ denotes the parameters of the deep reconstruction network. Combining (4.3)-(4.5) yields the overall end-to-end mapping:

$$\hat{\mathbf{x}} = d(h(g(\eta(f(\mathbf{x}; \theta)); \varphi); \psi); \chi) \quad (4.6)$$

This system model is utilized for both developed rate-adaptive compressive JSCC pipelines in this chapter.

4.3 RACS-JSCC: Rate Adaptive Compressive Sampling for Image JSCC

In this section, the architecture and performance of RACS-JSCC as a novel rate-adaptive compressive JSCC framework is presented. The main objective of RACS-JSCC is to introduce explicit blockwise rate control on top of the fixed-rate BCS-JSCC pipeline in Section 3.3, so that the number of transmitted measurements can be adjusted within a single trained model. In particular, RACS-JSCC replaces the single-rate CS sampling module with a rate-selection one coupled to multiple rate-specific CS samplers: for each block, the selector uses block content and SNR to choose a sampling ratio, and the corresponding sampler generates the compressed measurements. This mechanism enables the encoder to allocate fewer measurements to visually redundant or smooth regions, while reserving higher sampling rates for blocks with edges, textures, or fine structures that are more sensitive to distortion. The selected sampling ratio activates a corresponding sampling branch with shared convolutional sampling operators, which directly controls how many compressed measurements are produced for that block. The resulting measurements are then processed by a common JSCC encoder and decoder that are trained to operate reliably across the supported range of sampling ratios. This design establishes a baseline rate-adaptive architecture for the chapter in which rate flexibility is achieved through discrete, per-block sampling decisions, while the following URC-JSCC framework builds on the same objective with a more unified rate-controlled design and stronger coupling between rate selection, encoding, and reconstruction.

4.3.1 Model Architecture

The proposed RACS-JSCC framework introduces a strategy to optimize the image transmission process by dynamically adjusting the sampling rate using a CNN and an attention-based mechanism. In particular, visually less important or redundant regions are assigned lower sampling

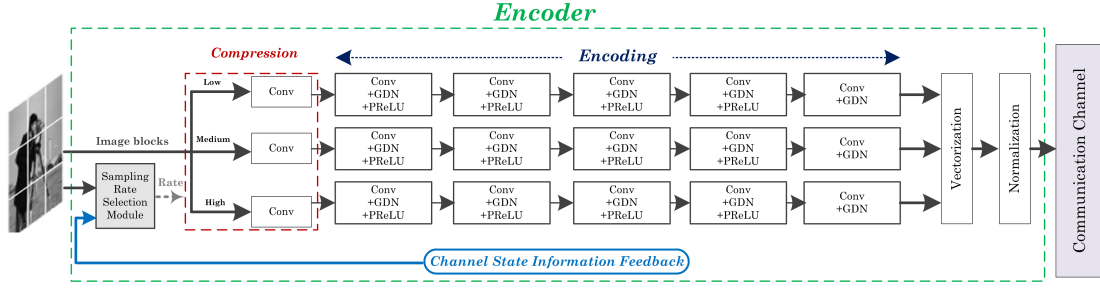


Figure 4.2: Architecture of the proposed RACS-JSCC encoder.

rates (stronger compression), while blocks containing edges, textures, or salient structures are sampled at higher rates to preserve essential details. The design details of the RACS-JSCC encoder and decoder are presented as follows.

Encoder Design

The overall architecture of the RACS-JSCC encoder is shown in Fig. 4.2. The encoding process begins by partitioning the image into non-overlapping blocks of size $B \times B \times C$, where C represents the number of colour channels and B denotes the block size. The image blocks are then input into the sampling rate selection module where both global and local features of each image block are calculated. The sampling rate is defined as the ratio of the number of compressed measurements generated by the compression module over the total number of pixels in the input block. Both the extracted features and the SNR of the communication channel are then used to determine the sampling rate for each image block. Once the optimal rate for the block is selected, the block is compressed and encoded by the corresponding branch of the encoder.

The sampling rate selection module utilizes a CNN architecture to extract a variety of global and local features from the input image. Here, global features refer to block-level statistics such as overall contrast or low-frequency energy, while local features capture fine-scale structure such as edge density or high-frequency texture energy; for example, a smooth sky block has low edge/texture energy, whereas a block containing object boundaries has high edge density and stronger high-frequency content. The CNN is followed by a combination of fully connected layers and attention mechanisms which combine the image features and the CSI to determine the optimal sampling rate for each block. Specifically, the sampling rate selection process involves the following key steps:

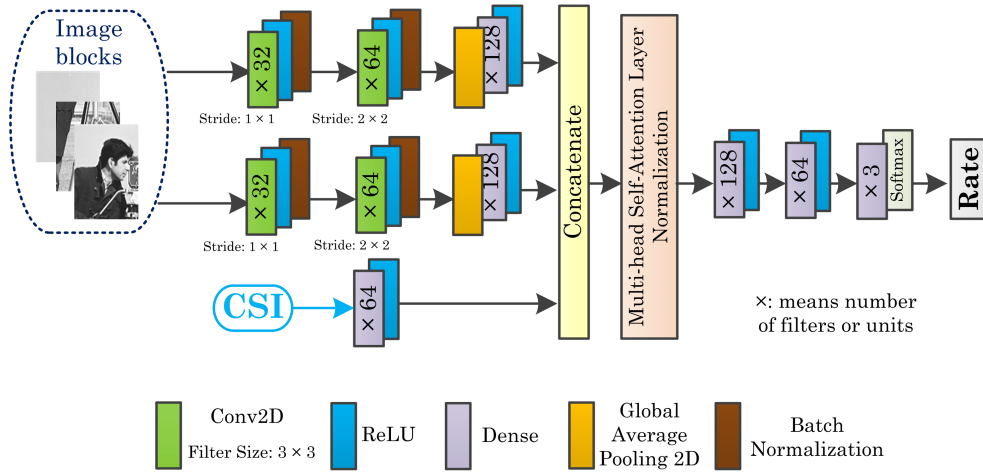


Figure 4.3: Architecture of the sampling rate selection module.

- **Feature fusion:** Global and local features are combined to form a comprehensive feature representation for each image block. This fusion captures both the broader context and fine details necessary for optimal sampling.
- **Attention mechanisms:** Attention layers dynamically weigh the importance of different features, allowing the model to focus on the most relevant aspects when deciding the sampling rate for each block. This ensures that critical regions are sampled more finely. It means that the block is assigned to a higher sampling-rate branch, which produces a larger number of CS measurements for that block, so less information is discarded, and the decoder receives a richer measurement vector for reconstruction.
- **Rate decision:** The module computes a sampling rate R for each block, based on the fused features and attention weights, as well as the channel SNR. The rate is chosen to balance the trade-off between compression efficiency and image quality, taking into account the current channel conditions and the block's importance.

The CNN used for feature extraction consists of several convolutional and pooling layers, followed by batch normalization and activation functions to capture and process the image features effectively. Attention mechanisms are implemented using multi-head self-attention layers that provide a robust method for focusing on different parts of the feature space. The architecture details of this module are shown in Fig. 4.3.

The rate selection process involves directing each block to one of three sampling modules

with high, medium and low sampling rates (Fig. 4.2). The high-rate module samples input blocks at a high rate when the blocks contain critical details or have high importance based on the extracted features, and when the channel conditions are good (i.e., high SNR). Conversely, the low-rate sampling module samples input blocks at lower sampling rates when image blocks have few details maximizing compression without significantly affecting perceived image quality, or when the channel noise is high and higher resilience against transmission errors is required at the expense of compression. In implementation, the rate-selection network outputs three scores (logits) per block, which are converted into a probability distribution over high, medium and low sampling rates using a softmax; a hard rate decision is then obtained by selecting the maximum-probability branch for routing.

This adaptive compression process shares a close theoretical relation with classical sub-band analysis and transform-based bit allocation. In traditional subband coding, an image is decomposed into distinct frequency bands using filter banks so that channel capacity can be prioritized toward high-energy or perceptually vital subbands [169]. In RACS-JSCC, the convolutional layers within the sampling-rate selection module function as a data-driven, spatial filter bank. By analyzing local spatial frequencies, edge densities, and texture complexities, this module implicitly segments the image into low-frequency regions (routed to the low-sampling branch to maximize compression) and high-frequency structural details (routed to the medium- or high-sampling branches to safeguard against distortion). While classical subband analysis performs this separation explicitly via fixed analytical transforms, RACS-JSCC unifies feature decomposition and rate selection within a learned, end-to-end framework optimized directly for continuous wireless channel impairments.

Once the sampling rate decision is made, the image blocks are processed through their respective compressing modules. Each module compresses the image blocks according to the sampling rates defined in the previous stage. A block-based CS sampling network, outlined in [73], generates compressed measurements using a sampling matrix Φ_B . The sampling process of the j -th block can be mathematically expressed as $\mathbf{y}_j = \Phi_B \mathbf{x}_j$, meaning that each row of the measurement matrix Φ_B acts as a filter. To implement the described sampling process using a neural network module, a convolutional layer is employed, with filter dimensions matching

the size of image blocks. For non-overlapping sampling, the convolutional layer adopts a stride of $B \times B$. No biases or activation functions are used. In essence, the output is feature maps, with each column of output encapsulating measurements originating from an image block. The learning process involves optimizing the sampling matrix alongside other network parameters through end-to-end training, as detailed in subsequent sections.

Following the sampling network, data flow proceeds through a series of convolution layers, PReLU activation functions, and a GDN layer, which constitute the encoding structure. These layers, except for the power normalization layer, are organized into five modules. Each of the first four modules comprises a convolution layer, a GDN layer, and a PReLU layer, while the fifth module consists of only a convolution layer and a GDN layer. This sequence of convolution layers, shown in Fig. 4.2, extracts essential features from the compressed image, which are then combined to produce the channel input samples. The incorporation of nonlinear activation functions, such as PReLU, plays a crucial role in learning a nonlinear mapping from the source signal space to the coded space, enabling the network to capture complex relationships within the data. Also, the GDN layer employs local divisive normalization, which is highly effective for tasks like image compression by capturing statistical dependencies within the image. The details of the implementation of these layers can be found in BCS-JSCC and DCS-JSCC information (Sections 3.3 and 3.4).

After compressing and encoding the image blocks, they are concatenated into a single vector for transmission over the channel. To ensure that the decoder can correctly identify and reconstruct each part of the transmitted vector, each encoder output is labeled based on its respective sampling rate. We employ three distinct labels corresponding to the three different sampling rates: high, medium, and low. These labels are appended to the encoded outputs, thus providing the decoder with the necessary information to identify the sampling rates used for each block. This enables the decoder to apply the appropriate reconstruction algorithm, ensuring accurate image reconstruction.

As the final step, the output of the encoder, which is a vector comprised of compressed encoded image blocks, is subjected to a normalization process as mentioned in Eq. (3.7) where the output is referred to as $\mathbf{z}$. After encoding, the joint source-channel coded sequence is

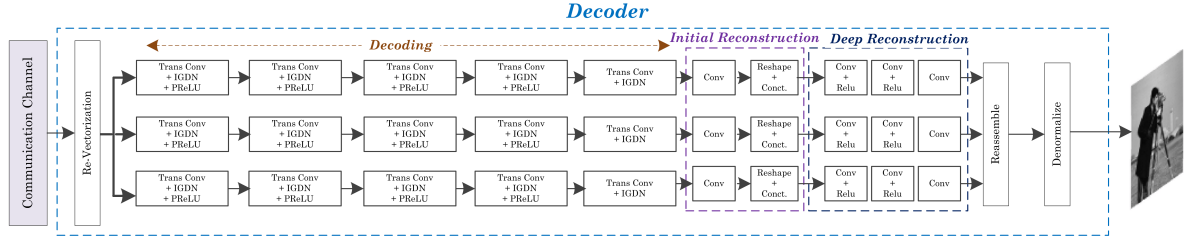


Figure 4.4: Architecture of the proposed RACS-JSCC decoder.

transmitted over the wireless communication channel, which is included in the architecture as a non-trainable layer, represented by the transfer function Eq. (3.5).

Decoder Design

As shown in Fig. 4.4, the first decoding step involves reversing the vectorization process performed during encoding. This requires identifying and separating each encoded sub-vector based on the labels attached during the encoding phase. These labels, which indicate the sampling rates used for each sub-vector, guide the decoder in directing each sub-vector to the appropriate decoding module. The decoder processes each sub-vector according to its label, utilizing different convolutional layers and deep learning models specifically tailored for the corresponding sampling rate.

Following the vector separation process, the decoding network consists of three distinct decoders, each of which operates in three consecutive stages: decoding, initial reconstruction, and deep reconstruction. In the first stage, the decoder aims to map the corrupted and compressed complex-valued signals back to an estimate of the original channel input. This stage effectively reverses the operations performed during the encoding process. The received corrupted coded inputs are processed through a sequence of transpose convolutional layers, which include PReLU activation functions and IGDN. The details of these layers can be found in BCS-JSCC and DCS-JSCC information (Sections 3.3 and 3.4). These layers work together to decode the noisy signals and perform an initial recovery, mitigating the effects of compression and noise introduced during transmission.

The reconstruction network comprises initial and deep reconstruction networks, ensuring accurate recovery of the original image from the CS-encoded measurements. For the initial reconstruction stage, similar to the compressive sampling process, CB^2 convolutional filters

with kernel size $1 \times 1 \times m_r$ and stride 1×1 are applied to generate each initial reconstructed block which m_r denotes the number of CS measurements produced for that block under branch. Subsequently, a combination layer is utilized, comprising a reshape function and a concatenation function, to obtain the initial reconstructed image. This layer first reshapes each $1 \times 1 \times CB^2$ reconstructed vector into a $B \times B \times C$ block, followed by concatenating all blocks to form the initial reconstructed image. This initial phase allows for the reconstruction of the entire image rather than individual blocks, enabling comprehensive utilization of both intra-block and inter-block information for improved reconstruction. Since there is no activation layer in the initial reconstruction network, it functions as a linear signal reconstruction network.

The initial reconstruction is followed by a non-linear reconstruction process which further improves the quality of the reconstructed image. We use a deep reconstruction sub-network, which realises the non-linear reconstruction process. The deep reconstruction sub-network contains m layers where all the layers but the first and the last ones are of the same type: d filters of size $f \times f \times d$ where a filter operates on a $f \times f$ spatial region across d channels (feature maps). The first layer of the deep reconstruction sub-network operates on the initial reconstructed output so that it has d filters of size $f \times f \times 3$. The last layer outputs the final image estimation and consists of a single filter of size $f \times f \times d$. In the experiments, d and f are set to $d = 64$ and $f = 3$. Furthermore, ReLU is utilized as an activation function after each convolution layer in the deep reconstruction sub-network. ReLU is appropriate in here because the network mainly learns sparse residual feature responses associated with edges, textures, and local artefact correction. Compared with the encoder bottleneck, preserving signed channel features is less critical at this stage, while ReLU provides a simple, stable, and computationally efficient nonlinearity for reconstruction refinement.

4.3.2 Loss Function

The proposed encoder and decoder networks are optimized jointly in an end-to-end manner. Given the input image $\mathbf{x}$, the goal is to obtain a highly compressed encoded measurement with the encoder, and then recover the original input image $\mathbf{x}$ from its noisy version with the decoder network. Since the encoder, decoder and communication channel form an end-to-end network,

they can be trained jointly. To do so, MSE is adopted as the cost function of the proposed RACS-JSCC network. The optimization objective is formulated as:

$$\min_{\theta, \varphi, \psi, \chi} \frac{1}{N} \sum_{i=1}^N \|d_{\chi}(h_{\psi}(g_{\varphi}(\eta(f_{\theta}(\mathbf{x}_i)))) - \mathbf{x}_i\|_2^2 \quad (4.7)$$

where N represents the number of samples or data points in the dataset, θ is the parameter set of the encoder network, $\{\varphi, \psi, \chi\}$ is the set of the parameters of the decoder network, and $d_{\chi}(h_{\psi}(g_{\varphi}(\eta(f_{\theta}(\mathbf{x}_i))))$ is the final reconstructed output $\hat{\mathbf{x}}$. It should be noted that the encoder and the decoder networks are trained jointly, but they can be utilized independently.

4.3.3 Results and Discussion

The proposed model is implemented using TensorFlow and optimized by the Adam algorithm. The performance of the RACS-JSCC algorithm is evaluated based on PSNR of the reconstructed images. To train the proposed RACS-JSCC architecture, CIFAR-10 and Imagenet datasets are used. Once trained, our model is tested on CIFAR-10 and Kodak datasets, and the results are compared with baseline JSCC methods Deep JSCC [6] and ADJSCC [51]. To achieve robustness to varying channel conditions, RACS-JSCC model is trained for SNRs ranging from 0 to 20 dB. This training strategy ensures that the model is exposed to a wide spectrum of noise conditions, enabling it to perform effectively under different scenarios eliminating the need for training a separate model for each SNR value. During each training epoch, the model randomly samples a batch of data with varying SNR levels. This approach allows the model to generalize better across different channel conditions, rather than being optimized for a specific SNR regime.

It is worth noting that both Deep JSCC and ADJSCC are designed to operate for a fixed compression ratio. Thus, the number of channel input symbols for both benchmark algorithms is always exactly k in all experiments. In contrast, proposed RACS-JSCC algorithm is rate-adaptive. Hence, for comparison purposes, RACS-JSCC is allowed to generate at most k channel input symbols, meaning that in practice due to the adaptive CS sampling, the number of actual channel input symbols may be less than k .

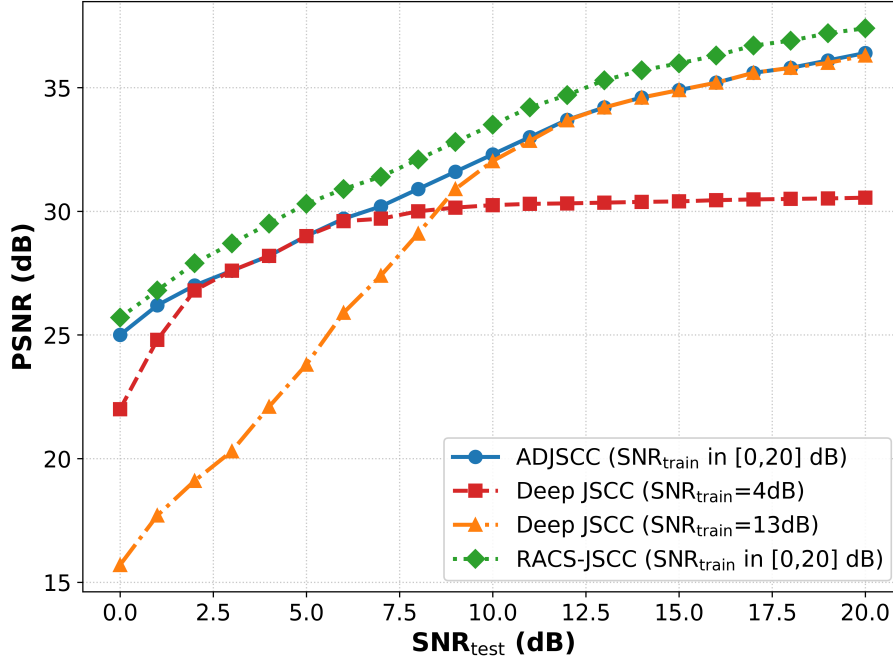


Figure 4.5: Performance of the proposed RACS-JSCC versus varying SNR_{test} over an AWGN channel on CIFAR-10 dataset (compression ratio $k/n = 1/6$)

Evaluation on CIFAR-10 Dataset

The CIFAR-10 training dataset is firstly considered for evaluation purposes. To assess the performance of RACS-JSCC algorithm, a separate set of 10,000 images from the CIFAR-10 dataset, distinct from those used for training is tested. Initially, a learning rate of 10^{-3} is utilized, which is then reduced to 10^{-4} after 500,000 iterations. Training is performed using mini-batches, each containing 64 samples until there is no further improvement of the loss function observed on the test dataset. In the experiments for this dataset, size of blocks are considered as $B = 8$. It is worth mentioning that the test set images are not utilized for tuning the network hyperparameters. To address the impact of channel-induced randomness during performance evaluation, each image is transmitted 10 times. Also, the sampling ratios for low, medium and high branches of the designed encoder are selected as $1/16$, $1/8$ and $1/4$, respectively.

The first evaluation study assesses the performance of the proposed algorithm for an AWGN channel by varying the channel SNR. Fig. 4.5 illustrates the PSNR of the reconstructed images plotted against the SNR of the channel for a compression ratio $k/n = 1/6$. This evaluation demonstrates the effectiveness of the proposed RACS-JSCC method when optimized over a broad SNR range and tested under different channel conditions (SNR_{test}). It can be seen that

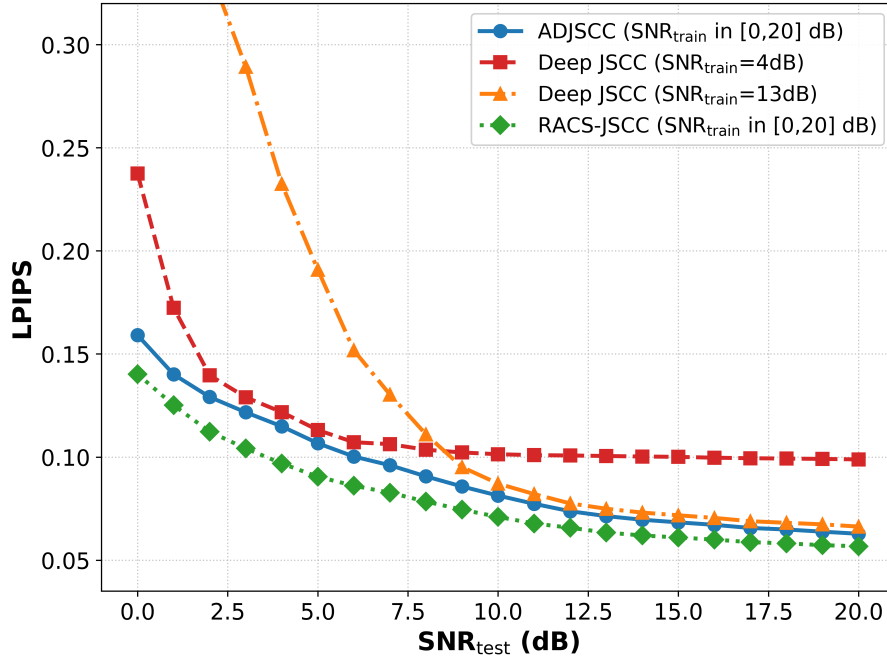


Figure 4.6: LPIPS comparison of RACS-JSCC and baseline methods over the AWGN channel on the CIFAR dataset.

RACS-JSCC is robust to changes in channel quality and can operate under varying channel conditions. Furthermore, the results reveal that the proposed method consistently outperforms the Deep JSCC and ADJSCC algorithms. Both the proposed method and ADJSCC exhibit adaptability to changing SNR levels, as indicated by their gradual performance degradation with decreasing SNR. However, RACS-JSCC surpasses ADJSCC, demonstrating better performance as SNR_{test} increases. This highlights the superiority of the proposed method in maintaining higher image quality under varying channel conditions while transmitting less or at most the same number of encoded symbols as the benchmark methods, thus utilizing less bandwidth resources.

To further evaluate perceptual reconstruction quality, LPIPS is reported for the RACS-JSCC experiments in addition to PSNR and SSIM. PSNR measures pixel-wise distortion, but it does not fully capture perceptual sharpness or texture consistency. Recent JSCC studies have therefore used LPIPS as a complementary perceptual metric, where lower values indicate greater perceptual similarity [147]. Fig. 4.6 shows the LPIPS comparison on the CIFAR-10 dataset over the AWGN channel. The LPIPS values decrease as the test SNR increases, which is consistent with the PSNR trend. RACS-JSCC achieves lower LPIPS than the fixed-SNR Deep JSCC baselines

and ADJSCC across most SNR values, indicating that the proposed rate-adaptive CS sampling strategy preserves perceptually relevant image content more effectively.

Evaluation on Kodak Dataset

To demonstrate the validity of the RACS-JSCC algorithm, the model is further trained on the Imagenet dataset. The input images are cropped to produce patches with dimensions of 224×224 . These patches are subsequently processed by the network in batches of 32 samples. In this series of experiments, the size of blocks is considered as $B = 32$. The training process is carried out until convergence is attained, with the model learning rate fixed at 10^{-4} . Values of $\text{SNR}_{\text{train}}$ ranging from 0 dB to 20 dB are utilized to train the model on this dataset. To do this, the dataset is divided into training and validation groups using a 9 : 1 ratio. The Kodak dataset is then used to evaluate the model, and throughout this procedure, each image is transmitted 100 times, allowing the performance to be averaged over several random channel instances.

Fig. 4.7 illustrates the comparison of average PSNR against SNR for a compression ratio of 1/6 with Deep JSCC and ADJSCC on the AWGN channel. The performance depicted in Fig. 4.7 reveals that RACS-JSCC surpasses Deep JSCC and ADJSCC by preserving essential visual details in compressed images, leading to enhanced reconstruction quality. These results indicate consistently elevated PSNR values across different SNR levels, emphasizing improved image fidelity and reduced channel-induced distortions. The proposed method demonstrates excellence in delivering superior image quality, even at higher compression levels and under noisy conditions.

The perceptual evaluation is also conducted on the Kodak dataset, as shown in Fig. 4.8. Kodak images contain richer textures and higher-resolution spatial details than CIFAR-10, so LPIPS provides useful additional information beyond distortion-based metrics. The results show that LPIPS decreases with increasing SNR for all methods. The fixed-SNR Deep JSCC model trained at 4 dB saturates at higher test SNR values, which is reflected by a slower LPIPS improvement. In contrast, RACS-JSCC maintains a lower perceptual distance over the full SNR range, showing that adaptive block-wise sampling improves both reconstruction fidelity and perceptual image quality.

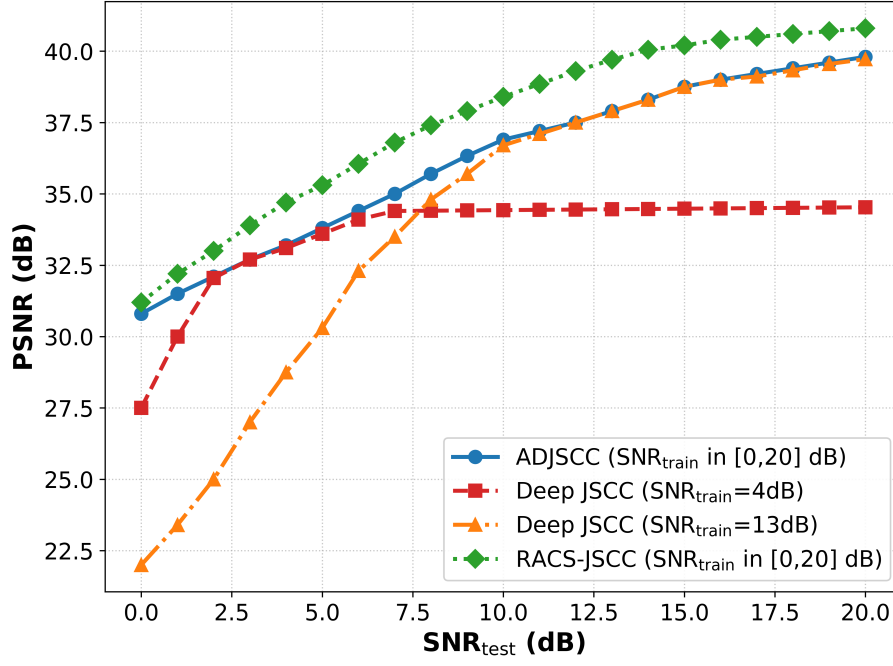


Figure 4.7: Performance of the proposed RACS-JSCC versus varying SNR_{test} over an AWGN channel on Kodak dataset (compression ratio $k/n = 1/6$)

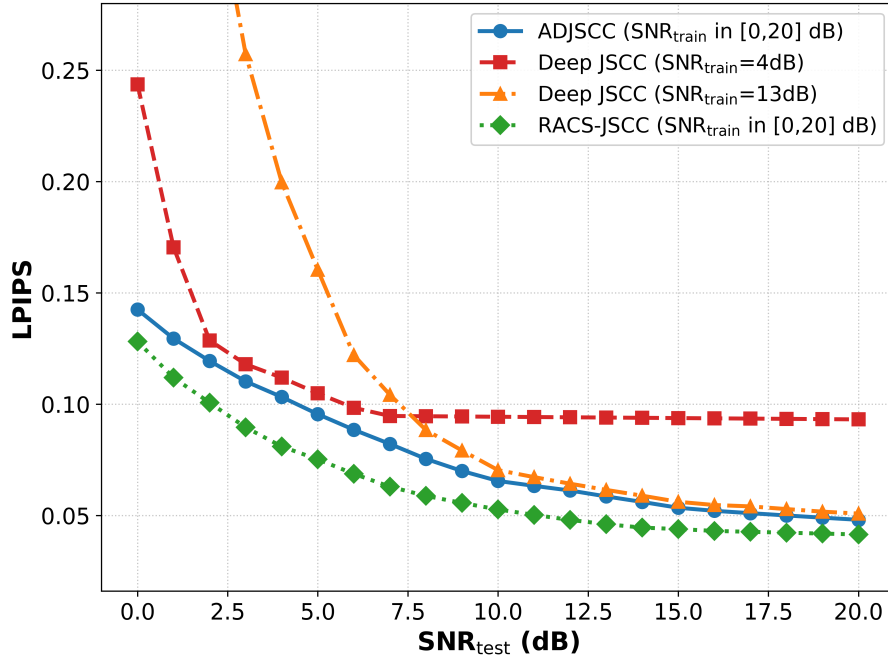


Figure 4.8: LPIPS comparison of RACS-JSCC and baseline methods over the AWGN channel on the Kodak dataset.

Analysis of the Sampling Rate Selection Module Behaviour

To further examine the behaviour of the proposed sampling rate selection module, Figs. 4.9 and 4.10 illustrate examples of the sampling decisions produced by RACS-JSCC under AWGN and

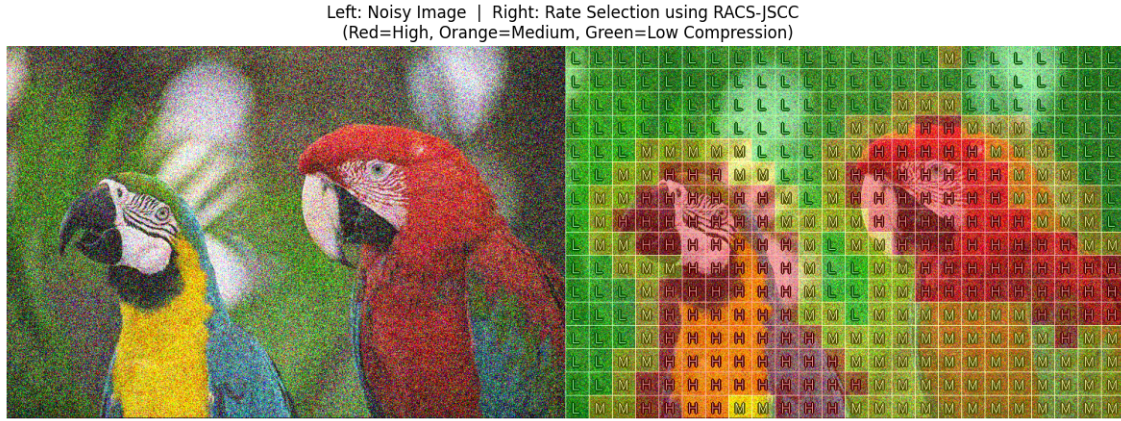


Figure 4.9: Example of block-wise sampling rate selection produced by RACS-JSCC encoder under a AWGN channel at $\text{SNR}_{\text{test}}=1\text{dB}$. The left image is the corrupted (noisy) input, and the right image visualizes the rate selection map from the RACS-JSCC sampling rate selection module. Each colour on rate map denotes one of the three sampling branches: high (red), medium (yellow), and low (green) rate.

Rayleigh fading channels at a $\text{SNR}_{\text{test}}=1\text{dB}$. In these examples, the input image is divided into 32 non-overlapping blocks, and for each block the encoder selects one of the three available sampling branches, high, medium, or low rate, based on the fused global-local feature representation and the channel condition. The resulting rate map, shown on right side of Figs. 4.9 and 4.10 provides insight into how the model distributes sampling resources spatially when operating under challenging fading and low-SNR conditions.

For the AWGN example in Fig. 4.9, the selected rate map shows a clear content-aware behaviour: blocks containing strong edges, textures, or fine structures are assigned medium or high sampling rates, while smoother background regions are predominantly assigned the low-rate branch. For instance, the blocks covering the bird's eye and body receive higher sampling density than the surrounding blurred background, reflecting that these regions require more measurements to preserve structural detail. This behaviour indicates that the rate-selection module avoids spending measurements on visually redundant areas and instead prioritizes regions that are more sensitive to distortion. A similar qualitative behaviour is observed for the Rayleigh fading example in Fig. 4.10. Blocks that contain prominent object boundaries and shadow transitions (e.g., around the hat and shadow regions) are more frequently mapped to medium/high-rate branches, whereas flatter regions are mapped to the low-rate branch. Overall, these examples confirm that RACS-JSCC produces spatially non-uniform measurement allocation

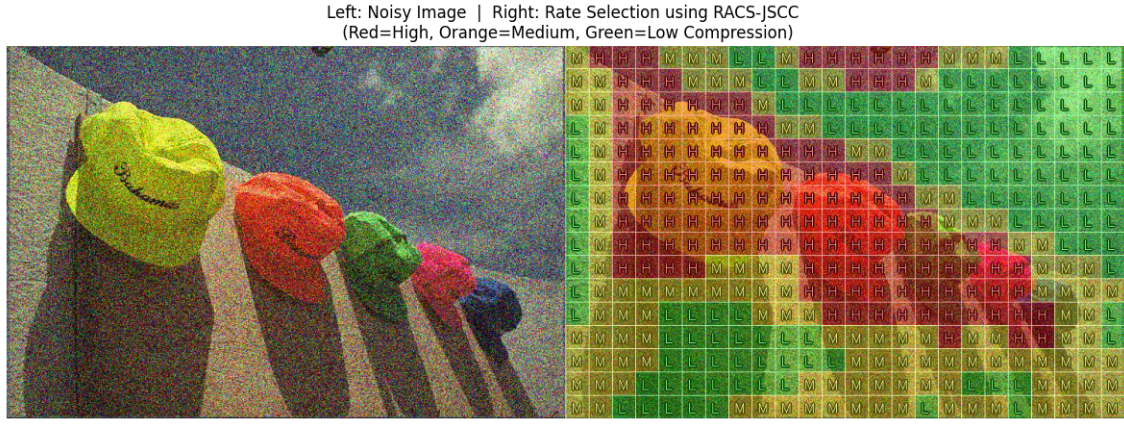


Figure 4.10: Example of block-wise sampling rate selection produced by RACS-JSCC encoder under a Rayleigh fading channel at $\text{SNR}_{\text{test}}=1\text{dB}$. The left image is the corrupted (noisy) input, and the right image visualizes the rate selection map from the RACS-JSCC sampling rate selection module. Each colour on rate map denotes one of the three sampling branches: high (red), medium (yellow), and low (green) rate.

that is consistent with local image complexity at low SNR.

It must be noted that from an architectural standpoint, evaluating the system explicitly without the sampling-rate selection module collapses the framework into a uniform fixed-rate pipeline, which is functionally equivalent to the baseline BCS-JSCC model presented in Section 3.3. Bypassing this rate-selection network eliminates the encoder’s capacity for spatial adaptivity, forcing the system to distribute measurements uniformly across all image blocks regardless of localized feature importance or instantaneous channel quality.

Ablation Study on Sampling Rate Selection Module Performance

To quantitatively analyze the performance isolation of the block-wise rate adaptivity mechanism, an ablation study was conducted on the CIFAR-10 test set over an AWGN channel. The full RACS-JSCC model was compared directly against three static configurations representing the network operating completely without the rate selection module active:

- Low-only Branch: All image blocks are uniformly fixed to a sampling ratio of $1/16$ (0.0625).
- Medium-only Branch: All image blocks are uniformly fixed to a sampling ratio of $1/8$ (0.1250) .

Table 4.1: Ablation study evaluating the impact of the sampling-rate selection module in RACS-JSCC on the CIFAR-10 test set over an AWGN channel.

Routing Configuration	PSNR (dB) @ 4 dB	PSNR (dB) @ 12 dB	$\bar{\alpha}$
Low-only (1/16)	25.12	27.85	0.0625
Medium-only (1/8)	26.90	30.15	0.1250
High-only (1/4)	28.24	33.10	0.2500
RACS-JSCC	28.15	33.18	0.1545

- **High-only Branch:** All image blocks are uniformly fixed to a sampling ratio of 1/4 (0.2500).

All configurations were evaluated across two representative channel noise levels ($\text{SNR}_{\text{test}} = 4$ dB and 12 dB) for a maximum budget constraint of $k/n = 1/6$. Table 4.1 outlines the reconstruction performance (PSNR) alongside the achieved average sampling ratio ($\bar{\alpha}$).

The results shown in Table 4.1 confirm that operating without the rate selection mechanism forces an inefficient distribution of communication resources. The Low-only and Medium-only configurations remain bandwidth-economical but suffer from severe information loss in high-frequency regions, bounding their maximum achievable PSNR. Conversely, while the High-only configuration reconstructs fine details effectively, it requires a fixed sampling ratio of 0.2500, over-allocating measurements to low-detailed or flat background regions.

By activating the adaptive sampling-rate selection module, RACS-JSCC achieves parity with or slightly exceeds the high-fidelity reconstruction numbers of the High-only configuration (28.15 dB vs. 28.24 dB at 4 dB SNR, and 33.18 dB vs. 33.10 dB at 12 dB SNR). Crucially, it secures these gains while dropping the average sampling ratio to 0.1545. This represents a 38.2% savings in measurement resource allocation over the non-adaptive high-rate pipeline, validating that context-aware block multiplexing effectively minimizes transmission overhead without degrading pixel-level structural accuracy.

Complexity and Latency Analysis

To provide a fairer comparison between the proposed RACS-JSCC framework and the baseline methods, computational complexity and inference latency are further evaluated. For all methods, MACs and latency are reported for input images with resolution $224 \times 224 \times 3$. All inference-time measurements are conducted on the same hardware platform using an NVIDIA RTX 3090

Table 4.2: Complexity and latency comparison of RACS-JSCC and baseline JSCC-based image transmission methods. Latency values are reported per image and averaged over the Kodak test set.

Method	Params (M)	MACs (G)	Enc. (ms)	Dec. (ms)	Total (ms)
Deep JSCC	0.76	0.42	1.5	1.8	3.3
ADJSCC	1.28	0.58	2.1	2.7	4.8
RACS-JSCC	2.08	1.12	3.4	6.9	10.3

GPU, and the reported latency values are averaged over the Kodak test dataset. The encoder latency includes source-to-channel-symbol mapping and the sampling-rate selection module, while the decoder latency includes channel-symbol recovery, rate-dependent reconstruction, and final image refinement.

Since RACS-JSCC is a rate-adaptive method, its MACs and latency depend on the selected sampling branches. Therefore, the reported values correspond to the average inference cost over the test images using the actual block-wise rate decisions produced by the sampling-rate selection module. This gives a more realistic estimate than reporting only the worst-case high-rate branch.

As shown in Table 4.2, Deep JSCC has the lowest parameter count and latency because it uses a compact fixed-rate encoder-decoder architecture without rate selection or CS-based reconstruction. ADJSCC introduces attention-based SNR adaptation, which increases the number of parameters and inference time moderately. The proposed RACS-JSCC introduces additional complexity due to the sampling-rate selection module and the rate-specific CS sampling and reconstruction branches. However, this additional cost remains moderate because the selector is lightweight and operates at the block level, while only one sampling branch is activated for each block.

The proposed RACS-JSCC requires higher latency than Deep JSCC and ADJSCC, mainly because the decoder performs rate-dependent separation, initial reconstruction, and deep nonlinear refinement. However, this additional cost is still within a practical range for learned wireless image transmission. The encoder remains relatively lightweight because the CS sampling operation is implemented using convolutional sampling filters, and the rate-selection network only produces three branch probabilities per block. More importantly, RACS-JSCC reduces the average number of transmitted measurements by allocating low or medium sampling rates to visually redundant regions. Therefore, the added computational cost is justified by the improved

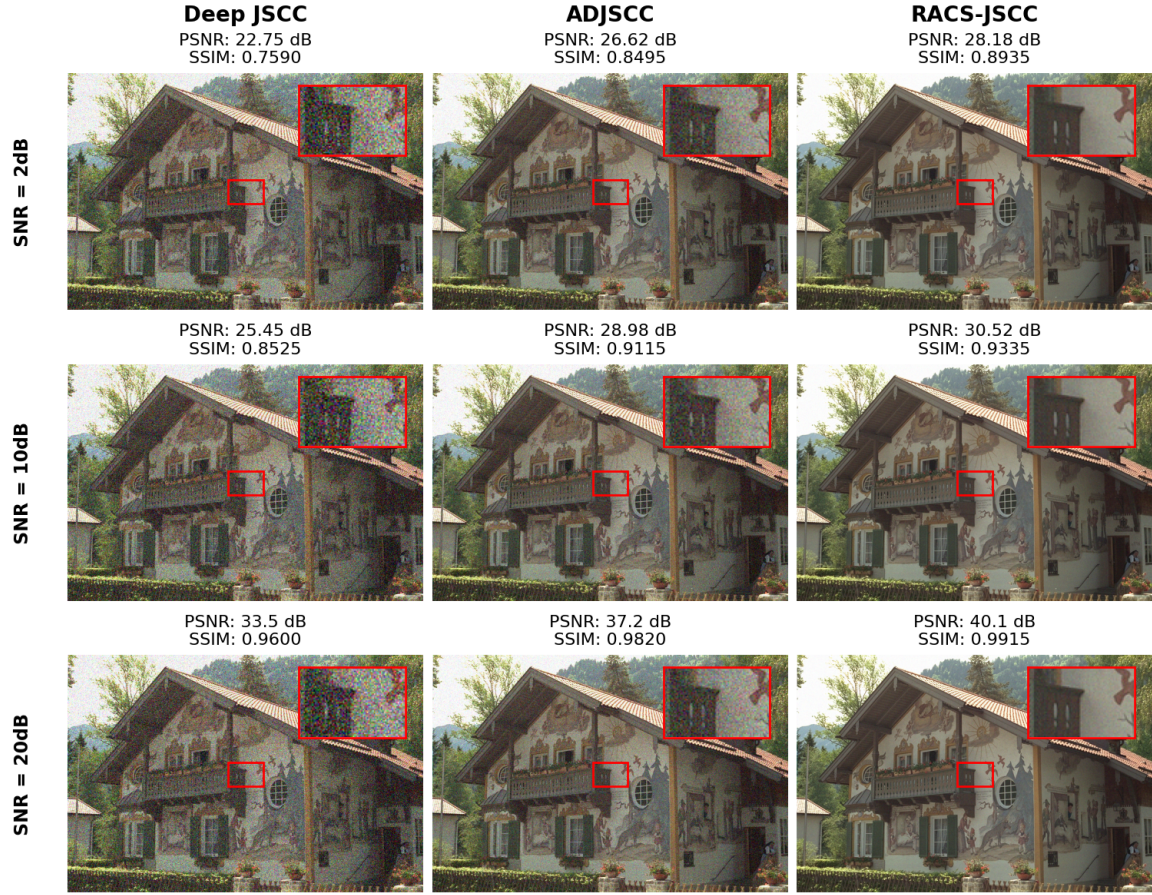


Figure 4.11: Reconstruction quality of an exemplary image (PSNR/SSIM) with RACS-JSCC and baseline methods under the compression ratio $k/n = 1/6$ and $SNR_{train}=4dB$

reconstruction quality and the reduction in effective bandwidth usage. This supports the design choice of using a block-wise rate-adaptive CS front end rather than a fixed-rate sampling strategy.

Visual Comparison

Figure 4.11 presents a detailed visual comparison of reconstructed images produced by Deep JSCC, ADJSCC, and the proposed RACS-JSCC framework under three channel conditions (2 dB, 10 dB, and 20 dB). All models are trained on the ImageNet dataset and evaluated over an AWGN channel. For each reconstruction, the corresponding PSNR and SSIM values are reported, and a magnified region of interest is included to highlight differences in local structural fidelity. At an SNR of 2 dB, Deep JSCC exhibits pronounced noise artefacts across the entire image. Textures on the house, windows, and surrounding vegetation appear heavily corrupted, and the enlarged patch reveals severe loss of edge information and colour consistency. ADJSCC reduces a portion

of this noise and produces a smoother visual appearance; however, its restored textures remain mottled and structurally unstable. In contrast, RACS-JSCC achieves a substantially cleaner reconstruction. Fine edges along window frames and roof structures are more sharply defined, colour transitions are more coherent, and noise amplification in smooth regions is significantly reduced. These qualitative improvements are reflected in the higher PSNR and SSIM values obtained by RACS-JSCC at this low SNR regime.

At 10 dB, the performance disparity becomes even more evident. Deep JSCC still retains noticeable grain and distortion, particularly in high-frequency textured areas such as the wall paintings and wooden balcony elements. ADJSCC exhibits improved stability, yet its reconstructed patterns remain partially blurred, and the zoomed-in patch reveals residual artefacts that compromise structural detail. RACS-JSCC, however, reconstructs both global and local features with markedly higher fidelity. Wall textures and architectural details are more precisely recovered, and the overall contrast and tonal transitions align more closely with the original image. The magnified region illustrates that RACS-JSCC not only suppresses channel-induced noise but also avoids the oversmoothing that characterizes the baseline methods, preserving crisp edges and fine textures.

At 20 dB, all models benefit from the improved channel conditions, but the advantages of RACS-JSCC remain clearly visible. Deep JSCC exhibits residual smoothing and faint noise patterns, while ADJSCC presents a cleaner output yet still struggles with subtle distortions in repetitive high-frequency structures. By comparison, RACS-JSCC produces the most natural and faithful reconstruction among the three. Edges are sharply delineated, textures exhibit high clarity, and the image maintains accurate colour reproduction across the scene. The enlarged patch highlights the ability of the proposed method to preserve delicate structural details with minimal distortion, yielding the highest SSIM score in this comparison. These observations demonstrate that the rate-adaptive design of RACS-JSCC not only enables efficient bandwidth utilization but also ensures robust, perceptually superior reconstruction across a wide range of channel qualities.

4.4 URC-JSCC: Unified Rate Controlled Deep Compressive JSCC

In this section, URC-JSCC is presented as a unified rate-controlled deep compressive JSCC framework that enables bandwidth adaptivity within a single end-to-end model while improving reconstruction quality compared with the RACS-JSCC design. The framework is motivated by two complementary insights from earlier chapters: DCS-JSCC (Section 3.4) shows that stronger feature normalization and a more expressive reconstruction pipeline can significantly enhance robustness under channel noise, whereas RACS-JSCC (Section 4.3) demonstrates that content- and SNR-aware blockwise rate selection provides an effective mechanism for varying the measurement budget across an image. URC-JSCC combines these principles but strengthens their integration by coupling rate selection, sampling, encoding, and reconstruction more tightly. In particular, the sampling stage features hierarchical learned block sampling modules that compact multi-scale information before measurement projection. Meanwhile, a hybrid CNN-Transformer backbone captures long-range dependencies and efficiently fuses block features from heterogeneous sampling rates. Lastly, a rate-aware training objective introduces a distortion-rate trade-off to dynamically shape the operating point during optimization. These design choices result in a more unified and controllable rate-adaptive pipeline with improved rate-distortion behaviour and higher reconstruction fidelity across varying SNR and bandwidth conditions.

4.4.1 Model Architecture

The design of URC-JSCC inherits the sampling rate selection module, GDN/IGDN normalization, and initial and deep reconstruction networks from prior DCS-JSCC in section 3.4 and RACS-JSCC in section 4.3, which are known to be effective for learned image transmission. However, it unifies the encoding and decoding process and strengthens the sampling stage by adopting a learned, hierarchical block sampler.

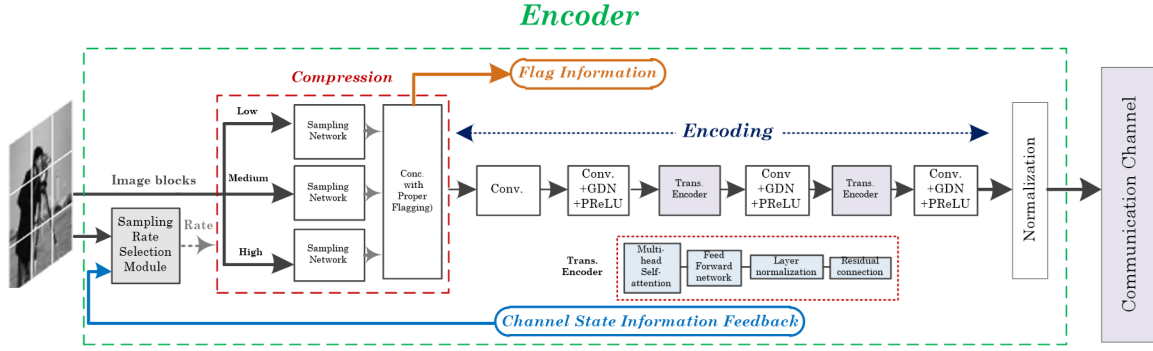


Figure 4.12: Architecture of the URC-JSCC encoder.

Encoder

The overall architecture of the URC-JSCC encoder is shown in Fig. 4.12. The encoding process begins by partitioning the image into non-overlapping blocks of size $B \times B \times C$, where C denotes the number of colour channels and B the block size. The image blocks are then fed into the sampling rate selection module, which computes both global and local statistics for each block. The sampling rate is defined as the ratio between the number of compressed measurements produced by the block-based CS sampler and the total number of pixels in the image. The extracted features, together with the SNR of the communication channel, are used to determine the sampling rate for each image block. Once the rate has been selected, the block is compressed by the corresponding sampling branch and the resulting measurements are passed to a unified encoder.

The sampling-rate selection module follows Section 4.3.1 (Fig. 4.3) and assigns each block to one of three discrete operating points, $r \in \{r_{\text{Low}}, r_{\text{Medium}}, r_{\text{High}}\}$, using block-level features together with the channel SNR. Conditioned on this decision, the block is processed by the corresponding rate-specific sampling network to produce a compact vector of CS measurements. The three sampling networks share the same architectural structure (i.e., the same type and ordering of layers), but use rate-specific parameters; within a given rate, the sampler weights are shared across all blocks routed to that rate.

Each rate-specific sampler is constructed by stacking K_r linear downsampling modules inspired by the GSM design in [170]. Fig. 4.13 depicts the sampling network, which shows the sampler at rate r that applies K_r such modules sequentially. As illustrated in Fig. 4.13 with a

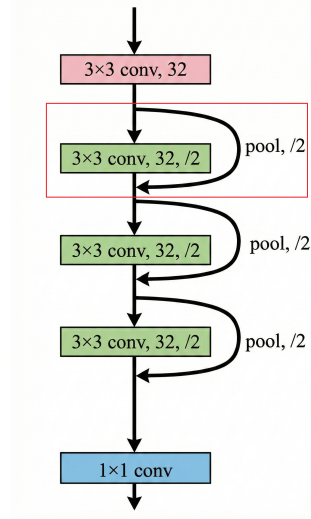


Figure 4.13: Architecture of a GSM-inspired linear downsampling module used as the URC-JSCC sampling network [170] (K_r is 3 in this figure).

red box, one sampling module combines two linear paths: (i) a 3×3 convolution with stride 2 and (ii) a 2×2 mean-pooling shortcut with stride 2; their outputs are summed to form a residual downsampled feature map. After the final module, a 1×1 convolution projects the resulting feature map to m_r channels, which are then flattened to obtain the measurement vector for each block (m_r denote the number of measurements allocated to a block when operating at rate r). Since the sampler uses only convolutions and mean-pooling without bias terms or activation functions, the resulting sampling operator remains linear while being learned end-to-end. The depth K_r controls the amount of spatial aggregation performed before the final projection. In practice, lower-rate branches (smaller m_r) typically use a deeper hierarchy (larger K_r) so that each measurement summarizes a wider spatial context within the block, which helps maintain informative measurements under a tight measurement budget. In the experiments reported in this section, a representative setting is $K_{\text{Low}} = 3$, $K_{\text{Medium}} = 2$, and $K_{\text{High}} = 1$.

Having m_r as defined in previous paragraph, the three per-block budgets are defined as:

$$(m_{r_{\text{Low}}}, m_{r_{\text{Medium}}}, m_{r_{\text{High}}}) = (\alpha_{r_{\text{Low}}}, \alpha_{r_{\text{Medium}}}, \alpha_{r_{\text{High}}}) B^2 C, \quad (4.8)$$

where $0 < \alpha_{r_{\text{Low}}} < \alpha_{r_{\text{Medium}}} < \alpha_{r_{\text{High}}} \leq 1$. The values of α_r specify the supported operating points and are chosen to span the target symbol-budget range used in the experiments. Here, the symbol budget refers to the maximum number of channel symbols permitted per image by the available

bandwidth. In implementation, m_r is obtained by rounding $\alpha_r B^2 C$ to the nearest integer, so the effective sampling ratio at rate r becomes $m_r/(B^2 C)$.

For each image block with the selected rate, a compact label is also attached that records this choice. The three rates are encoded with two bits, using $00 = r_{\text{Low}}$, $01 = r_{\text{Medium}}$ and $10 = r_{\text{High}}$, and leave the pattern 11 unused. Collecting the labels of all N_B blocks from left-to-right, top-to-bottom gives a short rate header of length $2N_B$ bits. Because the header is very small, it can be protected by repeating each bit several times (e.g., 3 or 5 times) and using majority vote at the receiver. This makes the rate flags robust to bit errors without adding noticeable overhead. The protected header is transmitted as a small control field with negligible bandwidth overhead and does not affect the main symbol budget.

Finally, the measurements from all blocks are combined into a single vector. The $\mathbf{y}_b$ vectors are taken in row-by-row order and then they are placed one after another to form:

$$\tilde{\mathbf{y}} = [\mathbf{y}_1^T, \mathbf{y}_2^T, \dots, \mathbf{y}_N^T]^T \in \mathbb{R}^M \quad (4.9)$$

where $M = \sum_{b=1}^{N_B} m_{r(b)}$ is the total number of CS measurements produced by the sampling networks. The protected rate header is sent alongside $\tilde{\mathbf{y}}$, allowing the decoder to identify the rate used for each block and route its measurements to the correct reconstruction branch.

After forming the measurement vector, the encoder maps it to a channel-ready latent representation using a hybrid CNN-Transformer stack. This encoding architecture combines the strengths of convolutional processing (strong local priors and spatial regularity) with the global modeling capabilities of Transformer attention, as illustrated in Fig. 4.12. The encoding pipeline consists of the following stages.

- *Initial convolutional projection:* The concatenated measurement vector $\tilde{\mathbf{y}}$ is reshaped into a two-dimensional tensor of size $H_e \times W_e \times 1$, where H_e and W_e are chosen such that $H_e W_e = M$. To maintain a balanced aspect ratio, H_e and W_e is chosen such that the resulting feature map is approximately square. However, any pair satisfying $H_e W_e = M$ is admissible. Since convolutions operate on spatially structured tensors, the one-dimensional measurement vector is reshaped into $H_e \times W_e$. This reshaping step introduces a spatial structure that

allows subsequent convolutional layers to operate effectively on the measurements, since convolutions require two-dimensional feature maps rather than one-dimensional vectors. To do so, a 3×3 convolution with stride 1 and 64 output channels is then applied. This produces a feature map of size $H_e \times W_e \times 64$. The purpose of this step is to lift the one-dimensional compressed measurements into a spatial feature representation that can be processed by subsequent CNN and Transformer layers.

- *Conv-GDN-PReLU module:* The resulting feature map is passed through a 3×3 convolution with stride 1 and 64 output channels, followed by a GDN layer and a PReLU activation. After this block, the feature tensor remains $H_e \times W_e \times 64$. The convolution models short-range dependencies between neighboring measurement groups. GDN, which is a parametric form of local gain control widely used in learned image compression, stabilizes activation statistics, and PReLU introduces a learnable non-linearity that improves convergence.
- *Transformer encoder layer:* To apply self-attention, the tensor is flattened into a sequence of $L = H_e W_e$ tokens, each of dimension 64. A linear projection maps each token to the Transformer dimension $d_{\text{model}} = 256$, producing a matrix of size $L \times 256$. Positional encodings are added to preserve spatial ordering. A Transformer encoder layer is then applied, consisting of multi-head self-attention with 4 heads, followed by a feed-forward network with hidden dimension 1024 (i.e., $4d_{\text{model}}$), along with layer normalization and residual connections. The output of this module remains $L \times 256$.
- *Second Conv-GDN-PReLU block:* The Transformer output is reshaped back to a spatial tensor of size $H_e \times W_e \times 256$. A 3×3 convolution with stride 1 and 64 output channels is applied, followed by GDN and PReLU. This reduces the channel depth back to 64 and restores local spatial structure after the attention operation.
- *Second Transformer encoder layer:* The tensor ($H_e \times W_e \times 64$) is again flattened into a sequence of L tokens and projected to dimension 256. A second Transformer encoder layer (processed identically to the first) refines and strengthens global dependencies and interactions between blocks sampled at different rates. The output again has size $L \times 256$,

which is reshaped back to $H_e \times W_e \times 256$.

- *Final Conv-GDN-PReLU block:* A final 3×3 convolution with 64 output channels, followed by GDN and PReLU, produces the latent tensor of size $H_e \times W_e \times 64$. This tensor is then linearly projected to complex channel symbols in the subsequent channel mapping stage.

The activation and normalization choices in URC-JSCC follow the same design logic as the earlier compressive JSCC models. PReLU is used in the CNN-based encoder and decoder blocks because it preserves negative responses in the continuous latent representation through a learnable slope, which is important before and after wireless transmission. GDN and IGDN are used to normalize and invert feature statistics, improving compactness and stability under channel noise. In contrast, ReLU is used in the final residual reconstruction branches because these layers operate in the image-domain refinement stage, where sparse nonlinear residual correction is sufficient and computationally efficient. Using above-mentioned consecutive modules and the alternating CNN and Transformer structure enables the encoder to jointly capture local textures, global relationships across the entire measurement sequence, and cross-block dependencies arising from heterogeneous sampling rates. This hybrid design produces a compact and robust latent representation suitable for reliable transmission over noisy wireless channels.

As a final step, the concatenated vector of encoded block representations is power-normalized according to Eq. (3.7), yielding the channel input vector $\mathbf{z}$. To map the variable-length measurement vector described in Eq. (4.9) to complex channel symbols, the encoding stage produces a real vector of length $2k$ and then forms k complex symbols by in-phase and quadrature pairing. Specifically, let $\mathbf{s} \in \mathbb{R}^{2k}$ denote the real-valued output of the final projection layer of the CNN-Transformer encoder. The transmitted symbols are constructed as

$$\mathbf{z} = \mathbf{s}_{1:k} + j\mathbf{s}_{k+1:2k} \in \mathbb{C}^k. \quad (4.10)$$

In URC-JSCC implementation, k is chosen proportional to the measurement budget, with $k = \lceil M/2 \rceil$ (and one zero is padded if M is odd before reshaping), so the number of channel uses varies with the selected block rates but remains bounded by $k \leq \lceil Nm_{\text{High}}/2 \rceil$. Then, The

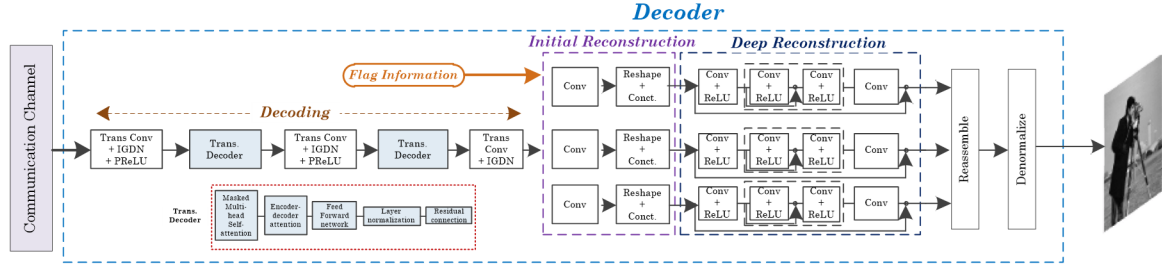


Figure 4.14: Architecture of the proposed decoder.

resulting joint source-channel coded sequence is then transmitted over the wireless channel, which is modeled as a non-trainable layer in the end-to-end architecture with transfer function as Eq. (3.5).

Decoder

The decoder receives the noisy channel symbols $\hat{\mathbf{z}}$ and the protected rate header. After error correction, the rate associated with each block is recovered and used to guide the reconstruction process. The received symbols are mapped through a learned linear projection to produce a latent tensor. This latent representation is then processed by a sequence of transposed convolutional blocks interleaved with Transformer decoder layers that mirror the structure of the encoder and restore coherent spatial and global relationships in the feature space. The overall decoder architecture is shown in Fig. 4.14.

The first stage applies a transposed convolution with kernel size 3×3 , stride 1 and 64 output channels, followed by IGDN and PReLU. IGDN serves as the inverse of the encoder's GDN transformation and restores the dynamic range and balance of the features. The output tensor of size $H_e \times W_e \times 64$ is flattened into a sequence of L tokens, each of dimension 64. A linear projection lifts each token to a 256-dimensional embedding, and positional encodings are added to preserve spatial ordering. A Transformer decoder layer is then applied, consisting of multi-head self-attention with four heads, a feed-forward network with hidden dimension 1024, layer normalization and residual connections. The output is reshaped back to $H_e \times W_e \times 256$ and passed to the next transposed convolutional stage.

The second transposed convolutional block uses a 3×3 transposed convolution with stride 1 and 64 output channels, followed by IGDN and PReLU, reducing the channel dimension from 256 back to 64. A second Transformer decoder layer, identical in structure to the first, refines

the global dependencies and enhances consistency across blocks sampled at different rates. Its output is reshaped to $H_e \times W_e \times 256$ and processed by a final transposed convolutional block with kernel size 3×3 , stride 1 and 64 output channels, followed by IGDN. This produces the decoded feature map that serves as input for blockwise reconstruction.

The decoded feature tensor is partitioned into block level feature vectors that correspond to the original image blocks. The recovered rate header determines which reconstruction branch each block is assigned to. Three reconstruction branches are used, one for each of the low, medium and high sampling rates. The first stage in each branch performs initial linear reconstruction. For each block, a 1×1 convolution with CB^2 output channels is applied to its feature vector. Because no activation is used, this layer acts as a purely linear operator that approximates the inverse mapping of the sampling process. The CB^2 outputs are reshaped to a block of size $B \times B \times C$, and all blocks are placed back into their correct spatial positions to form an initial image estimate.

The initial reconstruction is then refined by a deep residual reconstruction network that operates separately for each rate branch. Each branch begins with a 3×3 convolution with 64 output channels and ReLU activation that extracts high-dimensional features from the initial estimate. These features are processed by a sequence of residual blocks. Each residual block contains two 3×3 convolutions with 64 channels and ReLU activations, together with a skip connection that adds the input of the block to its output. The residual structure increases effective depth and receptive field while maintaining stable training, enabling the network to correct distortions introduced by compression and channel noise. A final 3×3 convolution with C output channels aggregates the refined features into a reconstructed block. A global skip connection adds the initial reconstruction to the output of the deep network so that the branch predicts residual corrections rather than the full image content, a strategy known to improve convergence and reconstruction fidelity.

After all blocks have been reconstructed, the outputs of the three branches are placed back into their original positions to form the full image. If overlapping blocks were used during encoding, overlapping regions are merged by simple averaging or a smooth blending function. Finally, a denormalization layer maps the pixel values of the reconstructed image back to the original intensity range, producing the decoder output $\hat{\mathbf{x}}$.

4.4.2 Loss Function

The proposed URC-JSCC encoder and decoder are trained jointly in an end-to-end pipeline consistent with the system model in Section 4.2. Let $\{\mathbf{x}_i\}_{i=1}^N$ denote the training set, where each image is represented as a real-valued vector $\mathbf{x}_i \in \mathbb{R}^n$. The encoder implements a mapping $f(\cdot; \theta)$ that includes the sampling-rate selection module, the rate-specific sampling networks, and the subsequent channel encoding stage. Given $\mathbf{x}_i$, the encoder produces complex-valued channel symbols as $\mathbf{z}_i = f(\mathbf{x}_i; \theta) \in \mathbb{C}^k$.

At the receiver, the decoder follows the three-stage structure described in the system model: a decoding network $g(\cdot; \varphi)$ that maps $\hat{\mathbf{z}}_i$ to an intermediate representation $\tilde{\mathbf{x}}_i$, an initial reconstruction network $h(\cdot; \psi)$ that produces $\hat{\mathbf{x}}_{i,\text{init}}$, and a deep reconstruction network $d(\cdot; \chi)$ that refines the estimate into the final reconstruction $\hat{\mathbf{x}}_i$:

$$\tilde{\mathbf{x}}_i = g(\hat{\mathbf{z}}_i; \varphi), \quad (4.11)$$

$$\hat{\mathbf{x}}_{i,\text{init}} = h(\tilde{\mathbf{x}}_i; \psi), \quad (4.12)$$

$$\hat{\mathbf{x}}_i = d(\hat{\mathbf{x}}_{i,\text{init}}; \chi). \quad (4.13)$$

The reconstruction objective is defined as the MSE between $\hat{\mathbf{x}}_i$ and $\mathbf{x}_i$:

$$L_{\text{rec}}(\theta, \varphi, \psi, \chi) = \frac{1}{N} \sum_{i=1}^N \|d(h(g(\eta(f(\mathbf{x}_i; \theta))); \varphi); \psi); \chi) - \mathbf{x}_i\|_2^2. \quad (4.14)$$

In addition to accurate reconstruction, URC-JSCC is designed to adapt the sampling rate across image blocks rather than using the highest sampling rate for all blocks. To encourage the encoder to use fewer measurements, a differentiable rate penalty is introduced. Each image is partitioned into N_B blocks, and the rate-selection module (as part of the encoder) produces, for block b of image i , a probability distribution $p_{i,b}(r; \theta_{\text{rs}})$ over the discrete rate set $r \in \{r_{\text{Low}}, r_{\text{Medium}}, r_{\text{High}}\}$. Let $\theta_{\text{rs}} \subset \theta$ denote the parameters of the sampling-rate selection module and α_r denote the sampling ratio associated with rate r . The expected sampling ratio for

block b of image i is defined as

$$\alpha_{i,b}(\theta_{\text{rs}}) = \sum_{r \in \{r_{\text{Low}}, r_{\text{Medium}}, r_{\text{High}}\}} p_{i,b}(r; \theta_{\text{rs}}) \alpha_r, \quad (4.15)$$

and the average sampling ratio for image i is

$$\bar{\alpha}_i(\theta_{\text{rs}}) = \frac{1}{N_B} \sum_{b=1}^{N_B} \alpha_{i,b}(\theta_{\text{rs}}). \quad (4.16)$$

Accordingly, the rate penalty is defined as

$$L_{\text{rate}}(\theta_{\text{rs}}) = \frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} \bar{\alpha}_i(\theta_{\text{rs}}). \quad (4.17)$$

Where N_{tr} is the number of training images (batch size during training). The dependency of $L_{\text{rate}}(\theta_{\text{rs}})$ on θ is through the probabilities $p_{i,b}(r; \theta_{\text{rs}})$ produced by the encoder-side rate-selection module, which makes the penalty fully differentiable during training.

Finally, the total training objective combines reconstruction fidelity and rate efficiency:

$$L(\theta, \varphi, \psi, \chi) = L_{\text{rec}}(\theta, \varphi, \psi, \chi) + \lambda L_{\text{rate}}(\theta_{\text{rs}}). \quad (4.18)$$

where $\lambda \geq 0$ controls the trade-off between reconstruction quality and the average sampling ratio. Setting $\lambda = 0$ reduces the objective to pure end-to-end MSE minimization, while larger λ encourages the rate-selection module to assign more blocks to lower sampling ratios when possible. An Ablation study on value of λ is done in the following.

4.4.3 Results and Discussion

The URC-JSCC encoder and decoder are trained jointly in an end-to-end pipeline using large-scale image datasets and randomly sampled channel realizations. Training is performed on CIFAR-10 and ImageNet. For ImageNet, images are randomly cropped into patches of size 224×224 . CIFAR-10 images of size 32×32 are used directly. The datasets are split into training and validation subsets using a ratio of 9 : 1, and none of the test images appear during training.

During training, each mini-batch is passed through a simulated AWGN or Rayleigh fading channel. For every forward pass, an SNR value is drawn uniformly from the interval $[0, 20]$ dB, which exposes the model to a wide range of distortion levels and enables SNR robustness. The rate-selection module receives both image features and the chosen SNR, ensuring that sampling decisions are conditioned on channel quality.

The models are optimized with the Adam optimizer. For CIFAR-10, the initial learning rate is set to 10^{-3} and reduced to 10^{-4} after 500000 iterations. For ImageNet-based training, a fixed learning rate of 10^{-4} is used throughout. All training uses mini-batches of size 64. The loss function described in subsection 4.4.2 is minimized by backpropagation using a fixed trade-off coefficient $\lambda = 0.1$ for all experiments. Unless stated otherwise, λ is kept constant across training and evaluation. Gradients flow through all of the encoder and decoder modules.

The discrete rate labels produced by the sampling-rate selection module are handled using a straight-through estimator. In the forward pass, the module uses the hard rate decision to select the sampling branch and outputs a discrete rate $r \in \{r_{\text{Low}}, r_{\text{Medium}}, r_{\text{High}}\}$; in the backward pass, gradients flow through the pre-softmax logits. This preserves differentiability and avoids instability during training. Power normalization is applied during training exactly as during inference so that the encoder learns to operate under the channel power constraint. All components are trained jointly, although the encoder and decoder can be executed independently after convergence.

The performance of the proposed URC-JSCC model is evaluated on both the CIFAR-10 and Kodak datasets. For the Kodak dataset, the 24 RGB images are center-cropped to the largest square region that fits within the original frame and then used at their native resolution. No data augmentation is applied during evaluation. All experiments assume transmission over AWGN or Rayleigh channel. For each SNR point, the channel noise is regenerated independently for every transmission. To reduce the effect of randomness, each image is transmitted multiple times and the final PSNR is obtained by averaging across trials. For CIFAR-10, each test image is transmitted 10 times. For Kodak, each image is transmitted 100 times.

The rates $(\alpha_{\text{Low}}, \alpha_{\text{Medium}}, \alpha_{\text{High}})$ are fixed hyperparameters that define the measurement ratios associated with the three sampling rates. In the URC-JSCC implementation, they are set

as:

$$(\alpha_{\text{Low}}, \alpha_{\text{Medium}}, \alpha_{\text{High}}) = (1/16, 1/8, 1/4) \quad (4.19)$$

which provides a clear separation between low, medium and high sampling densities while remaining compatible with the capacity of the GSM sampling networks. These values remain constant for all images during training and testing. The sampling-rate selection module adaptively chooses among these predefined ratios for each block based on image content and channel SNR, so the total number of CS measurements varies from image to image. Nevertheless, the total number of measurements per image is inherently bounded because each block selects its rate from a finite set. Let the image be partitioned into N_B non-overlapping blocks of size $B \times B \times C$. The total number of CS measurements is $M = \sum_{b=1}^{N_B} m_{r(b)}$, where $r(b)$ denotes the rate assigned to block b . Since $m_{r(b)} \leq m_{\text{High}}$ for all b , it follows that:

$$M \leq N_B m_{\text{High}} = N_B \alpha_{\text{High}} B^2 C. \quad (4.20)$$

The worst case occurs only when all blocks are assigned to the High branch. In practice, α_{High} is selected such that this worst-case bound remains within the maximum allowable symbol budget of the system, ensuring that URC-JSCC does not exceed the permissible transmission rate.

Results on CIFAR-10

Firstly, URC-JSCC is evaluated on the CIFAR-10 test set. The block size is set to $B = 8$, and for Deep JSCC and ADJSCC the overall compression ratio is fixed to $1/6$, while the other two benchmark methods, NTSCC and SwinJSCC, are rate adaptive ones. Fig. 4.15 shows the average PSNR as a function of SNR_{test} for an AWGN channel. In addition to reconstruction quality, the figure reports the effective bandwidth compression ratio achieved by URC-JSCC at selected SNR values as examples, defined as the ratio between the number of channel uses and the number of source symbols. For RGB images, the number of source symbols is equal to $H \times W \times 3$. Both baseline schemes operate at a fixed bandwidth compression ratio of $1/6 \approx 0.166$, whereas URC-JSCC adapts its average measurement rate on a per-image basis.

In Fig. 4.15, it can be observed that URC-JSCC consistently delivers higher PSNR than

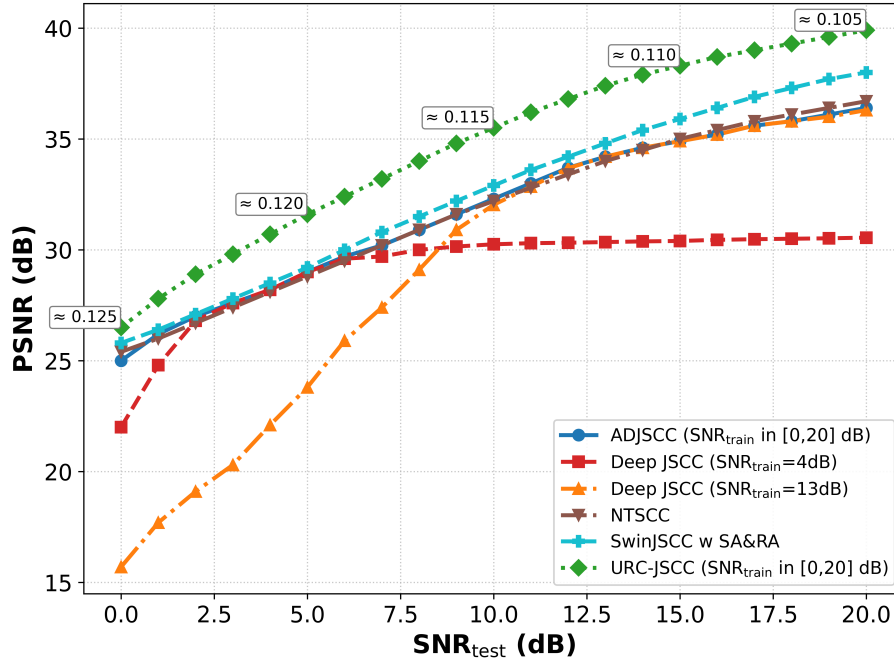


Figure 4.15: PSNR (y-axis), and bandwidth compression (boxed numbers) against varying SNR_{test} values [0-20 dB] for CIFAR-10 dataset over an AWGN channel. Deep JSCC and ADJSCC operate at a fixed bandwidth compression ratio of 1/6.

benchmark methods across the entire range $\text{SNR}_{\text{test}} \in [0, 20]$ dB. At low SNR, all schemes degrade gracefully, but URC-JSCC maintains a noticeable advantage by distributing the available symbol budget more effectively across blocks, prioritizing regions that contain edges, textures, and semantically important structures. Moreover, at medium and high SNR, the gain becomes more significant: URC-JSCC achieves higher PSNR while operating at a lower average bandwidth compression ratio than the baselines. This shows that the proposed rate-adaptive CS strategy is more efficient at exploiting redundancy across image regions. The annotated compression ratios in Fig. 4.15 reveal that the effective rate used by URC-JSCC decreases steadily as the SNR increases. At low SNR, the scheme transmits closer to 0.125 symbols per pixel to mitigate noise, whereas at high SNR the average ratio drops to around 0.107 because the channel reliability allows more aggressive compression with less redundancy. This behaviour confirms that the URC-JSCC sampling policy jointly adapts to image structure and channel quality. Smooth or low-detail regions are assigned to the low-rate branch, while textured or important blocks receive higher sampling density. As a result, the system preserves critical visual information while improving bandwidth efficiency, achieving higher PSNR with fewer

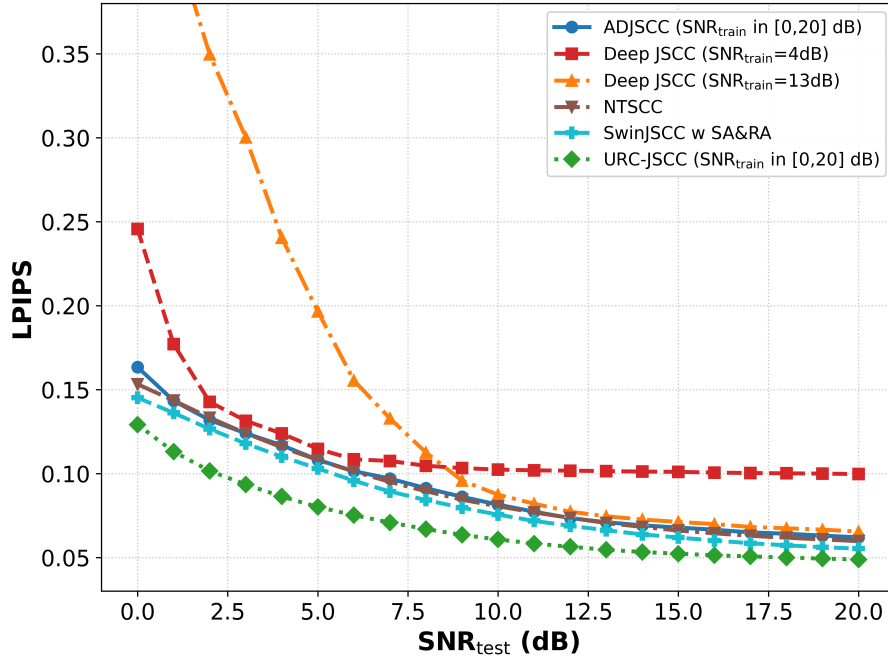


Figure 4.16: LPIPS comparison of URC-JSCC and baseline methods over the AWGN channel on the CIFAR-10 dataset.

transmitted symbols than benchmark methods.

To complement the PSNR-based evaluation, Fig. 4.16 reports the LPIPS comparison under the same setting. LPIPS evaluates perceptual similarity in a learned feature space and is useful because PSNR and SSIM do not fully capture perceptual sharpness and texture consistency. The LPIPS values decrease as the test SNR increases for all methods, which is consistent with the PSNR results. URC-JSCC obtains lower LPIPS than the baseline methods across most SNR values, indicating that the proposed rate-controlled reconstruction process improves not only pixel-level fidelity but also perceptual image quality.

Results on Kodak

URC-JSCC is further evaluated on the Kodak dataset to assess its behaviour on higher-resolution natural images. In this setting, a block size of $B = 32$ is considered, and a fixed compression ratio of $k/n = 1/6$ for Deep JSCC and ADJSCC is applied. Fig. 4.17 reports the average PSNR as a function of SNR_{test} under a Rayleigh channel. In addition to reconstruction quality, the figure also shows the effective bandwidth compression ratio achieved by URC-JSCC at each SNR level. Deep JSCC and ADJSCC operate at a fixed ratio of $1/6 \approx 0.166$, whereas URC-JSCC achieves

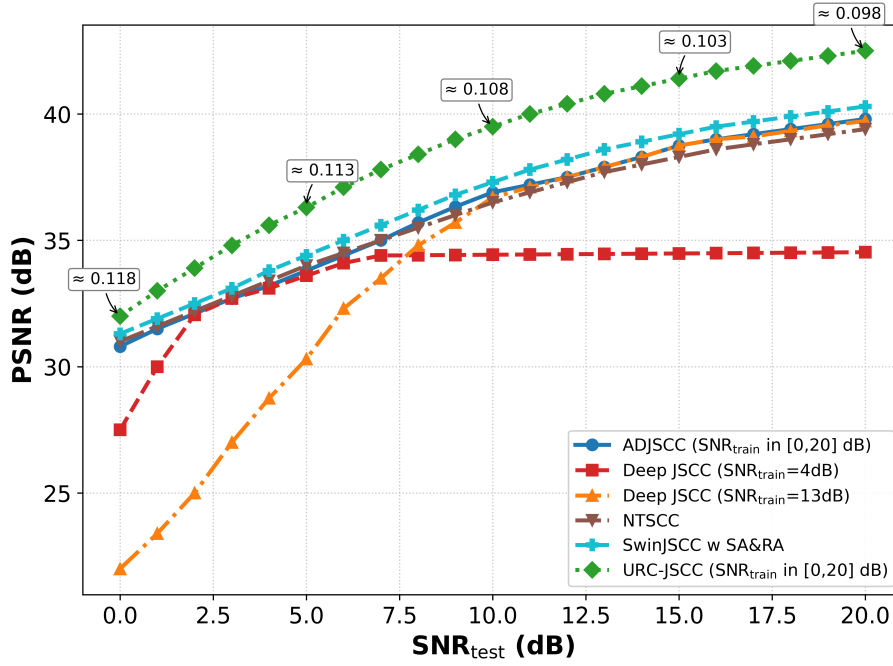


Figure 4.17: PSNR (y-axis), and bandwidth compression (boxed numbers) against varying SNR_{test} values [0-20 dB] for Kodak dataset over a Rayleigh channel. Deep JSCC and ADJSCC operate at a fixed bandwidth compression ratio of $1/6$.

substantially lower ratios ranging from roughly 0.118 at low SNR to about 0.100 at high SNR.

These results highlight two key advantages of the proposed system. First, URC-JSCC consistently achieves higher PSNR across the entire SNR range, owing to its blockwise adaptive sampling and hybrid CNN-Transformer encoder. Second, it does so with lower bandwidth usage than both baselines. The decreasing bandwidth compression ratio with increasing SNR confirms that the rate-selection module learns to exploit better channel conditions by allocating fewer symbols to each image while still improving reconstruction quality. Conversely, when the channel becomes noisy, URC-JSCC automatically increases the sampling of informative blocks, preserving robustness. This joint adaptation to image content and SNR enables URC-JSCC to preserve fine details, suppress channel artefacts and achieve superior rate-distortion performance compared to existing deep JSCC methods.

To further evaluate perceptual reconstruction quality, Fig. 4.18 reports the LPIPS comparison under the same Kodak AWGN setting. LPIPS measures perceptual distance in a learned feature space and therefore complements PSNR, which mainly reflects pixel-level distortion. The LPIPS values decrease as the test SNR increases for all methods, showing improved perceptual similarity under better channel conditions. URC-JSCC obtains the lowest LPIPS across most SNR values,

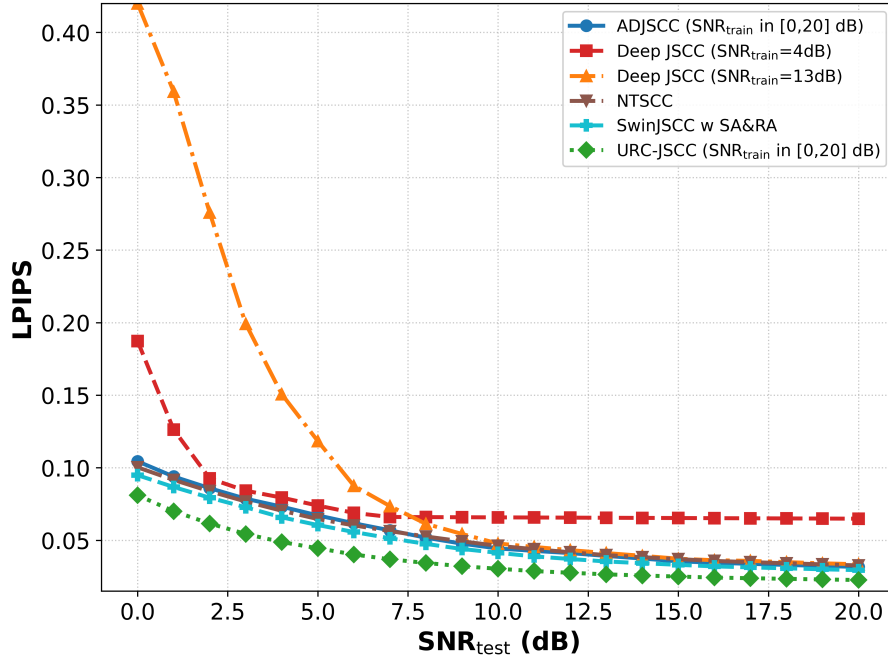


Figure 4.18: LPIPS comparison of URC-JSCC and baseline methods over the AWGN channel on the Kodak dataset.

indicating that the proposed rate-controlled reconstruction process improves not only pixel-level fidelity but also perceptual image quality. This is particularly important for Kodak images, where texture preservation and local structural details are more visible than in lower-resolution datasets.

Rate-distortion behaviour

To investigate the rate-distortion behaviour of URC-JSCC under varying bandwidth constraints, Fig. 4.19 reports the average PSNR versus the compression ratio (k) for the CIFAR-10 dataset over an AWGN channel at $\text{SNR}_{\text{test}} = 3$ dB. The results compare URC-JSCC with Deep JSCC, ADJSCC, NTSCC, and SwinJSCC w/o SA&RA. As expected, the PSNR of all methods increases monotonically as k increases, because a larger symbol budget allows more source information to be transmitted and reduces reconstruction distortion. Deep JSCC gives the lowest performance because it uses a fixed-rate encoder-decoder structure. ADJSCC improves the result through SNR-adaptive feature modulation, while NTSCC and SwinJSCC w/o SA&RA provide stronger reconstruction quality due to more expressive latent representations. However, URC-JSCC consistently achieves the highest PSNR across the evaluated range. This shows that the combination

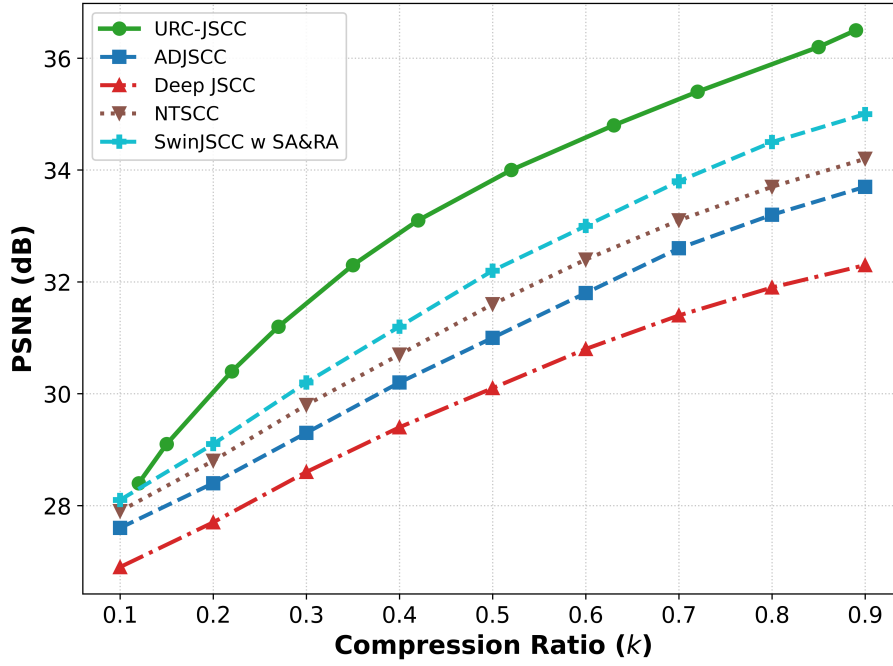


Figure 4.19: Rate-distortion comparison of URC-JSCC and baseline methods as PSNR versus compression ratio (k) for the CIFAR-10 dataset over an AWGN channel at $\text{SNR}_{\text{test}} = 3$ dB.

of block-wise rate selection, hierarchical CS sampling, protected rate signalling, and hybrid CNN-Transformer encoding allows URC-JSCC to exploit additional channel resources more effectively than the baseline methods.

The curves also show that URC-JSCC preserves a clear performance margin over the baselines throughout the whole range of symbol budgets. The gap becomes more visible at medium and high compression ratios, where the proposed architecture can benefit more from additional transmitted symbols. This behaviour is important for practical wireless image transmission systems where the available bandwidth may change over time and the JSCC scheme must operate reliably across different bandwidth constraints.

Ablation Study on the Contributions of URC-JSCC Core Components

To understand the contribution of core design choices in URC-JSCC, a series of ablation studies is conducted on the CIFAR-10 dataset, under a AWGN channel, at representative SNR values. Unless otherwise specified, all models are trained with the same procedure as the full URC-JSCC system. The results are summarised in Table 4.3.

Firstly, the benefit of blockwise rate adaptation is evaluated by comparing the full URC-

Table 4.3: Ablation studies on core components of URC-JSCC for CIFAR-10 over an AWGN channel. PSNR values are reported at two representative SNRs. The average sampling ratio $\bar{\alpha}$ is reported only where it is meaningful.

Configuration	PSNR (dB) @ 4 dB	PSNR (dB) @ 8 dB	$\bar{\alpha}$
Rate-adaptive sampling			
Low-only ($\alpha_{\text{Low}} = 1/16$)	29.1	31.0	0.0625
Medium-only ($\alpha_{\text{Medium}} = 1/8$)	30.6	32.8	0.1250
High-only ($\alpha_{\text{High}} = 1/4$)	31.4	33.4	0.2500
URC-JSCC (adaptive)	31.3	33.5	0.1600
Encoder architecture			
CNN only (no Transformer)	30.0	32.5	–
CNN + 1 Transformer layer	30.8	33.3	–
CNN + 2 Transformer layers	31.0	33.6	–
Reconstruction strategy			
Deep reconstruction only	30.2	32.8	–
Initial + deep reconstruction	31.0	33.6	–

JSCC model with three reduced variants that use a fixed sampling rate for all blocks: Low-only, Medium-only and High-only. All variants share the same encoder-decoder backbone and are trained with the corresponding fixed ratio α_r . As shown in Table 4.3, the fixed-rate models exhibit the expected trade-off between quality and symbol usage: the "Low-only" variant is the most efficient but yields the lowest PSNR. In contrast, the "High-only" variant provides the best quality at the expense of more measurements. URC-JSCC achieves PSNR close to or higher than the High-only model while using a substantially smaller average sampling ratio, which indicates that adaptive spatial allocation is more effective than any single fixed-rate configuration.

To assess the importance of the Transformer layers, the following three encoder variants are compared: a pure CNN encoder without any Transformer layers, a hybrid encoder with a single Transformer layer, and the proposed encoder with two Transformer layers. The sampling and decoder components are kept identical. Table 4.3 shows that adding a single Transformer layer already yields a clear PSNR gain, especially at higher SNRs, due to improved modelling of long-range dependencies and relationships between blocks sampled at different rates. Introducing a second Transformer layer brings an additional but smaller improvement, suggesting that a shallow Transformer stack is sufficient to capture the most important global interactions while keeping computational complexity manageable.

Finally, effect of the two-stage reconstruction strategy is examined by removing the initial

linear reconstruction module and training a decoder that directly maps the latent features to the final reconstructed image using only the deep residual network. As reported in Table 4.3, this modification leads to a consistent PSNR drop at both SNR points. The linear initial reconstruction provides a stable, task-specific inverse for the sampling operator and supplies a good starting point for the nonlinear refinement network. Learning the entire mapping purely with a deep CNN makes the optimisation problem harder and reduces robustness, particularly at low SNR. These results justify the use of an explicit linear reconstruction stage followed by a deep residual refinement.

Complexity and Latency Analysis

To further assess the practical cost of the proposed URC-JSCC framework, computational complexity and inference latency are evaluated and compared with the baseline methods. Similar to the previous complexity analysis, MACs and latency are reported for input images with resolution $224 \times 224 \times 3$. All measurements are performed on the same NVIDIA RTX 3090 GPU, and the reported latency values are averaged over the Kodak test dataset. The encoder latency includes rate selection, hierarchical CS sampling, CNN-Transformer feature encoding, and channel-symbol generation. The decoder latency includes channel-symbol recovery, Transformer-assisted decoding, rate-aware initial reconstruction, and residual deep reconstruction.

For URC-JSCC, the computational cost varies with the block-wise sampling decisions because different images may activate different combinations of low, medium, and high sampling branches. Therefore, the reported MACs and latency correspond to the average cost over the test set using the actual selected rates. The small protected rate header used by URC-JSCC introduces negligible computational overhead compared with the main encoder-decoder operations, and its cost is included in the total latency.

Table 4.4 reports the complexity and latency comparison. Deep JSCC remains the lightest method because it uses a compact fixed-rate CNN-based encoder-decoder architecture. ADJSCC adds an attention-based SNR adaptation mechanism and therefore has slightly higher complexity. URC-JSCC requires higher complexity than these CNN-based and CS-based baselines because it adds hierarchical GSM-inspired sampling, protected rate signalling, and CNN-Transformer

Table 4.4: Complexity and latency comparison of URC-JSCC and baseline JSCC-based image transmission methods. Latency values are reported per image and averaged over the Kodak test set.

Method	Params (M)	MACs (G)	Enc. (ms)	Dec. (ms)	Total (ms)
Deep JSCC	0.76	0.42	1.5	1.8	3.3
ADJSCC	1.28	0.58	2.1	2.7	4.8
SwinJSCC w SA&RA	16.92	3.72	11.4	11.9	23.3
NTSCC	32.90	5.42	36.0	15.0	51.0
URC-JSCC	4.72	2.86	6.8	9.7	16.5

feature fusion. However, compared with stronger recent baselines such as NTSCC and SwinJSCC, URC-JSCC has a substantially lower parameter count, MAC cost, and inference latency. This shows that the proposed design improves reconstruction quality without relying on a very large Transformer-only or entropy-model-based architecture.

The additional latency of URC-JSCC mainly comes from two components. First, the hierarchical sampling branches perform multi-stage linear downsampling before measurement projection, which is more expensive than the stride- B sampling used in RACS-JSCC. Second, the CNN-Transformer encoder and decoder introduce self-attention layers to model long-range dependencies between blocks sampled at different rates. This increases both MACs and latency compared with Deep JSCC and ADJSCC. However, this additional complexity is directly linked to the improved reconstruction performance reported in the previous results. The Transformer layers help fuse heterogeneous-rate block information, while the rate-regularized training objective encourages the model to use fewer transmitted symbols when high-rate sampling is not necessary.

Compared with NTSCC and SwinJSCC, URC-JSCC achieves a more favourable performance-complexity trade-off. Although NTSCC and SwinJSCC use more expressive architectures, their parameter counts and inference latencies are considerably higher. In contrast, URC-JSCC remains much lighter while achieving stronger reconstruction performance in the reported PSNR and LPIPS comparisons. Therefore, the added complexity of URC-JSCC is not introduced only to increase model capacity, but to improve the efficiency and reliability of rate-adaptive image transmission through hierarchical CS sampling, protected rate signalling, CNN-Transformer fusion, and explicit rate control.

Table 4.5: Ablation study on loss function hyper-parameter λ in Eq. (4.18) for URC-JSCC on CIFAR-10 over an AWGN channel. Values are reported at representative SNR_{test} points.

λ	$\text{SNR}_{\text{test}} = 0 \text{ dB}$		$\text{SNR}_{\text{test}} = 10 \text{ dB}$		$\text{SNR}_{\text{test}} = 20 \text{ dB}$	
	k/n	PSNR (dB)	k/n	PSNR (dB)	k/n	PSNR (dB)
0	0.135	26.8	0.125	35.8	0.115	40.2
0.05	0.130	26.6	0.120	35.7	0.110	40.0
0.10	0.125	26.5	0.115	35.5	0.105	39.9
0.20	0.120	26.2	0.110	35.2	0.100	39.5
0.50	0.112	25.7	0.102	34.6	0.092	38.8

Ablation Study on the Loss Function Hyper-parameter

The rate-regularization weight λ in the URC-JSCC loss function in Eq. (4.18) controls the distortion-rate trade-off learned by the sampling-rate selection module. Intuitively, a larger λ increases the cost of selecting higher sampling ratios, so the encoder is pushed to route more blocks through lower-rate branches whenever the reconstruction loss can be maintained, yielding a smaller achieved bandwidth compression ratio (k/n). Conversely, smaller λ values prioritize reconstruction fidelity and typically result in a higher achieved k/n . Since URC-JSCC is explicitly rate-adaptive, demonstrating sensitivity to λ is important to show that the gains are not obtained simply by transmitting more symbols.

To quantify this effect, URC-JSCC is trained with different λ values while keeping the architecture, data, and training protocol fixed, and evaluated on CIFAR-10 over an AWGN channel across $\text{SNR}_{\text{test}} \in [0, 20]$ dB. Table 4.5 reports PSNR together with the achieved effective k/n at representative SNR points. For reference, Deep JSCC and ADJSCC operate at a fixed $k/n = 1/6$ for all test SNR values, whereas URC-JSCC learns an SNR-dependent operating point: the achieved k/n decreases as SNR_{test} increases because higher channel reliability allows the model to use less redundancy. Overall, the table shows the expected monotonic trend: increasing λ reduces the achieved k/n but also slightly reduces PSNR, revealing a smooth controllable trade-off. In this chapter, $\lambda = 0.1$ is used as it provides a good balance between reconstruction quality and bandwidth efficiency.

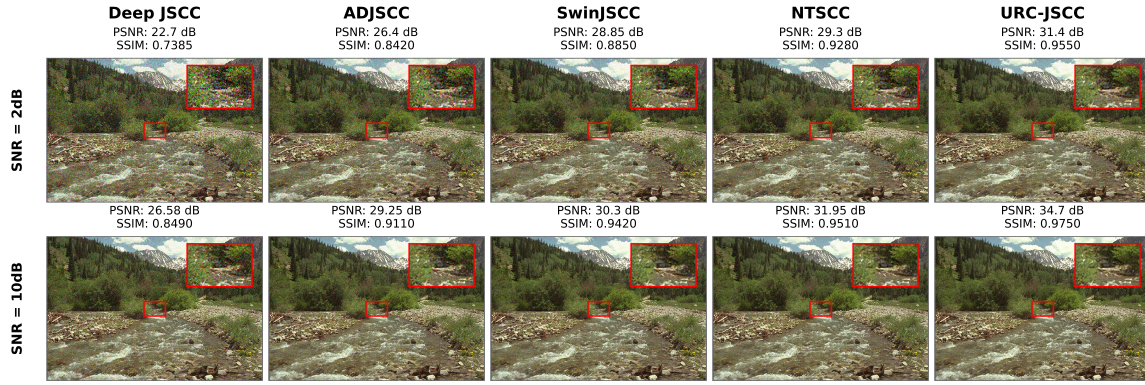


Figure 4.20: Reconstruction quality of an exemplary image (PSNR/SSIM) for URC-JSCC versus baseline methods at different SNR values. Deep JSCC and ADJSCC use the compression ratio $k/n = 1/6$.

Visual Comparison

Figure 4.20 provides a comparative visualization of the reconstructed images obtained by Deep JSCC, ADJSCC, SwinJSCC, NTSCC, and the proposed rate-adaptive URC-JSCC across two SNR levels (2 dB and 10 dB). Each reconstruction includes its corresponding PSNR and SSIM values, and enlarged patches highlight fine-grained details such as foliage textures, riverbed structures, and tree boundaries. The results clearly demonstrate that URC-JSCC consistently produces higher-quality reconstructions, owing to its adaptive control of transmission rate based on channel SNR and content complexity.

At 2 dB, Deep JSCC struggles to suppress channel noise, producing heavily distorted foliage and river textures. The zoomed-in region reveals significant structural collapse and graininess, making fine details nearly indistinguishable. ADJSCC improves stability through SNR-conditioned feature modulation, but artefacts and inconsistent textures persist. SwinJSCC and NTSCC provide clearer reconstructions than Deep JSCC and ADJSCC, with improved structural preservation and reduced local distortion. However, their outputs still show visible texture degradation in the detailed regions. In contrast, URC-JSCC delivers substantially cleaner outputs, preserving vegetation structure, river contours, and colour consistency. The patch highlights illustrate that although the channel is severely noisy, URC-JSCC allocates more resources to highly textured areas, enabling sharper boundaries and reduced noise. This selective rate allocation explains the pronounced SSIM increase relative to the baseline methods.

At 10 dB, Deep JSCC continues to exhibit noise in high-frequency regions, particularly

within the bushes, river rocks, and water ripples. ADJSCC reduces this noise but remains unable to restore crisp textures, leading to blurred or blotchy details. SwinJSCC improves the overall structural quality compared with the CNN-based baselines, while NTSCC further preserves local details and produces cleaner texture regions. Nevertheless, both methods still show some loss of fine-grained sharpness in the enlarged patch. URC-JSCC, by contrast, produces reconstructions with noticeably improved sharpness and texture fidelity. The riverbed stones and mid-range foliage are more accurately represented, and the enlarged patch clearly shows that URC-JSCC maintains local structure without oversmoothing. Across channel conditions, the adaptive rate mechanism prioritizes visually complex blocks with higher sampling density while assigning lower rates to smoother regions; what changes with channel quality is the overall symbol budget, with lower SNR prompting the transmitter to send more symbols, i.e., operate at a higher effective rate, to introduce additional redundancy and maintain robustness. Overall, the visual comparison confirms that URC-JSCC provides the strongest reconstruction quality among the compared methods, achieving higher PSNR and SSIM while better preserving perceptually important image details.

4.5 Overview of the Proposed Rate-Adaptive Compressive JSCC Frameworks

The proposed frameworks in this chapter share a common objective: to integrate learned compressive sampling with an end-to-end JSCC pipeline while enabling explicit control of the transmitted symbol budget under time-varying SNR and bandwidth. Both designs perform blockwise rate selection conditioned on image content and CSI (in the form of SNR feedback), and both employ a two-stage reconstruction strategy that combines an initial linear inversion with a deep refinement network. However, the key distinction is how the rate decision is integrated: RACS-JSCC treats the selected rate mainly as a routing choice among predefined branches, whereas URC-JSCC incorporates the rate choice into the end-to-end design and training objective, enabling more consistent reconstruction under mixed block rates and more direct control of the average symbol budget.

Table 4.6: Qualitative comparison of RACS-JSCC and URC-JSCC.

What to look for	RACS-JSCC	URC-JSCC
Rate adaptivity and control signaling (encoder-side decisions)		
Single-model multi-rate operation (no multiple models)	✓	✓
Explicit protected rate header / control field	×	✓
Rate identification robustness at receiver	Medium (label assumed reliable)	High (protected 2-bit/block header)
Train-time rate-distortion knob	×	✓ (λ in $L_{\text{rec}} + \lambda L_{\text{rate}}$)
Sampling operator and measurement formation (compressive front end)		
Sampling operator family	BCS-style stride- B conv sampler	GSM-style hierarchical linear sampler
Multi-scale feature compaction <i>before</i> encoding	×	✓
Measurement vector formation across blocks	Concatenate block measurements	Concatenate + header; variable M
Variable-length handling (padding/mask)	✓ (implicit)	✓ (explicit; bounded by k_{max})
Latent shaping and fusion (JSCC core)		
Transformer-based global fusion	×	✓
Designed to model cross-block dependencies (mixed rates)	×/ limited	✓
Reconstruction strategy (receiver)		
Rate-aware block demultiplexing/routing	✓ (by label)	✓ (by recovered header)
Initial reconstruction	✓ (linear, , rate-conditioned)	✓ (linear, rate-conditioned)
Deep reconstruction type	CNN refinement	Residual CNN refinement (per rate branch)
Various-rate reconstruction capability	✓	✓ (stronger consistency via fusion + header)
Training supervision and optimization objective		
Single loss on final output (MSE)	✓	×
Rate-regularized training objective	×	✓ (L_{rate})
Encourages economical sampling (lower average rate)	×(emerges implicitly)	✓ (explicit)
Stable gradients through discrete rate decisions	×/ not central	✓ (straight-through estimator)
Novelty / takeaway		
Primary capability	Lightweight blockwise rate-adaptive baseline	Unified rate-controlled design with robust signaling + stronger fusion
Best use-case intuition	Fast multi-rate extension of fixed-rate CS-JSCC	Controlled rate-distortion shaping + heterogeneous-rate robustness

RACS-JSCC provides the baseline rate-adaptive extension of the fixed-rate compressive JSCC pipeline from Chapter 3. Its main contribution is to introduce discrete blockwise rate control by routing each block to one of a small number of predefined sampling ratios, thereby varying the number of measurements (and hence channel symbols) produced by the encoder within a single trained model. This design keeps the overall encoder and decoder structure close to the fixed-rate setting, which makes it simple to implement and helps isolate the benefit of explicit rate control.

URC-JSCC advances the same objective with a more unified rate-controlled design. Compared with RACS-JSCC, URC-JSCC strengthens the sampling front end by replacing the BCS-style block sampler with a hierarchical learned sampling network that performs progressive linear spatial compaction before projecting each block to its measurements. It also enhances the

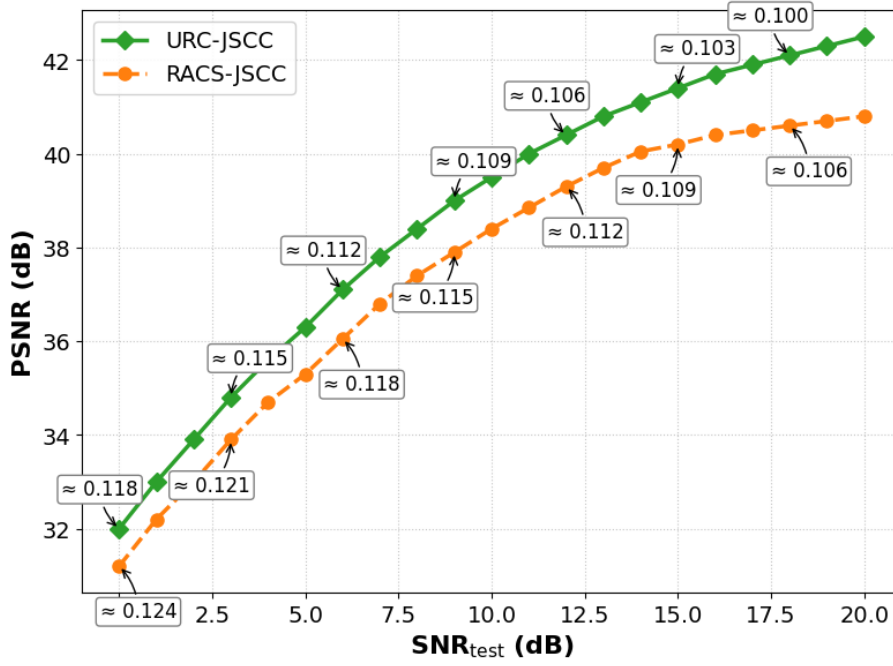


Figure 4.21: PSNR comparison of URC-JSCC and RACS-JSCC on the Kodak dataset over an AWGN channel as a function of SNR_{test} ; annotated boxes indicate the bandwidth ratio (k/n) at selected operating points.

encoder-decoder backbone with CNN-Transformer fusion to better model cross-block dependencies that arise when blocks are sampled at different rates. In addition, URC-JSCC incorporates an explicit rate penalty during training, enabling a direct distortion-rate trade-off that shapes the average sampling ratio in a controlled manner rather than relying only on discrete selection among fixed ratios. Table 4.6 summarizes these differences at a level that complements the detailed descriptions in Sections 4.3 and 4.4, and it clarifies the progression within this chapter from a lightweight rate-adaptive baseline (RACS-JSCC) to a more unified and controllable architecture (URC-JSCC).

Fig. 4.21 compares URC-JSCC and RACS-JSCC over the full test-SNR range on Kodak. Both methods exhibit a smooth and monotonic PSNR improvement as SNR_{test} increases, confirming stable operation under changing channel conditions. URC-JSCC consistently achieves higher reconstruction quality than RACS-JSCC across all SNRs, with the margin becoming more noticeable in the medium-to-high SNR regime where finer textures can be recovered more reliably. The annotated values indicate the effective bandwidth ratio (k/n) used by each method at selected operating points. As expected, both schemes transmit at higher rates when the channel is poor (to introduce additional redundancy) and gradually reduce the transmitted symbol budget

as the channel improves. Importantly, URC-JSCC attains its PSNR gains while maintaining a comparable or lower effective k/n than RACS-JSCC at the same SNR, highlighting that the improvement comes from a more efficient allocation and utilization of measurements rather than simply sending more symbols.

Table 4.7 also summarizes the shift in emphasis from Chapter 3 to Chapter 4. Chapter 3 develops fixed-rate compressive JSCC pipelines where learned sampling and reconstruction are optimized jointly, but the rate is selected before deployment and remains unchanged at run-time; consequently, the transmitted symbol budget is effectively fixed for a given trained model, and improvements are mainly obtained through stronger sampling/reconstruction modules and better latent shaping under a fixed k/n . Chapter 4 moves beyond this limitation by introducing explicit rate-control mechanisms that adapt the effective k/n on demand using both image content and SNR feedback. This enables a single trained system to operate at multiple operating points by redistributing the available rate across blocks and adjusting the overall symbol usage when channel conditions change. As reflected in the criteria of Table 4.7, the focus therefore expands from accuracy at one operating point to rate-distortion behaviour and robustness under bandwidth variability, including the ability to handle heterogeneous-rate blocks within the same image and to shape the rate-fidelity trade-off directly during training.

4.6 Conclusive Remarks

This chapter has extended the fixed-rate deep compressive JSCC framework of Chapter 3 to the rate-adaptive setting, where both bandwidth and SNR vary over time. Two architectures have been introduced and evaluated that integrate learned compressive sampling with channel-aware rate control inside a single trainable model:

- **RACS-JSCC:** The RACS-JSCC framework augments the BCS-JSCC design with a content and SNR aware sampling-rate selection module. Each image is partitioned into blocks and routed to one of three BCS branches operating at low, medium, or high sampling rates, based on fused global and local features and the current SNR. A shared convolutional encoder and a two-stage decoder then operate reliably across the resulting range of

Table 4.7: Comparison between developed frameworks of fixed-rate compressive JSCC (Chapter 3) vs. rate-adaptive compressive JSCC (Chapter 4).

What to look for	Chapter 3	Chapter 4
Problem setting and goal		
Primary goal: improve reconstruction at a fixed operating point	✓	×
Primary goal: operate under variable bandwidth/SNR without retraining	×	✓
Explicitly targets bandwidth variability (rate changes at run-time)	×	✓
Rate handling		
Single fixed k/n per trained model	✓	×
Multiple effective k/n within one trained model (multi-rate)	×	✓
Rate decision uses SNR feedback (CSI at transmitter)	×	✓
Content-aware rate allocation (block-level decisions)	✓ (CS-JSCC)	✓ (core)
Architecture focus		
Learned CS sampling embedded inside JSCC	✓	✓
Key bottleneck addressed: fixed-latent / fixed-rate constraint	×	✓
Designed for heterogeneous-rate blocks within the same image	×	✓
Training objective		
Reconstruction-only training (e.g., MSE / perceptual loss)	✓	✓
Explicit rate-distortion control term during training	×	✓
What the evaluation emphasizes		
Quality vs SNR at fixed rate	✓	✓
Quality vs rate/budget (rate-distortion behavior)	×/ limited	✓
Robustness to rate/SNR mismatch (operation away from training point)	×/ limited	✓

sampling ratios. Experiments on CIFAR-10 and Kodak over AWGN channels show that RACS-JSCC consistently outperforms Deep JSCC and ADJSCC in PSNR and SSIM while transmitting at most the same, and often fewer, channel symbols, demonstrating that blockwise rate adaptation and learned CS sampling can yield substantial bandwidth savings without sacrificing reconstruction quality.

- **URC-JSCC:** The second framework, URC-JSCC, generalizes these ideas into a more powerful architecture that couples hierarchical CS sampling with a hybrid CNN-Transformer backbone and an explicit rate regularization term. Blockwise content and SNR based rate selection is performed as in RACS-JSCC, but each rate branch employs a learned GSM inspired sampling network that captures multi-scale features before projection to measurements. Measurements from all blocks are fused by a CNN-Transformer encoder, power-normalized, and transmitted over the channel. At the receiver, a symmetric decoder with IGDN and Transformer layers, together with initial and deep reconstruction modules, reconstructs each block according to its sampling rate. The rate penalty in the loss function encourages proper use of measurements and leads to smooth variation of the effective bandwidth with SNR. Across CIFAR-10 and Kodak under both AWGN and Rayleigh

fading, URC-JSCC achieves the best rate-distortion performance among all considered methods, providing higher PSNR at significantly lower average symbol-per-pixel ratios than Deep JSCC, ADJSCC, and fixed-rate variants.

Beyond the individual models, several broader insights emerge from the analysis in this chapter. First, integrating learned rate selection with blockwise CS sampling provides a practical mechanism for adapting the measurement density to both visual content and channel quality, which leads to graceful performance degradation and efficient bandwidth utilisation. Second, explicit modelling of the trade off between distortion and sampling ratio through a rate penalty allows the network to learn meaningful operating points, where informative regions are preserved at higher rates while redundant or homogeneous areas are aggressively compressed. Third, hybrid encoder-decoder backbones that combine convolutional processing with Transformer layers are effective at aggregating information across blocks sampled at different rates and at stabilising reconstruction under severe noise and fading. Finally, the experiments confirm that maintaining an initial linear reconstruction stage, followed by a deep residual refinement network, remains beneficial in the rate-adaptive setting, since it simplifies the inversion of the CS operator and improves robustness across SNR regimes.

Overall, the RACS-JSCC and URC-JSCC frameworks establish that deep compressive JSCC can be extended from fixed-rate to rate-adaptive operation without resorting to multiple models or hand-crafted layering strategies. They show that a single learned system can adjust its symbol budget on demand while preserving or improving reconstruction quality relative to state-of-the-art fixed-rate JSCC baselines. At the same time, both frameworks are reconstruction oriented and still target pixel-level fidelity. This motivates the next step, pursued in Chapter 5, where the focus shifts from reconstructing images to communicating only what is required for downstream tasks. The task-oriented TCS-JSCC and FRCS-JSCC architectures developed there build on the compressive and rate-adaptive principles established in this chapter to support efficient and robust face detection and recognition over wireless channels.

Semantic and Task-Oriented Compressive JSCC for Vision over Wireless

5.1 Introduction and Related Works

The fixed-rate and rate-adaptive compressive JSCC frameworks developed in Chapters 3 and 4 demonstrated that integration of CS within JSCC framework can substantially enhance image transmission robustness and accurate image reconstruction under limited bandwidth and noisy conditions. However, many wireless and edge intelligence applications do not require pixel-accurate reconstruction. In tasks such as detection, recognition, and visual reasoning, the objective is to extract and convey only the information necessary for reliable inference rather than reproduce the entire image. Transmitting pixel-level data in such scenarios not only consumes excessive bandwidth but also increases latency, energy cost, and privacy exposure. This chapter extends the JSCC paradigm toward task-oriented communication, in which the transmitter encodes and sends only semantically meaningful representations that directly support downstream tasks. Two specialized frameworks are developed: TCS-JSCC for various tasks (case study: face detection, license plate detection and image classification) and FRCS-JSCC for face recognition. Both are designed to operate efficiently across AWGN and slow Rayleigh fading channels while maintaining high inference accuracy.

Task-oriented and semantic communication represent a fundamental departure from Shan-

non's traditional paradigm of reliable bit transmission [171]. Instead of minimizing reconstruction distortion, the goal is to maximize the success of a downstream task such as classification or recognition [9]. Deep learning-based JSCC provides a natural foundation for this paradigm, enabling end-to-end training of the encoder, channel, and task network under a unified objective. Early semantic communication systems demonstrated significant bandwidth savings for image classification, captioning, and object detection tasks by transmitting task-relevant features rather than raw pixels [172], [173], [174], [175]. These models were often built upon the Information Bottleneck (IB) principle, which encourages compact but sufficient representations for the target task. However, most of these systems employ fixed-dimensional bottlenecks and uniform feature compression, ignoring the content and channel variability that characterize practical wireless environments [132], [176]. Recent surveys emphasize that this lack of adaptive rate control and robust feature recovery remains one of the key limitations restricting the real-world deployment of semantic communication systems [9], [35], [177], [178], [179].

Recent advances in adaptive and importance-aware JSCC attempt to address these shortcomings by integrating channel and content awareness into the encoding process. Models such as I-JSCC [180] and IRCSC [181] allocate more communication resources to semantically important features or adjust the latent dimension according to the channel SNR. These mechanisms improve efficiency and robustness, but they still operate with dense latent representations and global attention mechanisms, offering limited spatial granularity and interpretability. Moreover, once trained, their compression ratio remains fixed, making them unsuitable for dynamic bandwidth conditions.

A related research direction focuses on transmitting intermediate CNN features or attention-selected regions for detection and recognition tasks [182], [183], [184]. These approaches reduce bandwidth relative to pixel-domain transmission but typically employ fixed-size feature tensors and require computationally heavy decoders to reconstruct semantic maps at the receiver. A continuing challenge in task-oriented JSCC is maintaining semantic consistency under noisy and variable channel conditions while keeping computational and communication costs low [9]. Approaches inspired by the Robust Information Bottleneck (RIB) framework introduce regularization to balance informativeness and redundancy, while diffusion-based JSCC employs

generative denoising to restore semantic features after noisy transmission [185], [186], [187]. Although effective, these models often require large architectures or complex training procedures.

Building on this background, the main contributions of this chapter are as follows. First, it introduces TCS-JSCC, a task-oriented compressive JSCC model that performs content and SNR-aware block-wise sampling for face detection. This design eliminates the reconstruction stage and allows direct inference on received features, achieving high detection accuracy with significantly reduced bandwidth. Second, it proposes FRCS-JSCC, a feature-domain JSCC framework for face recognition that integrates learned rate control, SNR-conditioned decoding, and embedding refinement to maintain identity consistency under channel noise. Together, these architectures unify CS and task-oriented communication, delivering superior task performance and bandwidth efficiency under diverse channel conditions compared to the baselines.

The remainder of this chapter is organized as follows. Section 5.2 presents the TCS-JSCC framework for face detection, which firstly defines its system model, then explains its architecture, and lastly demonstrates its results. Section 5.3 introduces the FRCS-JSCC system model and proposed architecture for face recognition and discusses its performance under varying SNR and bandwidth conditions. Section 5.4 compares the developed TCS-JSCC and FRCS-JSCC from various aspects and discusses about their specifications. And at last, section 5.5 summarizes the key insights from this chapter.

5.2 TCS-JSCC: Task-Oriented JSCC with Adaptive Deep Compressed Sensing

The first proposed task-oriented framework, termed TCS-JSCC, extends the principles of deep CS and joint source-channel coding to enable efficient, task-driven wireless transmission. Unlike conventional reconstruction-based systems, TCS-JSCC is designed to communicate only the information necessary for accurate tasks and applications such as object or face detection, thereby eliminating the need for full image reconstruction at the receiver. The main contributions of TCS-JSCC are as follows. First, it introduces a task-oriented JSCC pipeline that integrates block-wise CS into end-to-end training, transmitting only task-relevant features without image

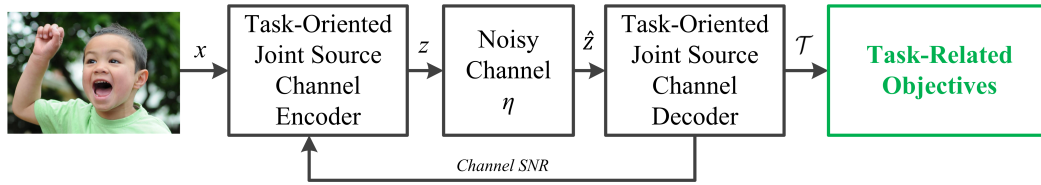


Figure 5.1: System model of proposed TCS-JSCC.

reconstruction. Second, it designs a rate-adaptive CS scheme that enables block-wise sampling guided by both semantic importance and channel SNR. Third, it integrates SNR-aware FiLM for efficient channel adaptation, making the framework practical for downstream tasks directly.

5.2.1 System Model

Fig. 5.1 illustrates the proposed framework. The encoder extracts task-specific features from the input RGB image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, compresses them using SNR-adaptive rates, and performs power normalization, preparing symbols for transmission over the wireless channel. The wireless channel is modeled as a block-fading AWGN channel. The received symbols $\hat{\mathbf{z}}$ are given by:

$$\hat{\mathbf{z}} = h\mathbf{z} + \mathbf{n} \quad (5.1)$$

where $\mathbf{z}$ is the channel input and $\mathbf{n} \sim \mathcal{CN}(0, \sigma_n^2 \mathbf{I})$ is a vector of i.i.d. noise samples drawn from a circularly symmetric complex Gaussian distribution, with σ_n^2 being the noise variance. The fading channel coefficient $h \sim \mathcal{CN}(0, 1)$ is a complex random variable with a Rayleigh-distributed amplitude and a uniformly distributed phase. The channel SNR is estimated at the receiver and is fed back to the transmitter for dynamic SNR-aware rate adaptation.

At the receiver, the task-specific features are decoded and aggregated to execute the task objectives. These objectives can be achieved with a task-oriented network responsible for producing the final task output. It must be noted that the receiver directly processes semantic features for downstream tasks without reconstructing the original images.

5.2.2 Model Architecture

The model architecture integrates a content and SNR-aware block-wise compressive sampling strategy with a task-adaptive encoder-decoder pipeline that directly supports a detection head. Through this design, TCS-JSCC concentrates bandwidth on visually critical regions, suppresses redundancy in homogeneous areas, and maintains robustness under varying channel conditions.

Encoder Architecture

The encoder, as shown in Fig. 5.2, consists of five main components: preprocessing, feature extraction, rate-adaptive CS, attention-based encoding, and SNR-aware feature adaptation. A 2D convolutional layer is denoted as $\text{Conv2D}(c_{in}, c_{out}, k, s, p)$ and a fully connected layer as $\text{FC}(c_{in}, c_{out})$, where c_{in} and c_{out} represent the number of input and output channels, respectively, k is the kernel size, s is the stride, and p is the padding.

During the preprocessing, the input image $\mathbf{x}$ is globally normalized to the range $[0, 1]$ and then divided into non-overlapping blocks of size $B \times B$, where $B = 16$ to balance computational complexity and feature granularity. Following the preprocessing step, a lightweight modified version of MobileNetV2 network [188] extracts features from each block. The architecture of the feature extraction network is as follows:

- Initial convolution: $\text{Conv2D}(3, 32, 3, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU6}$.
- First depthwise block: $\text{depthwise Conv2D}(32, 32, 3, 1, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU6} \rightarrow \text{pointwise Conv2D}(32, 64, 1, 1, 0) \rightarrow \text{BatchNorm}$.
- Second depthwise block: $\text{pointwise Conv2D}(64, 384, 1, 1, 0) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU6} \rightarrow \text{depthwise Conv2D}(384, 384, 3, 2, 1) \rightarrow \text{BatchNorm} \rightarrow \text{ReLU6} \rightarrow \text{pointwise Conv2D}(384, 128, 1, 1, 0) \rightarrow \text{BatchNorm}$.
- Feature aggregation: $\text{GlobAvgP} \rightarrow \text{Conv2D}(128, 256, 1, 1, 0) \rightarrow \text{ReLU6}$.

The extracted features (output of the feature extraction module) for the i -th block are denoted as $\mathbf{f}_i \in \mathbb{R}^{256 \times 1}$. ReLU6 is used in the lightweight MobileNetV2-based feature extractor because it is the standard activation used in MobileNet-style efficient convolutional networks. Its bounded

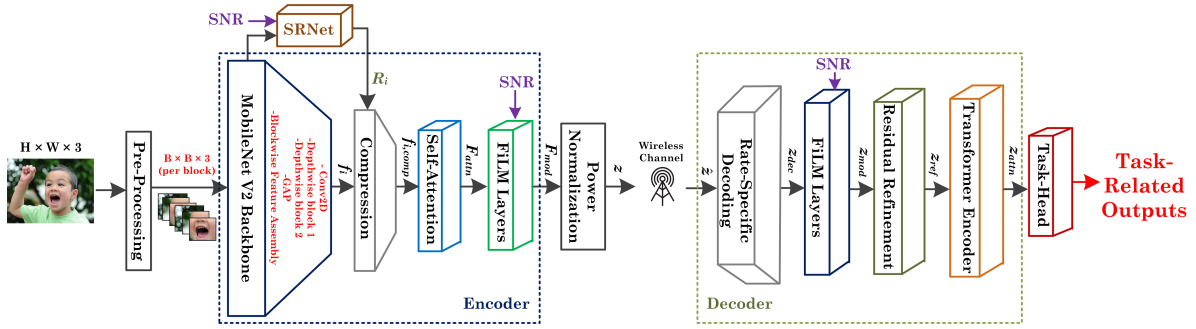


Figure 5.2: Block diagram of the proposed TCS-JSCC framework and architecture design of the encoder and decoder networks.

output range improves numerical stability and is suitable for low-complexity feature extraction, while still providing sufficient nonlinearity for extracting task-relevant local features from each image block. This is particularly useful in TCS-JSCC because the extracted features are later compressed and transmitted through a noisy channel, where stable feature magnitudes are beneficial.

To adaptively compress image regions, a multi-rate CS scheme is proposed to dynamically select sampling rates for each feature vector $\mathbf{f}_i$ from a discrete set $\mathcal{R}$ of available sampling rates based on task-related feature importance and channel conditions. Specifically, a module called SRNet (Figs. 5.2 and 5.3) is employed to predict the optimal compression rate $R_i \in \mathcal{R}$ for the i -th feature, determining how aggressively the i -th feature should be compressed. SRNet consists of a fully connected layer FC(257, 128) followed by a ReLU and a second fully connected layer FC(128, $|\mathcal{R}|$), where $|\mathcal{R}|$ is the number of available sampling rates, followed by a Softmax layer. The ReLU activation in SRNet introduces a simple nonlinearity that allows the rate-selection network to model the relationship between semantic feature importance, SNR, and the preferred sampling rate. Softmax is used at the output because the rate decision is formulated as a discrete selection problem over the candidate sampling rates in $\mathcal{R}$. It converts the output logits into a probability distribution, enabling both interpretable rate selection and differentiable training through the rate-regularization loss.

SRNet takes $\mathbf{f}_i$ and the channel state information (represented by the SNR value) as input, and outputs a probability distribution $\mathbf{r}_i$ across the available rates in $\mathcal{R}$. The sampling rate R_i for $\mathbf{f}_i$ is then chosen as the one with the highest probability in $\mathbf{r}_i$.

Unlike fixed or global rate allocation strategies, which sub-optimally allocate similar rate

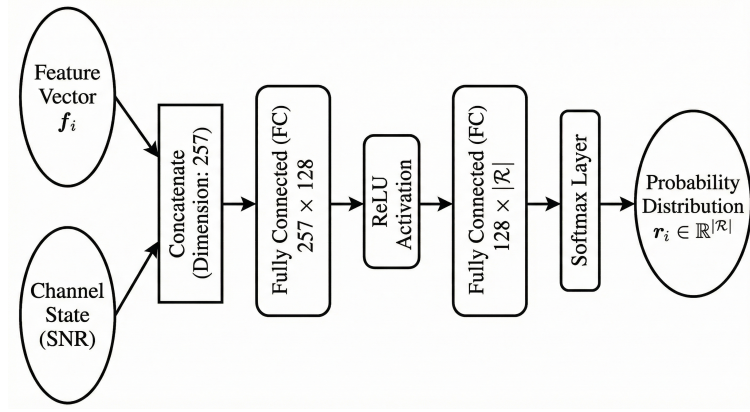


Figure 5.3: Block diagram of the proposed SRNet network.

budget to background and salient regions [189], SRNet learns to allocate higher sampling rates to edge-rich blocks at low SNR and lower sampling rates elsewhere, improving task accuracy for the same bandwidth. Furthermore, SRNet uses a single two-layer network to amortize rate selection, providing comparable adaptivity at a fraction of the computational complexity of policy-network based rate controllers that add substantial parameters and latency.

Given the predicted sampling rate R_i , each feature is compressed using learned projection matrices that map high-dimensional feature vectors into a lower-dimensional space:

$$\mathbf{f}_{i,\text{comp}} = \Phi_{R_i} \cdot \mathbf{f}_i + \mathbf{b}_{R_i}, \quad \Phi_{R_i} \in \mathbb{R}^{\lfloor R_i \cdot 256 \rfloor \times 256} \quad (5.2)$$

where $\mathbf{f}_{i,\text{comp}}$ is the compressed feature vector for the i -th block, $\mathbf{b}_{R_i}$ is the learnable rate-specific bias vector for rate $R_i \in \mathcal{R}$ and Φ_{R_i} is a rate-specific projection matrix. The rate-specific projection matrices and biases are learned jointly with all other network components during the end-to-end training.

The compressed feature vectors $\mathbf{f}_{i,\text{comp}}$ are then aggregated into a tensor $\mathbf{F}_{\text{comp}}$ maintaining the original block-wise structure and further refined through a self-attention mechanism, allowing the model to weigh the importance of different feature vectors relative to each other. This is implemented using a standard scaled dot-product attention module. The compressed feature tensor $\mathbf{F}_{\text{comp}}$ is projected using 1D convolutional layers to generate Query (Q), Key (K), and

Value (V) representations. The output of the attention module is then computed as:

$$\mathbf{F}_{\text{attn}} = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (5.3)$$

where d_k is the dimension of the key vectors. It should be noted that scaled dot-product attention over spatial blocks are applied using convolution-based Q, K, V projections. This allows the network to model long-range dependencies between feature blocks, ensuring that the most task-relevant information is prioritized for transmission. To improve gradient flow and combine the original features with the refined ones, a residual connection is added.

To improve robustness to the channel noise, the attention-refined feature vector $\mathbf{F}_{\text{attn}}$ undergoes an SNR-aware encoding process. Specifically, the SNR value (a scalar) is passed through two fully connected layers with ReLU activations, generating the scaling and bias parameters $\gamma_{\text{enc}}(\text{SNR})$ and $\beta_{\text{enc}}(\text{SNR})$, respectively, for FiLM [190]. This process ensures that the scalar SNR value is transformed into channel-specific modulation parameters, enabling robust adaptation to varying wireless channel conditions. The γ_{enc} and β_{enc} parameters are used to adaptively scale and shift the compressed features based on the communication channel conditions. The output of the FiLM layer can be written as

$$\mathbf{F}_{\text{mod}} = \gamma_{\text{enc}}(\text{SNR}) \odot \mathbf{F}_{\text{attn}} + \beta_{\text{enc}}(\text{SNR}) \quad (5.4)$$

where the operator $\odot$ denotes element-wise multiplication. FiLM layer applies channel-wise affine transformations, adapting the magnitude and bias of each feature channel based on the communication channel SNR. FiLM is utilized as a lightweight alternative to channel-attention modules; in TCS-JSCC setting a single scalar (i.e., SNR) controls channel-wise affine modulation, avoiding the memory/computation burden of attention yet delivering effective channel adaptation.

To meet the wireless power transmission constraints, the feature power is constrained via the power normalization rule described in Eq. (3.7). The final encoded representation $\mathbf{z}$ is transmitted over the wireless channel.

Decoder Architecture

The decoder, shown in Fig. 5.2, is designed to directly produce task-specific outputs from the received feature representation, completely bypassing image reconstruction. The architecture consists of a sequential pipeline: rate-specific feature reconstruction, SNR-aware feature modulation, a residual refinement block, a Transformer-based spatial refinement module, and a multi-scale task-oriented prediction network.

First, the received noisy compressed features in $\hat{\mathbf{z}}$ are reconstructed to their original dimensions (256×1) via inverse projections in a rate-specific manner. Specifically, for each projection matrix Φ_{R_i} at the encoder, a corresponding inverse projection matrix is learned at the decoder, to reconstruct features compressed at rate R_i . This operation is the inverse of the encoder's compression step and is vectorized for efficiency, yielding the decompressed feature tensor $\mathbf{z}_{\text{dec}}$.

Next, to mitigate channel-induced noise, a channel-adaptive FiLM layer conditioned on SNR is applied. This layer performs an affine transformation on the decoded features:

$$\mathbf{z}_{\text{mod}} = \gamma_{\text{dec}}(\text{SNR}) \odot \mathbf{z}_{\text{dec}} + \beta_{\text{dec}}(\text{SNR}) \quad (5.5)$$

where $\gamma_{\text{dec}}(\text{SNR})$ and $\beta_{\text{dec}}(\text{SNR})$ are channel-wise scaling and shifting parameters generated from the SNR value via two fully-connected layers. This step produces the channel-modulated feature map $\mathbf{z}_{\text{mod}}$.

Once the features are adapted, a residual refinement block is applied to enhance local spatial details while suppressing noise. The input $\mathbf{z}_{\text{mod}}$ is passed through a feature refinement block which can be written as $\text{Conv2D}(256, 128, 3, 1, 1) \rightarrow \text{BatchNorm} \rightarrow \text{LeakyReLU} \rightarrow \text{Conv2D}(128, 128, 3, 1, 1)$. The two convolutional layers is adopted in the residual branch as the smallest configuration that provides a non-trivial local correction with limited decoder complexity; deeper stacks offer diminishing returns at the cost of higher parameter count, while removing the refinement block degrades performance, particularly at low SNR. Also, LeakyReLU is used in here because the decoded semantic features can contain both positive and negative responses after channel transmission and FiLM modulation. Unlike standard ReLU, LeakyReLU does not completely suppress negative activations, which helps retain weak but potentially useful

semantic information while still providing nonlinear denoising and local feature correction. Then a skip connection, which includes a 1×1 convolution Conv2D (256,128,1,1,0) to match channel dimensions, is added to the output.

This residual structure ensures robust gradient flow and combines features at different levels of abstraction. To model global dependencies across all feature blocks, the output of the residual refinement block $\mathbf{z}_{\text{ref}}$ is then processed by a standard Transformer encoder module. This self-attention mechanism allows the network to capture long-range spatial relationships, producing an attention-aware feature map, $\mathbf{z}_{\text{attn}}$.

Finally, the Transformer block is followed by a task specific network which processes $\mathbf{z}_{\text{attn}}$ in a way that is appropriate for the considered task.

5.2.3 Loss Function

The proposed TCS-JSCC model is optimized end-to-end using a multi-objective loss function $\mathcal{L}_{\text{total}}$, designed to jointly enhance task accuracy, preserve semantic feature information, and ensure efficient rate adaptation:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{feat}} + \lambda_{\text{rate}} \mathcal{L}_{\text{rate}} \quad (5.6)$$

where λ_{rate} is a scalar weight that balances the contribution of the $\mathcal{L}_{\text{rate}}$ component.

Task-specific loss $\mathcal{L}_{\text{task}}$ is the primary loss for the task-related objectives. For example, for a classification task, this can take the form of a categorical cross entropy. In the following subsection, the task-specific loss for this thesis case study scenario is presented which is face detection.

To maintain semantic consistency across the end-to-end communication process, a feature alignment loss $\mathcal{L}_{\text{feat}}$ is introduced between the encoder's semantic features $\mathbf{f}_i$ and the decoder's reconstructed representation $\mathbf{z}_{\text{dec},i}$, where $\mathbf{z}_{\text{dec},i}$ is the decompressed feature for the i -th block. To ensure scale-invariant comparisons and reduce the impact of stochastic channel noise, both

vectors are ℓ_2 -normalized and defined the feature consistency loss as:

$$\mathcal{L}_{\text{feat}} = \frac{1}{N} \sum_{i=1}^N \left[\lambda_{\text{mse}} \|\mathbf{f}_i - \mathbf{z}_{\text{dec},i}\|_2^2 + \lambda_{\text{cos}} \left(1 - \frac{\mathbf{f}_i \cdot \mathbf{z}_{\text{dec},i}}{\|\mathbf{f}_i\| \|\mathbf{z}_{\text{dec},i}\|} \right) \right] \quad (5.7)$$

where N is the number of blocks. The first term preserves numerical consistency via MSE, while the second promotes angular alignment via cosine similarity. The weights λ_{mse} and λ_{cos} balance the two objectives. This formulation encourages the decoder to reconstruct meaningful semantic features despite compression and channel distortion.

To train SRNet for adaptive rate allocation, a self-regularized formulation is utilized that encourages the network to assign sampling rates based on both feature importance and channel conditions, without requiring explicit rate labels. A rate regularization loss is defined that penalizes unnecessary bandwidth usage while preserving task-critical information as:

$$\mathcal{L}_{\text{rate}} = \frac{1}{N} \sum_{i=1}^N \sum_{R \in \mathcal{R}} \mathbf{r}_i(R) R \quad (5.8)$$

where $\mathbf{r}_i(R)$ is the predicted distribution over $\mathcal{R}$ and $\sum_{R \in \mathcal{R}} \mathbf{r}_i(R) R$ is the expected sampling rate for the i -th block under $\mathbf{r}_i(R)$. The loss term $\mathcal{L}_{\text{rate}}$ is differentiable with respect to $\mathbf{r}_i(R)$ and allows gradient backpropagation during training.

5.2.4 Results and Discussion

A) Face Detection Task

The proposed TCS-JSCC framework is firstly evaluated on the task of face detection, which involves localizing human faces in images by predicting bounding box coordinates and associated confidence scores. This task is well-suited to the TCS-JSCC approach, as it requires retaining task-relevant semantic representations under varying channel conditions. It should be noted that the face detection task considered here is not a closed-set identity recognition problem. The detector is trained to distinguish face regions from background and to estimate their bounding boxes, rather than to classify the identities of the detected people. Therefore, the method can detect unseen identities at test time, provided that their visual appearance remains within the

general face distribution learned during training. Extending TCS-JSCC to explicit open-set object detection would require additional mechanisms such as uncertainty estimation, unknown-class rejection, or training with a wider set of object categories.

A lightweight task-oriented network is utilized to process $\mathbf{z}_{\text{attn}}$ at the decoder and generate the task-related output. To handle faces at various scales, first, a multi-scale fusion module is employed. This module uses three parallel convolutional layers with kernel sizes 1×1 , 3×3 , and 5×5 to extract features at multiple receptive fields and create a rich, multi-scale feature representation. This representation is then fed into two parallel prediction heads: a regression head that predicts the four bounding box coordinates for each location, and a classification head with a sigmoid activation that predicts the corresponding confidence score.

The task-specific term $\mathcal{L}_{\text{task}}$ of the loss function in Eq. (5.6) ensures accurate classification and localization of faces and is defined as:

$$\mathcal{L}_{\text{task}} = \mathcal{L}_{\text{cls}} + \lambda_{\text{IoU}} \mathcal{L}_{\text{IoU}} \quad (5.9)$$

where $\mathcal{L}_{\text{cls}}$ is a classification loss and $\mathcal{L}_{\text{IoU}}$ is a bounding box regression loss. The classification loss uses binary cross-entropy to distinguish face regions from the background. For each prediction $\hat{y}_i$ and ground truth label $y_i \in \{0, 1\}$, it is defined as:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (5.10)$$

For precise localization, the GIoU loss is utilized. GIoU improves IoU by incorporating the smallest enclosing box, providing meaningful gradients even when boxes do not overlap. It is applied only to positive samples (where a face is present).

Evaluation on WIDER Face dataset

The proposed TCS-JSCC model is evaluated on the WIDER Face dataset [133] under simulated wireless channels with SNR values from 0 to 20 dB. Unlike conventional schemes trained at a fixed SNR, a curriculum learning strategy is adopted: training begins at high SNR (20 dB) and gradually introduces lower SNRs until 0 dB, which stabilizes convergence and improves

robustness across conditions. The model is trained using Adam optimizer with learning rate 10^{-4} , batch size 32, and weight decay 10^{-5} . During inference, the SRNet dynamically selects block-wise sampling rates $R_i \in \{0.2, 0.4, 0.8\}$ based on feature importance and channel SNR, enabling adaptive and task-oriented compression. The numerical values used for all loss function hyperparameters in subsection 5.2.3 are following fixed values:

$$\lambda_{\text{rate}} = 0.1, \quad \lambda_{\text{mse}} = 0.5, \quad \lambda_{\text{cos}} = 0.5, \quad \lambda_{\text{IoU}} = 1.0 \quad (5.11)$$

and the bounding box regression loss term in $\mathcal{L}_{\text{IoU}}$ uses $\lambda_{\text{cls}} = 1.0$.

The proposed TCS-JSCC is benchmarked against three well-established baseline schemes. The first is the traditional separate source and channel coding scheme (SSCC), which consists of JPEG compression, rate 2/3 LDPC encoding and 16-QAM modulation followed by Haar Cascade detection [191]. The second benchmark integrates Deep JSCC [6] with MTCNN [192] and the third benchmark is an importance-aware scheme based on IRCSC [181] that transmits only top features via an SNR-dependent rate policy, using a fine-tuned MTCNN detector. Deep JSCC is trained end-to-end across SNRs using Adam optimizer with learning rate 10^{-4} and batch size of 32. MTCNN applies confidence thresholds of 0.9 for bounding boxes and 0.7 for landmarks.

To show the performance of TCS-JSCC, firstly F1-score versus SNR is reported. As shown in Fig. 5.4, TCS-JSCC significantly outperforms all baselines across all SNR levels. The F1-score of TCS-JSCC increases steadily from 0.705 at 0 dB to 0.931 at 20 dB, demonstrating robust performance under severe channel noise. This improvement is attributed to the proposed adaptive compression strategy, which prioritizes task-relevant features while maintaining bandwidth efficiency. Both Deep JSCC + MTCNN and SSCC follow a similar trend but remain consistently below TCS-JSCC, indicating the limitations of full-image reconstruction under challenging conditions. Also, it can observe that IRCSC+MTCNN improves over DeepJSCC+MTCNN but does not attain the performance of TCS-JSCC, demonstrating the additional gain achieved by our block-wise rate allocation and SNR-conditioned modulation.

In Fig. 5.4, the average bandwidth compression ratio for the proposed TCS-JSCC scheme is also reported, which is the ratio of the number of channel uses per image (i.e., number of

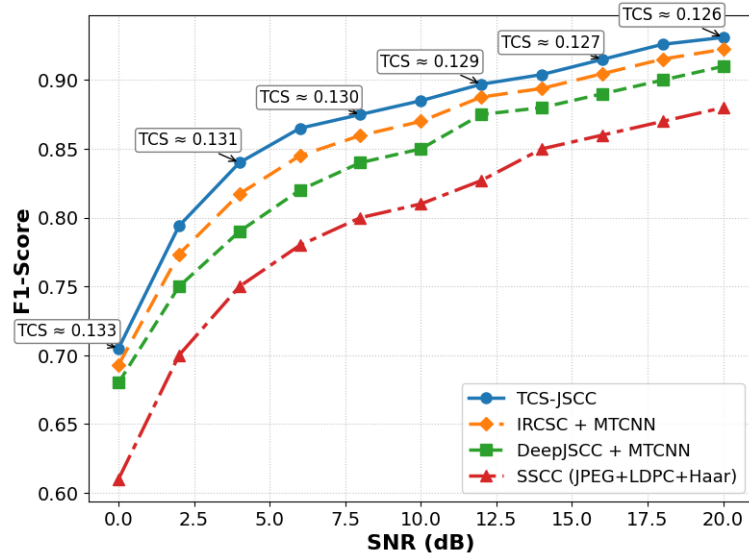


Figure 5.4: F1-score (y-axis), and bandwidth compression (boxed numbers) against varying SNR values [0-20 dB] for TCS-JSCC vs. benchmark methods.

transmitted symbols per image) divided by the number of source symbols, which in case of RGB images is $H \times W \times 3$. Both Deep JSCC + MTCNN and SSCC baseline schemes utilize a fixed bandwidth compression ratio of $1/6$ (≈ 0.166). However, the IRCSC+MTCNN baseline employs an SNR-dependent importance-aware rate control policy; so for fair comparison, its average bandwidth ratio is matched to the one of TCS-JSCC. It is evident from the reported values that TCS-JSCC consistently achieves superior performance with lower bandwidth consumption compared to the baselines across the entire range of SNR values. This is attributed to the TCS-JSCC rate-adaptive CS-enabled sampling strategy, which samples blocks containing non-task-relevant information like background at lower rates while concentrating rate on salient blocks. Furthermore, the reduction of the bandwidth compression ratio as SNR increases confirms that our SNR and rate-adaptive sampling strategy learns to compress task-relevant features more aggressively and adds less redundancy to protect the transmitted symbols against noise as the channel conditions improve, further improving bandwidth efficiency.

Beyond the F1-score versus SNR curve, precision and recall versus SNR curves are reported. These metrics are important because F1-score alone does not distinguish whether degradation at low SNR is dominated by missed detections (recall drop) or false alarms (precision drop). Under severe noise, a method can degrade primarily by missing true faces, which reduces recall, or by producing more false positives, which reduces precision. So with these curves, it would

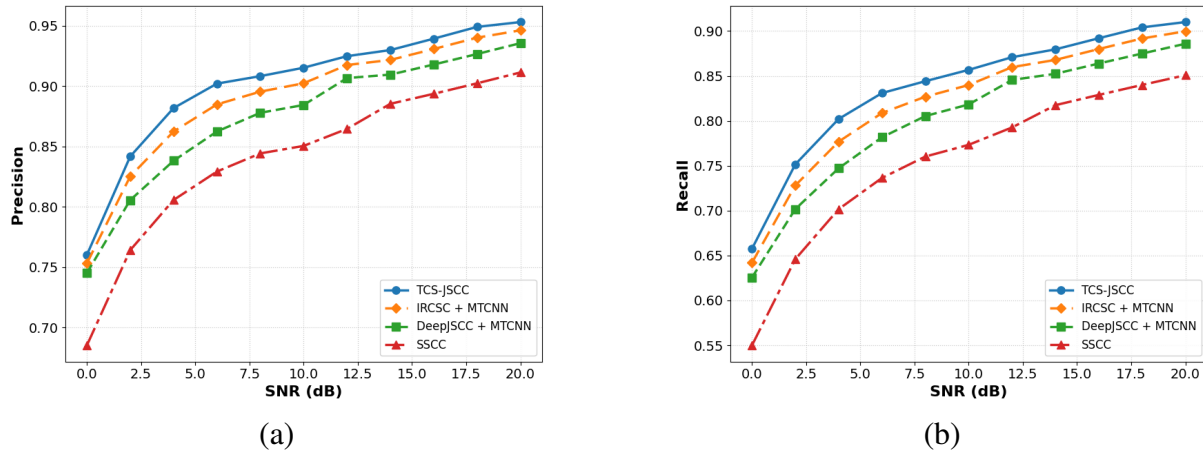


Figure 5.5: Performance metrics of TCS-JSCC for face detection task across SNR levels: (a) precision and (b) recall.

be possible to see which error mode dominates as SNR changes. To this end, the precision and recall metrics versus SNR are shown in Fig. 5.5. In these plots, the proposed TCS-JSCC achieves consistently higher precision and recall across the full SNR range, with the largest gap in the low and moderate SNR regime. The new baseline IRCSC+MTCNN is also included in these figures, and its behaviour is consistently between DeepJSCC+MTCNN and TCS-JSCC. This is aligned with the design intent of IRCSC as an importance-aware rate control scheme, while still highlighting the additional gain provided by our joint block-wise rate allocation and SNR-aware modulation.

Since the proposed TCS-JSCC framework relies on block-wise content-aware and SNR-aware sampling, a histogram-style summary that reports the percentage of blocks assigned to each sampling rate in $\mathcal{R} = \{0.2, 0.4, 0.8\}$ at representative SNR values is shown in Fig. 5.6. In this figure, the empirical distribution of SRNet decisions is reported which can provide a direct and quantitative view of how the overall rate-selection policy shifts with channel quality. This figure shows the number of blocks sampled at each rate divided by the total number of blocks at representative SNR values. The observed trend is consistent with the intended SNR-adaptivity: at low SNR the model assigns a larger fraction of blocks to the low-compression branch ($R = 0.8$) to increase feature robustness to severe noise, while at higher SNR it shifts mass toward higher compression ($R = 0.2$) because less redundancy is needed. Concretely, at 0 dB, 57% of blocks are encoded at $R = 0.8$ and only 7% at $R = 0.2$; at 20 dB, 61% of blocks use $R = 0.2$ and 11% use $R = 0.8$. The medium branch ($R = 0.4$) peaks in the mid-SNR regime (45% at 10 dB),

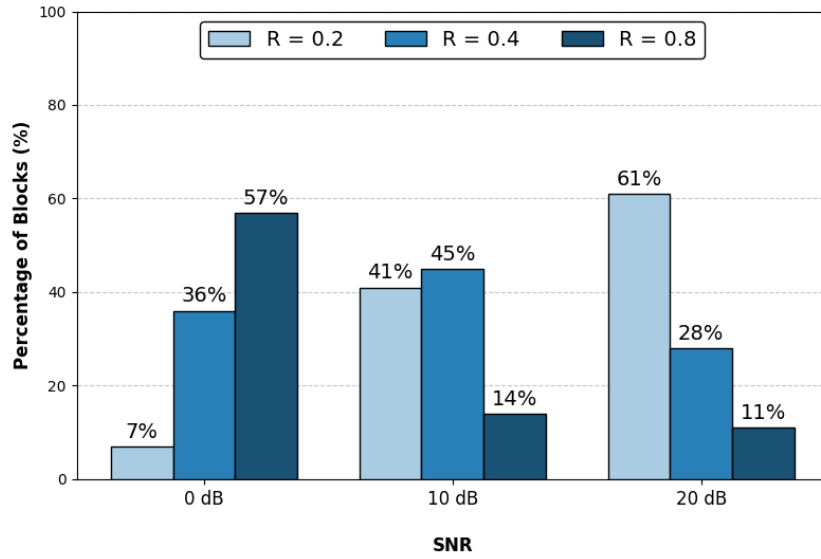


Figure 5.6: Empirical distribution of SRNet rate decisions across blocks at representative SNR values. Each bar reports the percentage of blocks sampled at each rate in $\mathcal{R} = \{0.2, 0.4, 0.8\}$.



Figure 5.7: Illustrative SRNet block-wise sampling rate selection at SNR = 1 dB. Left: clean image. Right: per-block sampling rate map selected from $\mathcal{R} = \{0.2, 0.4, 0.8\}$.

which reflects the transition between robustness-dominant allocation and compression-dominant allocation.

to make the content-aware and SNR-aware rate adaptivity mechanism interpretable at the per-image level and demonstrate the content-wise rate adaptivity, an illustrative SRNet rate-selection map is shown in Fig. 5.7 that visualizes the discrete per-block rate decisions at a low-SNR condition. Fig. 5.7 visualizes the clean image (left) and the corresponding SRNet rate-selection map (right) under a low-SNR test condition. Blocks around structurally salient regions are more frequently assigned lower compression (higher sampling rate), while smoother background regions are predominantly assigned higher compression (lower sampling rates).

Table 5.1: Sensitivity to loss-weight hyperparameters. Each row changes the strength of one loss component while keeping the other components at their default values. For the feature-consistency component, the overall strength is adjusted by setting λ_{mse} and λ_{cos} to the same value (weak or strong), while λ_{rate} and λ_{IoU} remain fixed.

Setting	SNR = 0 dB		SNR = 10 dB		SNR = 20 dB	
	F1	BW	F1	BW	F1	BW
Default: $\lambda_{\text{rate}}=0.10, \lambda_{\text{mse}}=0.50, \lambda_{\text{cos}}=0.50, \lambda_{\text{IoU}}=1.0$	0.705	0.133	0.885	0.130	0.931	0.126
$\lambda_{\text{rate}} = 0.05$ (weaker SRNet)	0.699	0.136	0.881	0.132	0.929	0.128
$\lambda_{\text{rate}} = 0.20$ (stronger SRNet)	0.703	0.132	0.884	0.129	0.930	0.125
$\lambda_{\text{mse}} = \lambda_{\text{cos}} = 0.25$ (weaker feat. consistency)	0.700	0.133	0.883	0.130	0.930	0.126
$\lambda_{\text{mse}} = \lambda_{\text{cos}} = 1.00$ (stronger feat. consistency)	0.702	0.133	0.882	0.130	0.929	0.126

Sensitivity Analysis On Loss-Weight Hyperparameters

To find out the sensitivity of TCS-JSCC performance to loss function hyperparameters, a compact one-factor-at-a-time sensitivity analysis is conducted. Starting from the default configuration, one weight is varied while keeping all other settings fixed, and the followings are reported (i) F1-score at representative SNR values and (ii) the resulting average effective bandwidth ratio (channel uses per image divided by the number of source symbols $H \times W \times 3$). The results are tabulated in Table 5.1. These results indicate that the TCS-JSCC framework is not brittle to moderate changes in the balancing weights. In particular, λ_{rate} mainly influences how reliably SRNet follows the intended rate-allocation supervision, which slightly affects both task performance and the learned bandwidth usage through the resulting branch-selection policy. The feature-consistency weights λ_{mse} and λ_{cos} mainly affect stability under low SNR, and overly weakening or strengthening them produces only minor changes around the default setting.

Component Ablation Study

In order to isolate the contributions of various components in the designed model, a study has been done to find out their attribution of the performance gains. To do so, a controlled ablation study is done which removes each module while keeping the remaining architecture, training schedule, and evaluation protocol unchanged. Specifically, the following variants are considered:

1. Full model (SRNet + FiLM): the proposed TCS-JSCC framework.
2. W/o FiLM: FiLM modulation is removed from both encoder and decoder while SRNet

Table 5.2: Component ablation study isolating SRNet and FiLM across representative SNR values.

Variant	SNR = 0 dB				SNR = 10 dB				SNR = 20 dB			
	Prec.	Rec.	F1	BW ratio	Prec.	Rec.	F1	BW ratio	Prec.	Rec.	F1	BW ratio
Full (SRNet+FiLM)	0.760	0.658	0.705	0.133	0.915	0.856	0.885	0.130	0.953	0.910	0.931	0.126
w/o FiLM	0.748	0.642	0.691	0.133	0.904	0.842	0.872	0.130	0.945	0.900	0.922	0.126
w/o SRNet (fixed sampling rate)	0.735	0.628	0.677	0.134	0.891	0.829	0.859	0.131	0.937	0.892	0.914	0.127
w/o SRNet & w/o FiLM	0.721	0.612	0.661	0.134	0.879	0.816	0.846	0.131	0.928	0.880	0.903	0.127

remains unchanged. In implementation terms, FiLM is disabled so that feature modulation reduces to the identity mapping, while all other components and losses are kept unchanged. This isolates the effect of SNR conditioned feature modulation.

3. W/o SRNet (fixed sampling rate): SRNet is removed and all blocks are sampled at a single fixed sampling rate selected from the same discrete set $\mathcal{R} = \{0.2, 0.4, 0.8\}$. For each SNR, the fixed rate is chosen in such a way that the resulting average bandwidth compression ratio for this variant matches the average bandwidth compression ratio of the full model. This ensures that the study evaluates the effect of the spatially adaptive rate allocation through SRNet under fair channel usage.
4. W/o SRNet and w/o FiLM: both the SRNet and FiLM modules are removed, providing a strict control that quantifies the combined effect of rate adaptivity and SNR conditioning. As in the variant w/o SRNet, in this variant a fixed sampling rate is used to ensure fairness in terms of channel usage.

All ablation variants use the same dataset split, channel model, SNR range, optimizer, and training schedule. The feature extractor, CS projection module, inverse projection, transformer refinement, and the face detector head are kept unchanged. The only changes are the inclusion or removal of SRNet and FiLM. To avoid confounding the comparison by unequal channel usage, the effective bandwidth compression ratio (defined as the average number of transmitted channel symbols divided by the number of image pixels $n = H \times W \times 3$.) is reported for every variant along with precision, recall, and F1 score. The ablation results are shown in Table 5.2 below. The results show consistent performance drops when removing either module (SRNet, FiLM), with the largest degradation when both are removed, confirming that SRNet (spatial rate allocation) and FiLM (SNR-conditioned modulation) provide complementary gains. Specifically,

the following observations can be drawn from these results:

1. First, removing FiLM while keeping SRNet active consistently reduces F1 across all SNR values, with a larger gap at low SNR. This confirms that SNR-conditioned feature modulation improves robustness to channel noise by adapting the internal feature statistics to the current SNR regime.
2. Second, removing SRNet while keeping FiLM active produces a larger performance drop than removing FiLM alone. This indicates that spatially adaptive rate allocation contributes more strongly to task performance than a purely global fixed-rate strategy at matched bandwidth.
3. Third, removing both SRNet and FiLM yields the lowest performance, showing that the two modules provide complementary gains: SRNet improves spatial allocation of measurements, while FiLM improves noise robustness of the learned representation.
4. Importantly, the bandwidth ratios of the ablation variants are matched closely to that of the full model, so the observed gains cannot be attributed to increased channel use. Instead, they reflect more effective use of the available bandwidth through spatial rate adaptivity and SNR-aware feature conditioning.

Design Selection Complexity Analysis

In order to find out the rationale behind using exactly two convolutional layers of refinement module and investigate how the number of convolutional layers affects the final performance, a study has been done. As stated in decoder design details, the refinement module is placed after the rate-specific inverse projection and the SNR-conditioned FiLM layer. At this point, the decoded features can exhibit local artifacts caused by measurement compression, inverse projection mismatch, and channel noise. The refinement module is therefore designed as a lightweight local correction stage that suppresses local distortions before the Transformer-based spatial refinement and the detection head. It is not intended to replace the global modelling capacity of the Transformer, but to provide a low-cost local denoising and smoothing operation that improves stability, particularly at low SNR. So, a shallow residual structure is adopted with

3×3 kernels as a practical compromise. This approach increases the effective local receptive field compared to a single layer, while avoiding the cost of deeper stacks. In TCS-JSCC design, two 3×3 convolutions in the residual branch are the smallest configuration that supports a non-trivial residual correction and preserves a lightweight decoder footprint. Deeper variants increase computation and parameters, and their benefit must be justified by measurable gains.

To do so, the effect of the number of convolutional layers in the refinement block on the performance of TCS-JSCC model have investigated by varying the depth of the residual branch $L \in \{0, 1, 2, 3\}$. Here, $L = 0$ corresponds to the case where the refinement module is removed completely, while a refinement module of depth $L > 0$ uses L stacked 3×3 convolutions in the residual branch with the same skip connection as the one used in the proposed model. All other components of the network, the training schedule, and the evaluation protocol remain unchanged.

In Table 5.3, the F1-score at three representative channel SNR values (low, medium and high SNR regimes) and for different depths L of the residual branch, together with the parameter count (weights only) of the refinement block are reported. The parameter count of the refinement block is calculated as follows. The refinement block consists of

- A residual branch with an initial convolutional layer Conv2D(256,128,3) with $256 \cdot 128 \cdot 9 = 294,912$ parameters;
- $(L - 1)$ subsequent convolutional layers Conv2D(128,128,3) with $128 \cdot 128 \cdot 9 = 147,456$ parameters each;
- A skip connection with a convolutional layer Conv2D(256,128,1) with $256 \cdot 128 \cdot 1 = 32,768$ parameters.

The results in Table 5.3 show that increasing depth beyond two layers does not yield a meaningful additional gain, and the $L = 2$ configuration used in the design provides the best accuracy-complexity trade-off among the tested settings. The results also validate that removing the refinement module ($L = 0$) causes the largest degradation, especially at low channel SNR values, which is consistent with the refinement block suppressing local projection and noise artifacts before the Transformer and detection head.

Table 5.3: Sensitivity to the depth of the residual refinement module. L denotes the number of 3×3 convolutional layers in the residual branch. Parameter counts are exact (weights only). The $L = 2$ row corresponds to the proposed configuration and matches the main results.

Depth L	Params	F1@0dB	F1@10dB	F1@20dB	Notes
0	0	0.690	0.872	0.923	Block removed
1	327,680	0.699	0.880	0.928	Shallow refinement
2	475,136	0.705	0.885	0.931	Proposed
3	622,592	0.704	0.884	0.930	Deeper refinement

Sensitivity Analysis on Block Size

The block size sets the spatial resolution at which SRNet estimates importance and assigns a sampling branch, and it also determines the number of CS projections and inverse projections executed per image. In TCS-JSCC implementation, all experiments use input resolution 256×256 , so an image is partitioned into $N = \left(\frac{256}{B}\right)^2$ non-overlapping blocks of size $B \times B$. Therefore, changing B modifies both the granularity of rate allocation decisions and the computational footprint of the sampling and decoding pipeline.

From a modelling perspective, smaller blocks (e.g., 8×8) provide finer localization of task-relevant regions. This can be beneficial for detection because SRNet can separate foreground boundaries from background with higher spatial resolution. However, it increases the number of blocks by a factor of four relative to 16×16 (from 256 to 1024 blocks per image), which increases the number of rate decisions and the number of per-block inverse-projection operations. This impacts training and inference cost, and it can also make rate maps noisier because decisions are made on smaller receptive fields. Conversely, larger blocks (e.g., 32×32) reduce the number of blocks (to 64 per image), which reduces overhead, but they coarsen importance estimation. In this regime, task-relevant and irrelevant pixels are more frequently mixed within the same block, which forces a shared sampling decision and can reduce the benefit of spatially adaptive rate allocation.

To provide concrete evidence for this trade-off, a sensitivity check around the default 16×16 setting is done by evaluating the performance of TCS-JSCC framework for block sizes $B \in \{8, 16, 32\}$ under the same setting as the experiments presented in the paper. Table 5.4 reports the F1-score at representative SNR points together with the average effective bandwidth ratio of TCS-JSCC. The results support the choice of 16×16 block size as a practical operating

Table 5.4: Block size sensitivity analysis for TCS-JSCC at input resolution 256×256 input. $N = (256/B)^2$ represents the number of blocks per image, while BW denotes the effective bandwidth compression ratio (as defined in the manuscript).

Block Size B	N	SNR = 0 dB		SNR = 10 dB		SNR = 20 dB	
		F1	BW	F1	BW	F1	BW
8×8	1024	0.712	0.136	0.888	0.132	0.933	0.128
16×16	256	0.705	0.133	0.885	0.130	0.931	0.126
32×32	64	0.693	0.131	0.876	0.128	0.924	0.124

point. Block size of 8×8 offers only a marginal improvement at low SNR while increasing the number of blocks per image by $4\times$, whereas 32×32 reduces computational overhead but degrades detection accuracy due to coarser rate allocation.

The results in Table 5.4 align with the expected mechanism of SRNet. With 32×32 blocks, the importance estimation is spatially coarse and cannot isolate localized task cues cleanly, which reduces the advantage of non-uniform sampling. With 8×8 blocks, the granularity increases, but the computational overhead grows substantially because the system executes many more block-wise sampling and inverse-projection operations. The 16×16 configuration therefore provides an effective balance between rate-allocation granularity and computational complexity while maintaining the bandwidth-efficient operating regime targeted by TCS-JSCC framework.

Visual Comparison

Visual comparisons in Fig. 5.8 confirm above-mentioned findings. At 20 dB, TCS-JSCC achieves near-perfect detections with bounding boxes closely aligned with ground truth, while the benchmark methods show slight confidence degradation. At 10 dB, TCS-JSCC maintains robust performance, whereas Deep JSCC + MTCNN exhibits minor misalignment and SSCC suffers from lower confidence. At the highly challenging SNR of 1 dB, TCS-JSCC still detects all faces, albeit with slight localization errors, whereas SSCC fails completely and Deep JSCC detects only a single face. These results confirm the effectiveness of TCS-JSCC in preserving critical semantic features and performing robustly under severe noise. It should be noted that reconstructed images in Fig. 5.8 are shown for visualization purposes only; in deployment, the receiver outputs only detection metadata (bounding boxes and confidence scores) without reconstructing the images.



Figure 5.8: Visual comparison of detection outputs for SSCC, Deep JSCC+MTCNN, and TCS-JSCC at SNR levels of 20 dB, 10 dB, and 1 dB. Yellow boxes indicate ground truth, while red boxes show the detection results produced by each method. The overplotted numbers represent the detection confidence score.

B) License Plate Detection Task:

To provide additional evidence beyond face detection, the proposed TCS-JSCC framework is trained and evaluated the same communication backbone on a license plate detection dataset including Chinese City Plate Dataset (CCPD) [193] and Application-Oriented License Plate Dataset (AOLP) [194]. This task differs from face detection in object geometry, aspect ratio, scale distribution, and surrounding context, making it a meaningful secondary validation of task adaptability. In this experiment:

- The encoder, SRNet, CS projections, SNR-conditioned FiLM modulation, and decoder remain identical to the face detection configuration.

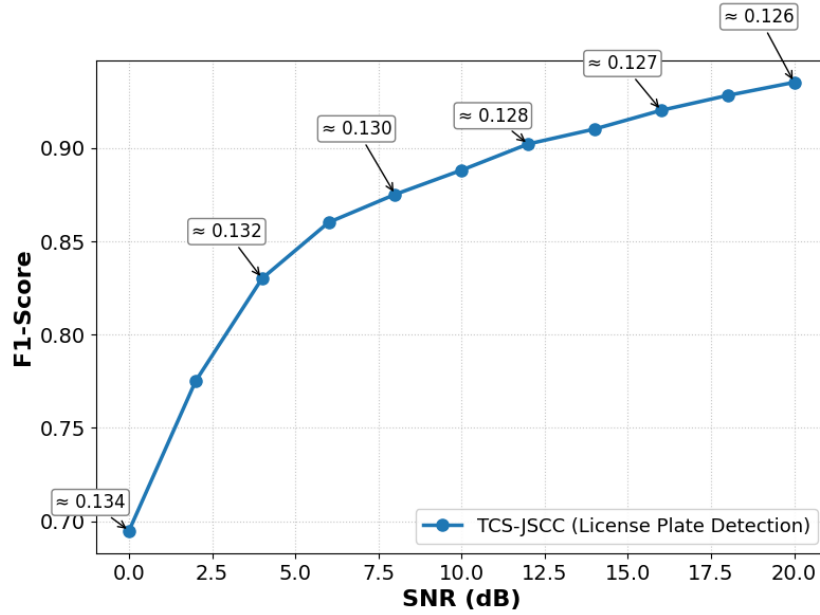


Figure 5.9: F1-score versus SNR for the evaluated methods under the same detection protocol.

- The detection head architecture is unchanged and follows the same bounding-box regression and confidence prediction structure used for face detection.
- The loss formulation is identical to that used in the face detection experiments, combining classification confidence and localization (IoU-based) terms without modification.
- The model is retrained from scratch on the license plate dataset using its ground-truth annotations, and evaluation follows the same IoU-based detection protocol as in the face detection setting.

This setup ensures that only the training data and annotations change, while the communication backbone, detection head structure, and optimization objective remain the same, thereby demonstrating that the proposed framework is task-adaptable rather than face-specific

To show the performance of proposed framework in this task, F1-score versus SNR is reported for in Fig. 5.9. The observed behaviour mirrors the face detection case: the model maintains stable detection performance under moderate SNR degradation and degrades gracefully at very low SNR. For example, at low SNR, the model still detects the primary license plate with correct localization in the majority of samples, while at 20 dB performance approaches the clean-image baseline.

Figure 5.10 shows qualitative detection results at 20 dB, 10 dB, and 3 dB. Even under

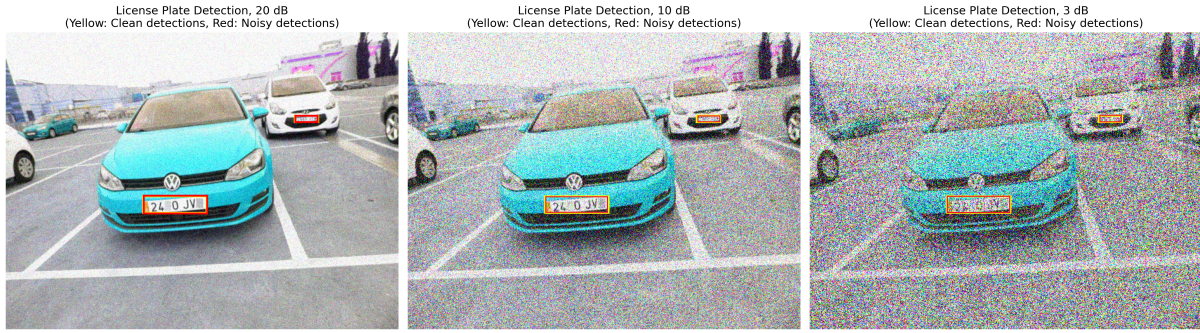


Figure 5.10: License plate detection results at different SNR values. Yellow boxes indicate detections on clean input, red boxes indicate detections under noisy channel conditions.

strong channel noise (3 dB), the model preserves the dominant structural features required for localization. The visual degradation is substantial, yet the predicted bounding box remains aligned with the ground-truth plate region, indicating that the transmitted representation retains task-relevant geometry rather than pixel fidelity.

To further analyze whether the adaptive communication mechanism behaves consistently for this new task, the SRNet block-wise rate selection map for a license plate image at low SNR is also visualized. Figure 5.11 shows the noisy received image alongside the sampling rate map selected from $\mathcal{R} = \{0.2, 0.4, 0.8\}$. Two technical observations can be made from Fig. 5.11:

1. **Spatial concentration:** Blocks overlapping the vehicle front and license plate region are assigned higher sampling density (red and orange blocks), while large background regions such as asphalt and sky are predominantly assigned low sampling density (green blocks). This confirms that SRNet responds to task-relevant structure even when the semantic category differs from faces.
2. **Channel adaptivity:** At low SNR, a larger fraction of blocks near structural edges receive higher rates compared to higher SNR conditions, indicating that the rate controller compensates for channel noise by allocating more measurements to salient regions.

These results demonstrate that the adaptive CS and SNR-aware modulation mechanisms are not tied to facial structure, but instead learn to preserve spatially localized features that are relevant for the active task head. The license plate experiment therefore provides additional empirical support for the task-adaptable nature of the TCS-JSCC backbone.



Figure 5.11: Block-wise sampling rate selection for license plate detection at low SNR. Red denotes high sampling density, orange medium, and green low.

C) Image Classification Task:

To realize that TCS-JSCC framework is general purpose and task-agnostic, an additional validation study is done on a large-scale image classification task using ImageNet-1K dataset [130]. This task differs from the detection ones in two essential aspects. First, the output is a single semantic label rather than a set of bounding boxes. Second, the supervision is a classification objective rather than a localization objective. As a result, it provides a direct check that the proposed communication backbone is not tied to a specific head structure or a detection-specific loss.

In the classification task, the communication side of the pipeline is kept unchanged. Specifically, the encoder feature extractor, the SRNet block-wise rate selection, the rate-dependent projection and inverse-projection operators, and the SNR-conditioned FiLM modulation remain identical to the face detection configuration. The decoder also remains identical up to the shared representation $\mathbf{z}_{\text{attn}}$ produced after the residual refinement and Transformer module. The only change is the terminal task module that consumes $\mathbf{z}_{\text{attn}}$, which is consistent with the task-oriented design principle used throughout the thesis.

To map the decoded block-wise feature representation to an ImageNet label, we use a lightweight linear classification head that aggregates spatial information and predicts a 1000-way class distribution. Concretely, global average pooling (GAP) is applied over the spatial block dimension of $\mathbf{z}_{\text{attn}}$ to obtain a single global feature vector $\mathbf{g}$, and then use a single fully-connected

(FC) layer to produce logits for the 1000 classes:

$$\mathbf{g} = \text{GAP}(\mathbf{z}_{\text{attn}}), \quad \hat{\mathbf{y}} = \text{Softmax}(\text{FC}(\mathbf{g})), \quad (5.12)$$

where $\hat{\mathbf{y}}$ is the predicted class distribution. This linear-probe style head is intentionally lightweight, aligning with the design goal of avoiding heavy receiver-side computation and keeping the receiver comparable in complexity to the detection instantiation.

The primary task loss for this implementation is the standard cross-entropy loss on ImageNet-1K labels:

$$\mathcal{L}_{\text{cls}} = - \sum_{c=1}^{1000} y_c \log(\hat{y}_c), \quad (5.13)$$

where y_c is the one-hot ground-truth label. To strengthen supervision without introducing image reconstruction at the receiver, a label-space distillation signal from a fixed pretrained ImageNet classifier is additionally incorporated. Concretely, for each clean input image, a pretrained ResNet-50 is utilized to produce a teacher class distribution $\mathbf{y}^T$. Then, a Kullback–Leibler (KL) divergence term is added between the teacher distribution and the predicted distribution:

$$\mathcal{L}_{\text{distill}} = \text{KL}(\mathbf{y}^T \parallel \hat{\mathbf{y}}) \quad (5.14)$$

so that the total task loss $\mathcal{L}_{\text{task}}$ becomes:

$$\mathcal{L}_{\text{task}} = \mathcal{L}_{\text{cls}} + \lambda_{\text{distill}} \mathcal{L}_{\text{distill}}. \quad (5.15)$$

The training and evaluation of TCS-JSCC model for the classification task follow the same protocol style as in the face detection experiments. The model is trained end-to-end across the same channel model and SNR range used in the face detection task, using the same optimizer configuration and the same SNR curriculum strategy (training begins at high SNR and gradually introduces lower SNR values to improve robustness).

The overall loss objective has the same structure as in the face detection case, namely task loss plus the feature-consistency term and the rate-related term. The same loss weights reported in the face detection task are used for $\mathcal{L}_{\text{feat}}$ and $\mathcal{L}_{\text{rate}}$, and only $\mathcal{L}_{\text{task}}$ is replaced by the loss in

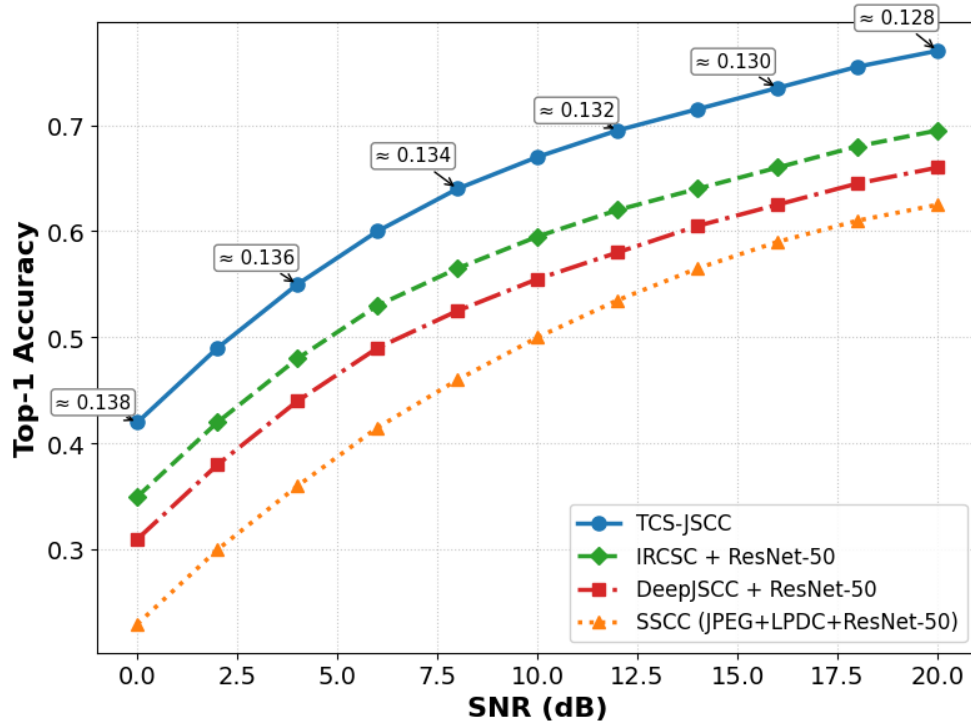


Figure 5.12: Top-1 accuracy versus SNR for the evaluated methods.

Eq. (5.15) where $\lambda_{\text{distill}} = 0.5$.

It should be noted that the ResNet-50 is frozen and used only to provide soft targets during training. It is not used as the receiver-side classifier at inference, and it is not part of the deployed TCS-JSCC pipeline. So in summary, the full TCS-JSCC model (encoder, SRNet, decoder, and the linear head) is trained end-to-end on ImageNet-1K, while the teacher network is frozen. This keeps the method task-oriented because the supervision operates directly in the semantic label space and does not require reconstructing the input image.

For performance evaluation, Top-1 accuracy versus SNR is reported on the standard ImageNet-1K validation set in Fig. 5.12. This metric is the natural analogue of the curves reported for image classification, and it allows a direct interpretation of how semantic inference accuracy degrades with channel quality. For fairness, the same pretrained ResNet-50 is used as the final classifier for SSCC, DeepJSCC, and IRCSC, while for TCS-JSCC it is used only as a frozen teacher during training. The curves in Fig. 5.12 show a consistent accuracy improvement for TCS-JSCC across all SNR values, with the largest gains in the low-to-moderate SNR regime, which is where robustness to channel noise is most critical.

5.3 FRCS-JSCC: Task-Oriented JSCC for Face Recognition over Wireless Networks

The second proposed framework, termed FRCS-JSCC, extends the task-oriented communication paradigm to face recognition. Unlike conventional or reconstruction-centric methods, FRCS-JSCC operates entirely in the feature domain, transmitting identity embeddings instead of pixel-level images. The main contributions of FRCS-JSCC are as follows. First, it introduces a rate-adaptive feature encoder that performs policy-driven compression without requiring multiple models for different SNR levels. Second, it incorporates an SNR-conditioned decoder that dynamically adapts feature reconstruction to varying channel conditions, thereby maintaining identity stability without retraining. Third, it proposes a lightweight diffusion-based refinement module operating directly in the embedding space, which effectively corrects channel-induced distortions and aligns the reconstructed features with the clean identity manifold.

The remainder of this section presents the detailed design and evaluation of FRCS-JSCC. The system model outlining the end-to-end feature-domain transmission and recognition pipeline is described first. Then, the encoder and decoder architectures are explained, followed by the formulation of the loss function. Finally, the experimental results are presented, demonstrating the robustness and efficiency of the proposed framework under various SNR and bandwidth conditions.

5.3.1 System Model

The proposed FRCS-JSCC framework, illustrated in Fig. 5.13, performs face recognition over wireless channels by transmitting identity features instead of pixel-domain images. The system consists of three main components: a task-oriented encoder at the transmitter, a wireless fading channel, and a task-oriented decoder at the receiver. At the transmitter, an input RGB image $\mathbf{x}_{\text{input}} \in \mathbb{R}^{H \times W \times 3}$ (height, width, and number of color channels) is first processed by a face detection and alignment module to obtain a normalized facial crop:

$$\mathbf{x}_{\text{face}} \in \mathbb{R}^{H' \times W' \times 3}. \quad (5.16)$$

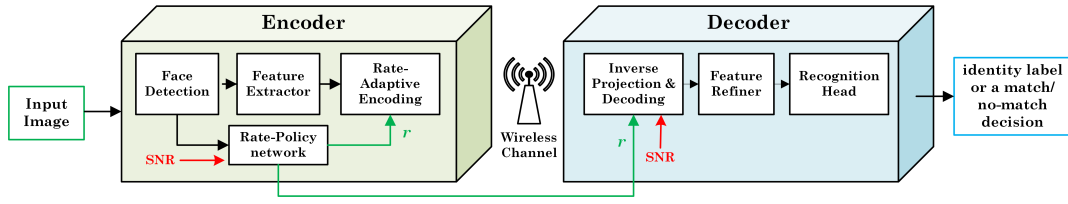


Figure 5.13: System model of the proposed FRCS-JSCC framework

with H' and W' as height and width of cropped image. A task-specific feature extractor $f_{\text{task}}(\cdot)$ maps this crop to an identity embedding

$$\mathbf{f} = f_{\text{task}}(\mathbf{x}_{\text{face}}) \in \mathbb{R}^{d_{\text{feat}}}, \quad (5.17)$$

where d_{feat} denotes the feature dimension. To adapt the transmitted bandwidth, a policy network selects a compression ratio

$$r \in \mathcal{R}, \quad (5.18)$$

where $\mathcal{R}$ is a predefined discrete rate set. The selected ratio determines the number of transmitted channel symbols:

$$d_{\text{ch}} = r d_{\text{feat}}, \quad (5.19)$$

and the embedding is compressed to $\mathbf{z} \in \mathbb{R}^{d_{\text{ch}}}$. An SNR-aware modulation layer uses the channel SNR, i.e., $\mu \in \mathbb{R}$, to condition the transmitted representation, after which power normalization ensures unit average transmit power.

The wireless channel follows the same complex-valued model as in Eq. (3.5). In particular, the encoder outputs a complex channel input vector $\mathbf{z} \in \mathbb{C}^k$, and the receiver observes $\hat{\mathbf{z}}$ as defined in Eq. (3.5) (see Section 3.2). Both AWGN and Rayleigh fading is considered in this framework as well. The average SNR estimate μ is fed back to the transmitter through a reliable link.

At the receiver, a task-oriented decoder reconstructs the semantic vector using:

$$\hat{\mathbf{f}} = f_{\text{dec}}(\hat{\mathbf{z}}, \mu), \quad (5.20)$$

where $f_{\text{dec}}(\cdot)$ denotes the SNR-conditioned reconstruction network. The rate index r is transmitted through a short error-protected control header and enables the decoder to select the correct

inverse projection. Then, a diffusion-based refinement module further improves the reconstructed embedding:

$$\hat{\mathbf{f}}_{\text{refined}} = g_{\text{diff}}(\hat{\mathbf{f}}), \quad (5.21)$$

where $g_{\text{diff}}(\cdot)$ denotes the refiner. The final identity decision is produced by a fine-tuned face recognition head:

$$\mathcal{T} = g_{\text{task}}(\hat{\mathbf{f}}_{\text{refined}}), \quad (5.22)$$

where $\mathcal{T}$ represents the face recognition output. All operations occur entirely in the feature domain without reconstructing the pixel-level image.

5.3.2 Model Architecture

The FRCS-JSCC framework is an end-to-end task-oriented communication system developed for robust face recognition over wireless channels. It integrates three key principles: rate-adaptive compressive encoding, SNR-conditioned decoding, and diffusion-based embedding refinement. The details of the proposed design are presented in the next subsections.

Preprocessing

As shown in Fig. 5.14, prior to encoding, the input RGB image $\mathbf{x}_{\text{input}}$ is processed by a lightweight face detection and alignment pipeline to extract the facial region. A YOLOv3 Tiny detector localizes the face via bounding box estimation [195]:

$$\mathbf{b} = [x, y, w, h], \quad (5.23)$$

where (x, y) denotes the top-left coordinates, and w, h are the width and height of the region of interest. A spatial transformer network (STN) [196] then performs differentiable cropping and warping, to produce a normalized facial image $\mathbf{x}_{\text{face}}$. This step reduces variability due to scale, translation and in-plane rotation, providing a consistent input for subsequent stages. The aligned crop size is set to $H' = W' = 112$ which is consistent with the canonical input resolution used by subsequent modules, including MobileFaceNet for feature extraction [30] and ArcFace for face recognition [31]. These models are pretrained on 112×112 aligned faces, and their architectures

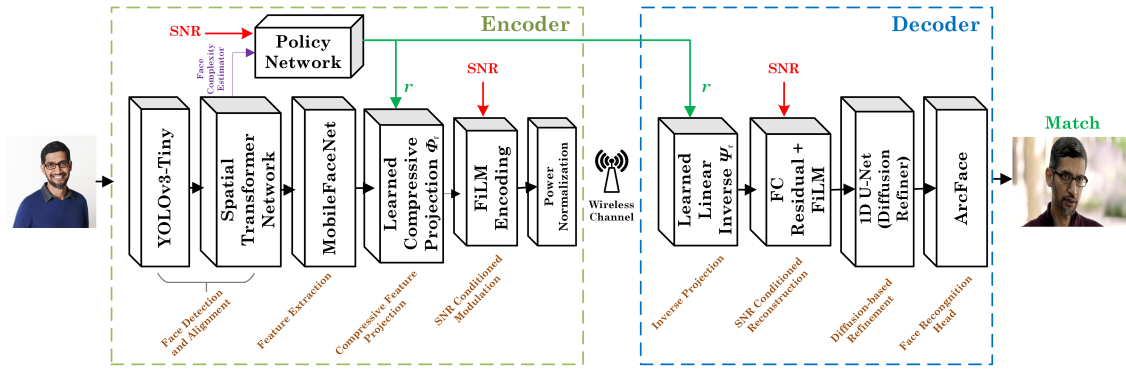


Figure 5.14: Detailed architecture of proposed FRCS-JSCC including encoder and decoder networks

and parameters are calibrated for this input configuration. Using a crop size of 112×112 ensures consistent spatial scaling, maintains identity-discriminative information, and avoids introducing distribution shifts that would otherwise require retraining.

Encoder Architecture

In FRCS-JSCC framework, the encoder maps an input facial image to a compact, identity preserving representation that is suitable for transmission over a noisy wireless channel. As illustrated in Fig. 5.14, it performs three key operations: (i) identity feature extraction, (ii) adaptive rate control with CS-based feature projection, and (iii) SNR-conditioned modulation. In the following, each step is described in details.

Given an aligned facial crop, a task-specific feature extractor $f_{\text{task}}(\cdot)$ is first applied to obtain a compact identity embedding $\mathbf{f}$. In this work, $f_{\text{task}}(\cdot)$ is instantiated using a MobileFaceNet backbone composed of depthwise separable convolutions and linear bottleneck layers, as shown in Fig. 5.15 [197]. This design offers a favorable trade-off between recognition accuracy and computational efficiency, making it suitable for edge deployment. The embedding dimensionality is set to $d_{\text{feat}} = 256$, which is consistent with the standard output configuration of MobileFaceNet and is commonly adopted in ArcFace-based face-recognition systems. The embedding dimensionality is set to $d_{\text{feat}} = 256$, following the MobileFaceNet configuration used in this work. More generally, the proposed rate-adaptive projection can be applied to any feature dimension by selecting the projected channel dimension as $d_{\text{ch}} = \lfloor rd_{\text{feat}} \rfloor$ for each compression ratio $r \in \mathcal{R}$.

The extracted embedding $\mathbf{f}$ is then passed to a compression module with adaptive rate

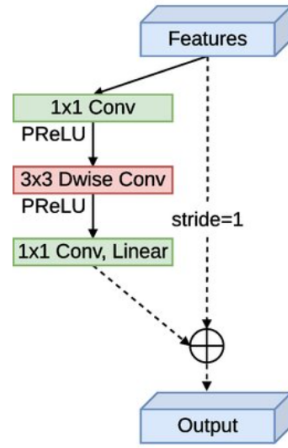


Figure 5.15: Detailed architecture of MobileFaceNet [197].

control, which adjusts the length of the latent representation according to both channel conditions (average SNR) and the visual complexity of the input face. In practical settings, using a fixed compression ratio for all inputs is suboptimal: when the channel quality is poor or for visually complex inputs (e.g., occlusions, facial hair, or accessories), aggressive compression may discard identity-relevant information and degrade recognition performance. Conversely, at high SNR or for visually simple inputs, transmitting the full embedding leads to inefficient use of channel resources. To address this, the encoder adapts the compression level based on both the average SNR and a measure of visual complexity of the input face. Specifically, the complexity of the input face is quantified using the ℓ_1 -norm of its spatial gradient:

$$\mathcal{C}(\mathbf{x}_{\text{face}}) = \|\nabla \mathbf{x}_{\text{face}}\|_1, \quad (5.24)$$

where ∇ denotes the channel-wise gradient operator. Images with stronger edge activity, such as those with facial hair, glasses, or strong wrinkles, typically require a larger feature budget to preserve discriminative details [198].

The compression ratio for each embedding is selected by a policy network which jointly considers the input complexity and the channel conditions. Specifically, let μ denote the average SNR. The pair $(\mu, \mathcal{C}(\mathbf{x}_{\text{face}}))$ is fed to a lightweight policy network $f_{\text{policy}}(\cdot)$ which selects a compression ratio from a predefined discrete set:

$$\mathcal{R} = \{0.25, 0.50, 0.75, 1.00\}. \quad (5.25)$$

The compression ratios set is restricted to four levels to balance adaptability and training stability. Increasing the number of rate levels (e.g., more than six levels) resulted in only marginal performance gains while introducing higher variance and instability during training. The selected set, therefore, achieves an effective trade-off between flexibility and robustness.

The policy network is implemented as a two-layer perceptron with a ReLU activation in the hidden layer and a softmax output over the $K = |\mathcal{R}|$ candidate compression ratios. The network produces logits over the discrete rate set $\mathcal{R}$, which are converted into a probability mass function $p(r)$ via the softmax layer. The selected compression ratio is then given by:

$$r = \arg \max_{r \in \mathcal{R}} p(r). \quad (5.26)$$

Based on the selected rate, the number of channel symbols assigned to the current embedding is computed according to Eq. (5.19). This discrete rate selection provides a balance between adaptability and model complexity, enabling the encoder to respond to variations in channel conditions and input complexity without requiring separate models for different operating points.

During training, the policy network is optimized jointly with the rest of the system using the end-to-end loss. To keep the discrete rate selection compatible with gradient-based learning, a straight-through Gumbel-Softmax estimator is used [199]: the network outputs logits, which are converted to a relaxed categorical distribution during backpropagation, while the forward pass uses the hard $\arg \max$ choice. This allows gradients to flow through the rate selection without changing the inference behaviour. To keep the discrete rate selection differentiable during training, the policy also outputs the softmax probabilities $\{p_k\}_{k=1}^K$ over the candidate ratios in $\mathcal{R}$; these probabilities are later used in the loss function to form a differentiable expected-rate term that encourages the policy to select lower compression ratios when they are sufficient for accurate recognition, while still permitting higher ratios when additional semantic fidelity is needed under harsher channel conditions.

The selected compression ratio directly determines the dimensionality of the transmitted representation and, consequently, the corresponding projection matrix. Specifically, for each rate $r \in \mathcal{R}$, a learned CS projection matrix $\Phi_r \in \mathbb{R}^{d_{\text{ch}} \times d_{\text{feat}}}$ is used to map the embedding $\mathbf{f}$ to a

lower-dimensional representation:

$$\mathbf{z}_0 = \Phi_r \mathbf{f} \in \mathbb{R}^{d_{\text{ch}}} \quad (5.27)$$

Unlike conventional CS approaches that rely on random projections, the matrices Φ_r in the proposed framework are learned jointly with the encoder, decoder, and the face recognition head (ArcFace). As a result, the projection is not purely dimensionality-reducing, but instead optimized to preserve identity-discriminative structure while suppressing nuisance variations. During training, the learned mapping emphasizes embedding components that contribute to accurate identity recognition while attenuating those associated with factors such as illumination, pose or other non-discriminative factors. Furthermore, the learned projection adapts to the selected rate. At low rates, it concentrates identity information into a compact subset of components, whereas at higher rates, it distributes information more broadly to retain finer details. Importantly, this task-aware structure leads to improved robustness compared to random projections, which were observed to cause significant performance degradation at low SNR due to unstable mixing of embedding components.

To adapt the transmitted representation to the channel quality, the encoder conditions its compressed feature vector on the estimated SNR using FiLM modulation. The FiLM layer requires a pair of scale/shift parameters whose dimensionality must match the vector being modulated. Since the encoder operates on a rate-dependent compressed representation, the FiLM dimension depends on the selected compression ratio. Specifically, for each selected rate ($r \in \mathcal{R}$), the active compressed dimension is $d_r = \lfloor rd_{\text{feat}} \rfloor$.

Therefore, the FiLM parameters are not generated with arbitrary lengths for every sample. Instead, each candidate rate has its own lightweight SNR embedding head, and the selected rate (r) determines which head is used:

$$(\gamma_r^{\text{enc}}(\mu), \beta_r^{\text{enc}}(\mu)) = f_{\text{snr},r}^{\text{enc}}(\mu), \quad (5.28)$$

where $f_{\text{snr},r}^{\text{enc}}$ outputs vectors in $\mathbb{R}^{d_r}$, and therefore $\gamma_r^{\text{enc}}(\mu), \beta_r^{\text{enc}}(\mu) \in \mathbb{R}^{d_r}$. The compressed feature for the selected rate is denoted as $\mathbf{z}_0^{(r)} \in \mathbb{R}^{d_r}$. The modulated channel-domain feature is then

obtained as:

$$\mathbf{z}_{\text{enc}}^{(r)} = \gamma_r^{\text{enc}}(\mu) \odot \mathbf{z}_0^{(r)} + \beta_r^{\text{enc}}(\mu), \quad (5.29)$$

where $\odot$ denotes element-wise multiplication. In practice, mini-batches are implemented either by grouping samples with the same selected rate or by padding all compressed vectors to the maximum dimension and applying a binary mask, so that only the first d_r active components are normalized, transmitted, and decoded. The conditioning rescales and shifts the compressed feature vector $\mathbf{z}_0$ prior to power normalization, allowing the encoder to adjust its feature amplitudes in response to the channel state.

At the final stage of the encoder, the active real-valued output corresponding to the selected rate (r) is normalized before transmission. Since the compressed dimension depends on the selected rate, the number of active real-valued components, $d_r = \lfloor rd_{\text{feat}} \rfloor$ is then converted into complex channel symbols by pairing consecutive components as the real and imaginary parts. If (d_r) is odd, one zero is appended before pairing. Therefore, the number of complex channel symbols is

$$m_r = \left\lceil \frac{d_r}{2} \right\rceil. \quad (5.30)$$

The resulting complex vector is denoted by ($\tilde{\mathbf{z}}_{\text{enc}}^{(r)} \in \mathbb{C}^{m_r}$), and it is normalized as:

$$\mathbf{z}^{(r)} = \sqrt{m_r P} \frac{\tilde{\mathbf{z}}_{\text{enc}}^{*(r)}}{\sqrt{(\tilde{\mathbf{z}}_{\text{enc}}^{*(r)})^* \tilde{\mathbf{z}}_{\text{enc}}^{(r)}}}. \quad (5.31)$$

This ensures that

$$\frac{1}{m_r} \left\| \mathbf{z}^{(r)} \right\|_2^2 = P, \quad (5.32)$$

where (P) is the average transmit power per complex channel symbol. Once normalized, the complex joint source-channel coded sequence ($\mathbf{z}^{(r)}$) is transmitted over the communication channel. Each complex symbol is mapped to one channel use, with its real and imaginary parts corresponding to the in-phase and quadrature components of the signal. During transmission, the channel introduces random impairments that corrupt the transmitted symbols. To allow end-to-end optimization of the entire system, the channel is incorporated into the network architecture as a non-trainable layer, modeled using the transfer function described in Eq.(3.5).

Decoder Architecture

At the receiver, the noisy channel symbols are processed by a decoder to reconstruct the transmitted identity embedding without attempting pixel-level image recovery. This design focuses the reconstruction on the semantic objective of identity preservation rather than visual fidelity. As shown in Fig. 5.14, the decoder consists of three main components: (i) a rate-aware inverse projection module, (ii) an SNR conditioned reconstruction network, and (iii) a lightweight diffusion-based refinement stage followed by an ArcFace classification head. These components are described in detail below.

Let $(\hat{\mathbf{z}}_{\text{c}}^{*(r)} \in \mathbb{C}^{m_r})$ denote the received noisy complex channel output corresponding to the transmitted vector $(\mathbf{z}^{(r)})$, where $(m_r = \lceil d_r/2 \rceil)$ is the number of complex channel symbols. Before applying the inverse projection, the receiver converts this complex vector back to a real-valued representation by concatenating its real and imaginary components. If a zero entry was appended before transmission, the corresponding padded component is discarded after reception. This gives the real-valued received vector

$$\hat{\mathbf{z}}_{\text{r}}^{*(r)} \in \mathbb{R}^{d_r}. \quad (5.33)$$

Since the encoder applies a rate-dependent CS projection (Φ_r) , the decoder first maps this real-valued received vector back to the feature space before performing semantic inference. For each rate $(r \in \mathcal{R})$, a corresponding inverse projection matrix

$$\Psi_r \in \mathbb{R}^{d_{\text{feat}} \times d_r} \quad (5.34)$$

is learned jointly with the encoder. At the receiver, the rate index is recovered from the control header, so that the appropriate inverse projection matrix (Ψ_r) can be selected for each received vector. The initial estimate of the identity embedding is then obtained as

$$\hat{\mathbf{f}}_0 = \Psi_r \hat{\mathbf{z}}_{\text{r}}^{*(r)}. \quad (5.35)$$

It is important to emphasize that the pair (Φ_r, Ψ_r) does not perform classical CS encoding

and recovery, nor does Ψ_r invert Φ_r in the strict sense. Since $d_{\text{ch}} < d_{\text{feat}}$ for most operating points, exact inversion is neither possible nor required. Instead, Ψ_r performs a rate-dependent semantic lifting, mapping the compressed channel output vector into a coarse identity-preserving representation in $\mathbb{R}^{d_{\text{feat}}}$. This representation serves as the input to the subsequent SNR-conditioned reconstruction network. As a result, the entire decoding process operates in the feature domain, without reconstructing pixel-level images. The role of Ψ_r is therefore not to reconstruct the original embedding, but to produce a consistent latent representation from which identity structure can be reliably recovered by the subsequent modules of the decoder.

Following inverse projection, the vector $\hat{\mathbf{f}}_0$ still contains distortions introduced by channel noise, fading, and dimensionality reduction. To refine this representation, the decoder employs an SNR-conditioned feature enhancement network $f_{\text{enh}}(\cdot)$ implemented as a stack of fully connected residual blocks with FiLM modulation. A dedicated SNR-embedding network generates the modulation parameters conditioned on the estimated SNR μ :

$$(\gamma^{\text{enh}}(\mu), \beta^{\text{enh}}(\mu)) = f_{\text{snr}}^{\text{enh}}(\mu), \quad (5.36)$$

where $\gamma^{\text{enh}}(\mu), \beta^{\text{enh}}(\mu) \in \mathbb{R}^{d_{\text{feat}}}$.

Let $\mathbf{u}_\ell \in \mathbb{R}^{d_{\text{feat}}}$ denote the input to the ℓ th residual block, with $\mathbf{u}_1 = \hat{\mathbf{f}}_0$. Inside each block, the normalized hidden activation is first computed and then FiLM-modulated as:

$$\mathbf{h}_\ell = \gamma^{\text{enh}}(\mu) \odot \text{LN}(\mathbf{u}_\ell) + \beta^{\text{enh}}(\mu), \quad (5.37)$$

where $\text{LN}(\cdot)$ denotes layer normalization and $\mathbf{h}_\ell$ is the modulated activation used for residual update. The block output produces $\mathbf{u}_{\ell+1}$, and after L stacked blocks, the feature enhancement network outputs

$$\hat{\mathbf{f}} = f_{\text{enh}}(\hat{\mathbf{f}}_0, \mu). \quad (5.38)$$

This SNR conditioning enables adaptive refinement of the latent representation under varying channel conditions, improving robustness to severe distortions while preserving discriminative information when the channel quality is high. ReLU is used inside the fully connected residual reconstruction blocks because this stage performs feature-domain denoising and correction

after inverse projection. The residual structure already preserves the input embedding through skip connections, while ReLU provides a simple and stable nonlinearity for learning corrective transformations. Since the representation is no longer a raw channel-symbol vector but an identity embedding estimate, a standard ReLU is sufficient and keeps the reconstruction network lightweight.

Although $\hat{\mathbf{f}}$ captures the coarse identity structure, channel noise and dimensionality reduction may push the embedding away from the clean feature manifold. To mitigate this effect, a lightweight diffusion-based refinement module is employed, which operates entirely in the embedding space [200]. The refiner performs a small number of reverse denoising steps ($T = 5$ in FRCS-JSCC implementation), resulting in low computational overhead. The reverse diffusion process is initialized from the output $\hat{\mathbf{f}}$ of the SNR-conditioned feature enhancement module as:

$$\hat{\mathbf{f}}_T = \hat{\mathbf{f}}, \quad (5.39)$$

and iteratively refines the embedding from $t = T$ down to $t = 1$. Let α_t denote a predefined noise schedule (a cosine schedule is used for $t = 1, \dots, T$). At each step t , the model predicts the noise residual in the embedding domain:

$$\boldsymbol{\varepsilon}_\theta(\hat{\mathbf{f}}_t, t), \quad (5.40)$$

and updates the estimate using a deterministic denoising diffusion implicit model (DDIM)-style step:

$$\hat{\mathbf{f}}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\hat{\mathbf{f}}_t - (1 - \alpha_t) \boldsymbol{\varepsilon}_\theta(\hat{\mathbf{f}}_t, t) \right). \quad (5.41)$$

The diffusion-based refinement network is implemented as a shallow one-dimensional U-shaped architecture with fully connected layers and skip connections. Since the diffusion process operates on low-dimensional embeddings rather than images, only a few reverse steps are required. The final refined embedding is obtained after the last reverse step as:

$$\hat{\mathbf{f}}_{\text{refined}} = \hat{\mathbf{f}}_1. \quad (5.42)$$

Table 5.5: Implementation-level architecture of FRCS-JSCC.

Module	Architecture	Technical Details / Shapes
Encoder		
Face Detector	YOLOv3-Tiny	Pretrained; 416×416 input; single-class face detector
STN Alignment	CNN + affine grid	$\text{Conv}(3 \rightarrow 16, 7 \times 7, s_2) \rightarrow \text{Conv}(16 \rightarrow 32, 5 \times 5, s_2) \rightarrow \text{Conv}(32 \rightarrow 64, 3 \times 3, s_2) \rightarrow \text{FC}(64 \cdot 14 \cdot 14 \rightarrow 32) - \text{ReLU} - \text{FC}(32 \rightarrow 6)$; Outputs 112×112 aligned face
Identity Extractor	MobileFaceNet	Depthwise-separable CNN; Output: 256-D embedding (L2-normalized); Params: $\approx 0.99\text{M}$
Face Complexity Metric	Gradient energy	$\mathcal{C} = \ \nabla x\ _1$ computed on aligned 112×112 crop (scalar)
Policy Network	MLP($2 \rightarrow 64 \rightarrow 4$)	Inputs: $(\mu, \mathcal{C})$; Hidden: ReLU; Output: Softmax over 4 rates; Rate set: $\{0.25, 0.50, 0.75, 1.00\}$
CS Projection Φ_r	Linear layers (per rate)	Maps $256 \rightarrow d_{\text{ch}}$; $d_{\text{ch}} \in \{64, 128, 192, 256\}$; Bias included; matrices not shared across rates
Encoder FiLM	Rate-specific MLPs $f_{\text{SNR},r}^{\text{enc}}$	Input: scalar SNR; for each selected rate r , outputs $(\gamma_r^{\text{enc}}, \beta_r^{\text{enc}}) \in \mathbb{R}^{d_r}$, where $d_r = \lfloor r d_{\text{feat}} \rfloor$; implemented by rate grouping or padding/masking.
Decoder		
Inverse Projection Ψ_r	Linear layers (per rate)	$d_{\text{ch}} \rightarrow 256$; bias included; separately learned for each rate
Decoder FiLM	MLP($1 \rightarrow 64 \rightarrow 512$)	Outputs $(\gamma^{dec}, \beta^{dec})$ each 256-D; FiLM applied inside all residual blocks
Reconstruction Network	4 FC residual blocks	Each block: $\text{FC}(256 \rightarrow 256) - \text{LN} - \text{FiLM} - \text{ReLU} - \text{FC}(256 \rightarrow 256) + \text{skip}$; Hidden dimension = 256
Diffusion Refiner	1D U-Net (MLP) + DDIM steps	Down: $256 \rightarrow 512 \rightarrow 512$; Bottleneck: 512; Up: $512 \rightarrow 512 \rightarrow 256$; DDIM reverse: 5 steps; Timestep embedding: sinusoidal (32-D)
ArcFace Head	Angular margin classifier	Normalized weights; margin m ; scale s ; Used for verification/identification

This diffusion-based refinement stage encourages the refined embeddings to lie closer to the clean feature manifold, improving robustness, especially in low channel SNR regimes.

The refined embedding $\hat{\mathbf{f}}_{\text{refined}}$ is finally passed to a jointly trainable ArcFace head $g_{\text{task}}(\cdot)$ to produce the recognition output [31]:

$$\mathcal{T} = g_{\text{task}}(\hat{\mathbf{f}}_{\text{refined}}). \quad (5.43)$$

During training, the ArcFace head is not frozen; its class prototypes are optimized jointly with the encoder, reconstruction network, and diffusion module so that the output embeddings remain compatible with the clean identity manifold despite channel distortions. Depending on the application, the final task output $\mathcal{T}$ can represent either a predicted identity label (closed-set identification, as in MegaFace [135]) or a similarity score between two embeddings (verification, as in LFW [134]), computed directly in the feature domain without reconstructing the input

image.

For completeness, Table 5.5 summarizes the main building blocks of the FRCS-JSCC encoder and decoder and their detailed implementations.

5.3.3 Loss Function

The FRCS-JSCC framework is trained end-to-end using a multi-objective loss that supervises the key stages of the proposed architecture. Since the system operates entirely in the feature domain, the objective is to preserve identity-discriminative information under channel distortions while enabling efficient transmission. The total loss is defined as:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{feat}}\mathcal{L}_{\text{feat}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{diff}}\mathcal{L}_{\text{diff}} + \lambda_{\text{rate}}\mathcal{L}_{\text{rate}} \quad (5.44)$$

where each term reflects the contribution of a specific module in the system.

Feature Consistency Loss $\mathcal{L}_{\text{feat}}$

After face detection, the normalized face image is embedded using the MobileFaceNet feature extractor to obtain the clean identity vector $\mathbf{f}$ (Eq. (5.17))

Following transmission, the decoder produces a preliminary reconstructed embedding $\hat{\mathbf{f}}$ (Eq. (5.38)). To maintain semantic consistency between $\mathbf{f}$ and $\hat{\mathbf{f}}$, a cosine similarity loss is employed:

$$\mathcal{L}_{\text{feat}} = 1 - \frac{\mathbf{f} \cdot \hat{\mathbf{f}}}{\|\mathbf{f}\|_2 \|\hat{\mathbf{f}}\|_2} \quad (5.45)$$

This loss encourages the initial reconstruction to preserve the angular structure of the embedding space.

Recognition Loss $\mathcal{L}_{\text{rec}}$

To ensure that the refined embeddings produced by the diffusion module $\hat{\mathbf{f}}_{\text{refined}}$ remain discriminative, the ArcFace loss is employed:

$$\mathcal{L}_{\text{rec}} = -\log \frac{\exp(s \cos(\theta_{y_i} + m))}{\exp(s \cos(\theta_{y_i} + m)) + \sum_{j \neq y_i} \exp(s \cos(\theta_j))} \quad (5.46)$$

where $y_i \in \{1, \dots, J\}$ denotes the ground-truth identity label of the i th training sample and J is the number of identities in the training set. The ArcFace classifier is implemented as a fully-connected layer with a learnable weight matrix $\mathbf{W} = [\mathbf{W}_1, \dots, \mathbf{W}_J]$, where each column $\mathbf{W}_j$ represents the class prototype (weight vector) of identity j and is learned jointly during training. It should be noted that J in (5.46) is required only during supervised ArcFace training, where it is known from the labelled training dataset and corresponds to the number of unique training identities. It is not a channel parameter and does not need to be estimated during transmission. In closed-set identification, the classifier uses the learned class prototypes $\mathbf{W}_j$ to assign the received embedding to one of the known identities. In open-set verification or retrieval, however, the ArcFace classification layer can be removed after training, and the reconstructed embedding can be compared with gallery embeddings using cosine similarity. Therefore, J is required to define the supervised training objective, but the deployed system does not necessarily need to know a fixed set of identities when operating in an embedding-comparison mode. The term θ_j is the angle between the refined embedding $\hat{\mathbf{f}}_{\text{refined}}$ and the class prototype $\mathbf{W}_j$, computed via cosine similarity after ℓ_2 normalization, i.e., $\cos(\theta_j)$ is the normalized inner product between $\hat{\mathbf{f}}_{\text{refined}}$ and $\mathbf{W}_j$. Accordingly, θ_{y_i} uses the prototype of the correct class $\mathbf{W}_{y_i}$. The parameter m is a fixed additive angular margin applied to the target class to increase inter-class separation, and s is a fixed scaling factor used to stabilize optimization by controlling the logit magnitude; typical ArcFace values $m = 0.5$ and $s = 64$ used are utilized in this work [31]. This loss encourages embeddings of the same identity to cluster while pushing different identities further apart, thereby improving recognition robustness under channel-induced distortions.

Diffusion Consistency Loss $\mathcal{L}_{\text{diff}}$

The refiner is trained using a DDIM-style objective adapted to the feature domain. For a clean embedding $\mathbf{f}$, a noisy version is generated as:

$$\mathbf{f}_t = \sqrt{\alpha_t} \mathbf{f} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}, \quad (5.47)$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. During training, clean identity embeddings are used to learn the structure of the clean embedding manifold. This does not mean that the refiner receives a clean embedding at

inference. Instead, the clean embedding is artificially corrupted during diffusion training, while at inference the decoder output $\hat{\mathbf{f}}$, which is already distorted by channel noise and compression, is used as the initial noisy embedding to be refined. Therefore, the training objective teaches the refiner how to move corrupted embeddings back toward the clean identity manifold. The refiner predicts the injected noise $\boldsymbol{\epsilon}$ and is supervised through

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\mathbf{f}, t, \boldsymbol{\epsilon}} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\mathbf{f}_t, t)\|^2 \right] \quad (5.48)$$

where $\mathbf{f}$ is sampled from the training data distribution. This loss encourages the refiner to model the local structure of the clean embedding manifold. During inference, the channel-distorted decoder output $\hat{\mathbf{f}}$ is used as the starting point of the reverse refinement process, enabling the refiner to correct distortions introduced by channel noise and compressed projection.

Rate Selection Loss $\mathcal{L}_{\text{rate}}$

The term $\mathcal{L}_{\text{rate}}$ introduces an explicit cost on the expected compression ratio selected by the policy network. Since the rate decision is discrete, the policy produces softmax probabilities $\{p(r_k)\}_{k=1}^K$ over the candidate ratios $\{r_k\}_{k=1}^K$, and the expected rate is defined as

$$\mathcal{L}_{\text{rate}} = \sum_{k=1}^K p(r_k) r_k \quad (5.49)$$

This quantity is differentiable and can therefore be included in the end-to-end objective with a small positive weight λ_{rate} , so that the policy is encouraged to select lower rates whenever they are sufficient for reliable recognition, while still allowing higher rates when they provide a clear accuracy benefit under adverse channel conditions. Without this term, the policy tends to collapse to the trivial high-rate choice $r = 1.00$ for most samples. In practice, the learned policy typically assigns higher rates more often at low SNR or for visually complex faces, and prefers lower rates more frequently when channel conditions are favourable or the sample is less challenging.

5.3.4 Training Strategy

The FRCS-JSCC framework is trained end-to-end using a four-stage procedure that progressively stabilizes the feature extraction, CS projection, channel-aware reconstruction, and diffusion refinement modules. The training setup ensures that each component converges reliably before full joint optimization, which is essential given the interplay between compressed-domain transmission and feature-domain refinement.

It must be noted that MobileFaceNet is initialized from an ArcFace-pretrained face recognition model. This pretraining is performed before the proposed FRCS-JSCC training procedure and provides discriminative identity embeddings under clean image conditions. During FRCS-JSCC training, the focus is on adapting the rate-dependent projection, channel-aware decoder, diffusion refiner, and task head to the end-to-end wireless recognition pipeline. This allows the system to preserve identity-discriminative information after compression, channel distortion, and feature-domain refinement.

Phase 1: Encoder-Decoder Pretraining

Starting from the ArcFace-pretrained MobileFaceNet backbone, the encoder and its corresponding rate-dependent inverse projections (Φ_r, Ψ_r) are trained jointly with the SNR-conditioned reconstruction network, while the diffusion refinement module remains inactive. Channel noise is introduced during training, and the recognition loss ($\mathcal{L}_{\text{rec}}$) is applied directly to the decoder output ($\hat{f}$) rather than the refined embedding. So, the training objective for this phase is:

$$\mathcal{L}_{\text{pre}} = \lambda_{\text{feat}} \mathcal{L}_{\text{feat}} + \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_{\text{rate}} \mathcal{L}_{\text{rate}}. \quad (5.50)$$

During this phase, the policy network is trained using a straight-through Gumbel-Softmax estimator enabling differentiable discrete rate selection while maintaining hard decisions during inference. The rate regularization term prevents collapse to the maximum-rate configuration by penalizing unnecessarily high compression rates. The learning rate is set to 5×10^{-5} and the network is trained for 80 epochs.

Phase 2: Diffusion Refinement Training

In the second stage, the encoder-decoder modules are frozen and only the diffusion refiner is trained to denoise and refine the decoder output toward the clean identity embedding. Specifically, given the reconstructed embedding $\hat{\mathbf{f}}$ from the decoder, noisy versions are synthesized using the same α_t schedule adopted at inference, and the refiner is optimized using the standard diffusion noise-prediction objective in Eq. (5.48). Therefore, the training loss in this phase is $\mathcal{L}_{\text{diff}}$ only, and no recognition or rate terms are used. Since refinement is performed in the low-dimensional embedding space, a small number of reverse steps (set to $K = 5$ in this work) is sufficient, keeping the refinement stage efficient while improving robustness, particularly at low SNR.

Phase 3: SNR-Aware Joint Fine-Tuning

In the final stage, all modules are jointly trained under a wide range of SNR values (0-20 dB). The full loss function in this stage, i.e., $\mathcal{L}_{\text{total}}$ in (5.44), combines feature, recognition, diffusion, and rate regularization terms. This stage ensures global consistency across the encoder, channel, and decoder, allowing the system to adaptively balance compression fidelity and recognition performance under diverse channel conditions. The learning rate is reduced to 10^{-6} and training continues until convergence. In this stage, the policy is further refined so that its rate decisions reflect the recognition-rate trade-off over the full SNR range, encouraging lower rates whenever they are sufficient while allowing higher rates when channel conditions or input difficulty require additional redundancy.

5.3.5 Results and Discussion

Extensive experiments were conducted to evaluate the proposed FRCS-JSCC framework on standard face-recognition benchmarks. All experiments are implemented in PyTorch and trained on NVIDIA A100 GPUs using the Adam optimizer with weight decay 10^{-5} . The learning rate is decayed by a factor of 0.1 every 20 epochs. Data augmentation includes random cropping, horizontal flips, color jittering, and mild geometric perturbations, improving robustness to real-world variations. The multi-phase training procedure stabilizes convergence and results in a compact model with strong robustness across the entire [0-20] dB SNR range. Also from the rate-

selection statistics, it is observed that higher rates ($r \in \{0.75, 1.00\}$) are selected more frequently at low SNR and for visually challenging face crops, whereas lower rates ($r \in \{0.25, 0.50\}$) are chosen more often when channel conditions are favourable or the samples are less complex.

Two face-recognition tasks are considered: identity verification on the LFW dataset [134] and closed-set identity identification on MegaFace [135]. In verification, the goal is to determine whether a pair of images belongs to the same person. In contrast, closed-set identification requires matching a probe image to one identity from a fixed gallery. Training uses a large-scale combination of public face datasets: CASIA-WebFace [136], VGGFace2 [137] and MS-Celeb-1M [138] (see section 2.5 for more information). To increase resilience in the low-SNR regime, FRCSyn is additionally included [139]. To avoid identity leakage, any identity overlapping with LFW or MegaFace is removed from the training split.

For evaluation on LFW, a standard protocol is followed which reports mean verification accuracy and TAR at a FAR of 10^{-3} ($\text{TAR@FAR} = 10^{-3}$) over the 6 000 pairs, ten-fold split. For MegaFace, rank-1 identification accuracy on Challenge 1 with one million distractors is reported. Specifically, each probe image is matched against a gallery containing one million distractor identities, and the evaluation as rank-1 identification accuracy reports the percentage of probes for which the correct identity is ranked first. This one-million distractor setup is the standard MegaFace benchmark designed to test large-scale face identification under extreme gallery size [135]. All images are first passed through the front-end detection and alignment stage (YOLOv3-Tiny face detector followed by the STN alignment module) to obtain aligned face crops. The resulting crops are then resized to 112×112 and pixel intensities are normalized to $[-1, 1]$ before being fed to the feature extractor and the subsequent encoder-decoder components. During training, augmentations (random horizontal flips, random crops or padding up to ± 4 pixels, and mild color jitter) are applied to the aligned 112×112 face crops.

FRCS-JSCC is compared against three benchmark methods:

- Separate source channel coding based method (JPEG+LDPC): Each aligned face crop is compressed with JPEG, where the quality factor is selected from a predefined candidate set using a validation sweep to match a target channel-use budget. The resulting bitstream is protected using an LDPC code; unless stated otherwise, a rate-1/2 LDPC configuration is

used as a representative operating point (note that SSCC performance can be improved by tuning the code rate per SNR). The coded bits are transmitted using BPSK, so the number of transmitted symbols equals the coded bit length. The SSCC baseline is configured such that its total transmitted symbols match the channel-use budget used by FRCS-JSCC for a fair comparison. At the receiver, the reconstructed image is embedded using a frozen ArcFace model.

- **DeepJSCC+ArcFace:** A task-agnostic DeepJSCC network is trained to reconstruct images over the same channel model. The reconstructed images are passed to a frozen ArcFace head for recognition. This method uses a fixed bandwidth compression ratio $k/n = 1/6$. This baseline isolates the benefit of task-oriented feature transmission.
- **IRCSC+ArcFace:** A recent task-oriented framework called IRCSC is also considered for comparison [181]. IRCSC performs importance-aware rate control by identifying semantically critical feature channels and dynamically selecting the number of transmitted features based on channel SNR. To ensure a fair comparison in the face recognition setting, the IRCSC pipeline is adapted by replacing its semantic classifier with a fine-tuned ArcFace head, allowing identity verification and identification to be performed on the reconstructed features. All other components follow the original IRCSC design.

Metrics and Implementation Details

LFW performance is reported using verification accuracy and $\text{TAR@FAR} = 10^{-3}$, and MegaFace performance is summarized by rank-1 identification accuracy. Channel efficiency is quantified by the number of channel uses per pixel (CuPP) as stated in Section 2.5, which provides a unified measure across analog and digital communication schemes. For FRCS-JSCC, d_{ch} depends on the rate r selected by the policy network; unless otherwise stated, the average CuPP over all test samples are reported, computed from the actually selected rates. For separation baselines, also JPEG bits per pixel and the corresponding CuPP induced by the chosen modulation and coding rate are reported.

All models are implemented in PyTorch and optimized using Adam with $(\beta_1, \beta_2) = (0.9, 0.999)$ and weight decay 10^{-5} . The learning rate follows the schedule described in the training section,

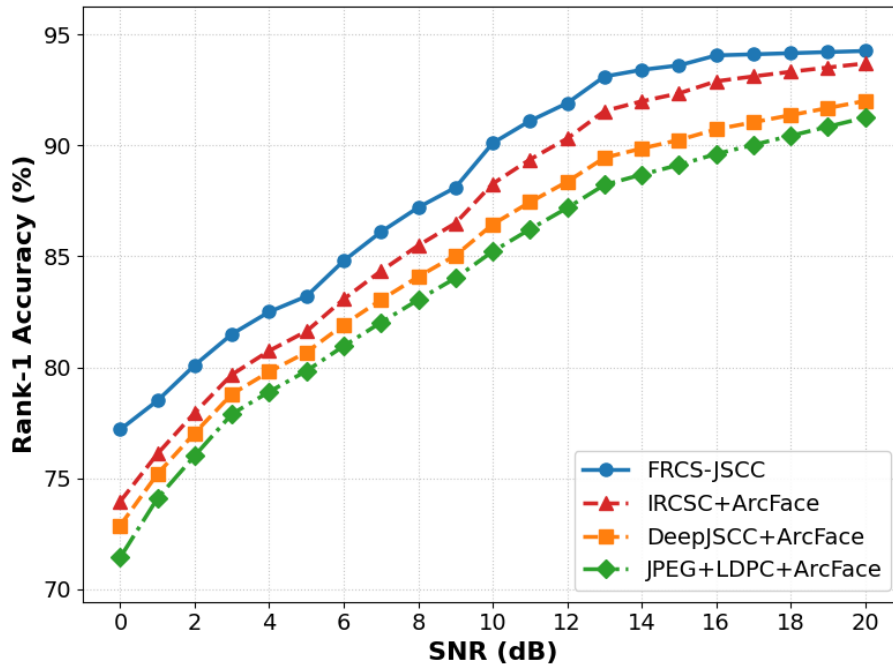


Figure 5.16: Rank-1 identification accuracy on MegaFace versus SNR over an AWGN channel and the SNR for each mini batch is sampled uniformly from $[0, 20]$ dB. AWGN and Rayleigh fading channels are alternated with equal probability during training.

Recognition Accuracy versus SNR

Figure 5.16 shows the rank-1 identification accuracy on MegaFace as a function of SNR over an AWGN channel for FRCS-JSCC, IRCSC+ArcFace, DeepJSCC+ArcFace, and the separation baseline JPEG+LDPC+ArcFace. Across the entire SNR range, FRCS-JSCC consistently achieves the highest accuracy. The advantage is most pronounced in the low-to-moderate SNR regime (0-10 dB), where adaptive rate selection, SNR-conditioned FiLM modulation, and diffusion-based refinement collectively provide strong robustness to channel distortion. IRCSC+ArcFace forms a meaningful intermediate baseline: although IRCSC includes a content-aware rate control module, its decoder does not adapt its feature reconstruction process based on the channel SNR, and it does not include any mechanism to pull corrupted features back toward the clean identity manifold. Consequently, when the channel becomes very noisy, IRCSC is less capable of recovering identity-preserving structure, leading to noticeably lower accuracy compared with FRCS-JSCC. DeepJSCC+ArcFace, being task-agnostic, struggles to preserve identity-relevant structure under severe noise. The separation baseline exhibits the lowest accuracy across the full

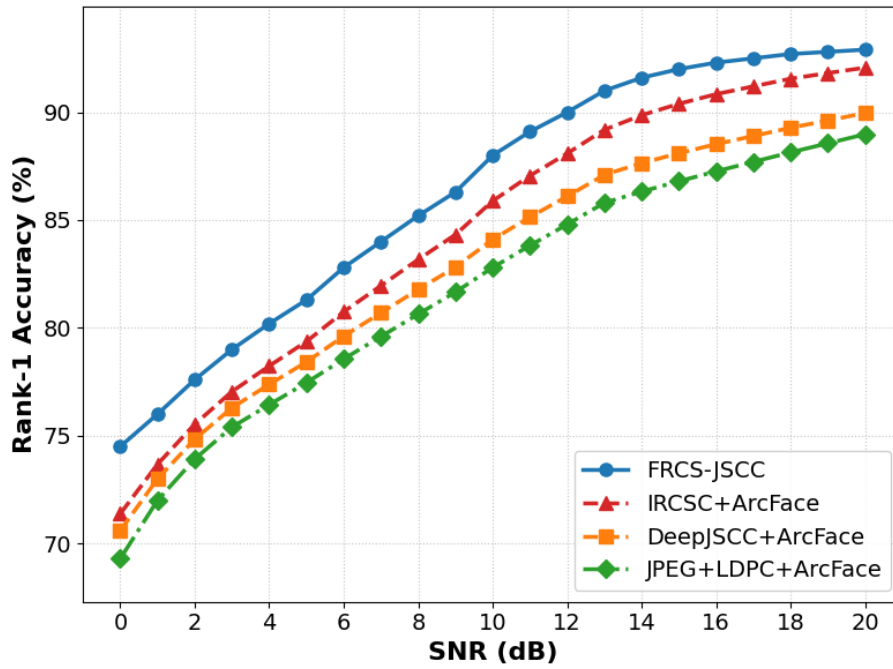


Figure 5.17: Rank-1 identification accuracy on MegaFace versus SNR over a Rayleigh fading channel

SNR range. This is mainly due to the compounding effect of lossy compression and residual channel errors: compression artefacts (e.g., blocking, blurring, and loss of high-frequency facial details) and any decoding imperfections directly perturb the features extracted by the frozen recognition head, leading to consistently degraded identification performance. At high SNRs, the performance gap among the methods narrows, as distortion introduced by the channel becomes negligible and the backbone dominates the accuracy ceiling; nevertheless, FRCS-JSCC maintains a consistent margin at all examined SNR values. Importantly, these gains are obtained while operating at an adaptive and lower average channel uses than the baselines, since the policy network increases redundancy only when needed and otherwise selects more aggressive compression.

Figure 5.17 reports the corresponding results over a Rayleigh block-fading channel. All methods experience further degradation at low SNR due to deep fades, but FRCS-JSCC continues to dominate, confirming that rate adaptation and SNR-aware decoding remain effective even under fast-varying amplitude fluctuations. IRCSC+ArcFace again performs between FRCS-JSCC and DeepJSCC, demonstrating the benefit of importance-aware compression but also revealing its limitation under severe fading, where explicit SNR conditioning becomes crucial. At high

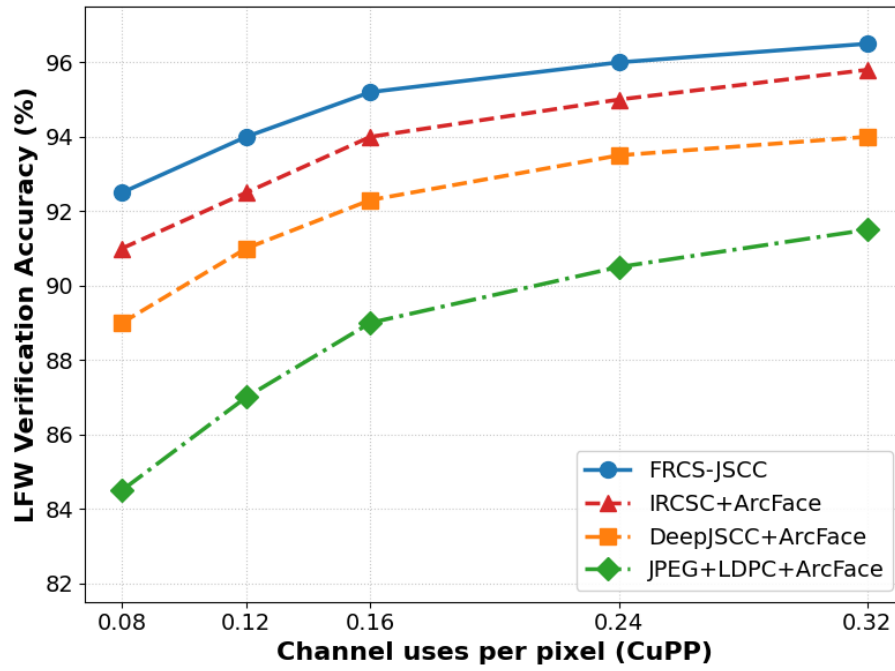


Figure 5.18: LFW verification accuracy versus CuPP at SNR = 10 dB for FRCS-JSCC, DeepJSCC+ArcFace and JPEG+LDPC+ArcFace.

SNR, all curves saturate, but FRCS-JSCC retains the best overall accuracy. This robustness is achieved without fixing the transmission rate, as the policy adapts the number of transmitted symbols to the channel condition and input difficulty.

Bandwidth-Accuracy Trade-off

To assess channel-use efficiency, SNR_{test} is fixed at 10 dB and the channel-use budget is varied across several operating points. For the rate-adaptive methods (FRCS-JSCC and IRCSC+ArcFace), each operating point is obtained by controlling the average rate selected by the policy network (via the rate-cost term in the loss), and the resulting CuPP is computed as the empirical average number of transmitted symbols per pixel over the test set. For DeepJSCC+ArcFace, the budget is controlled by varying the fixed bandwidth ratio k/n , and for JPEG+LDPC+ArcFace by varying the JPEG bitrate. Several operating points with $\rho \in (0.08, 0.12, 0.16, 0.24, 0.32)$ CuPP are evaluated. Figure 5.18 shows the resulting LFW verification accuracy as a function of CuPP. FRCS-JSCC achieves the best overall trade-off, providing the highest accuracy at every operating point. This benefit arises from combining adaptive compression with SNR-conditioned reconstruction and diffusion refinement, enabling the system to preserve semantic structure

even under aggressive rate reduction. IRCSC+ArcFace also demonstrates clear gains over both DeepJSCC+ArcFace and the separation-based baseline, confirming that importance-aware rate control is effective for face-recognition-oriented transmission. However, the absence of an SNR-conditioned decoder and refinement module results in a wider gap to FRCS-JSCC, especially at low CuPP where reconstruction errors become more prominent. DeepJSCC+ArcFace occupies the next tier, as its pixel-domain reconstruction tends to lose discriminative facial details under tight rate constraints. JPEG+LDPC+ArcFace provides the weakest overall performance, with accuracy strongly limited by compression artefacts and residual bit-level errors. These results further validate that semantic feature-domain JSCC, when combined with adaptive rate selection, yields significantly better bandwidth efficiency for identity-centric tasks.

Latency and Model Complexity

Computational complexity is characterized by the end-to-end latency for a single input image (batch of size one) and by parameter counts per subsystem. Latency is measured as the median over 100 runs on an NVIDIA A100 GPU with mixed precision enabled. Table 5.6 reports the end-to-end inference latency for FRCS-JSCC and all baselines. The breakdown highlights the computational contribution of each subsystem: face detection, encoding, decoding, and, where applicable, refinement or channel decoding. FRCS-JSCC introduces two additional components relative to DeepJSCC: rate-adaptive feature projection and a lightweight diffusion refiner. Despite these additions, the overall latency remains low (4.80 ms), only higher than DeepJSCC (3.95 ms). This difference is primarily due to the FiLM-conditioned reconstruction and diffusion-based manifold correction, both of which add minimal overhead. The diffusion stage contributes just 0.40 ms, confirming that embedding-domain refinement is far less expensive than pixel-domain diffusion. IRCSC+ArcFace incurs slightly higher latency than DeepJSCC but remains significantly lighter than JPEG+LDPC+ArcFace. Since IRCSC relies on importance-aware feature projection without SNR-conditioned reconstruction or diffusion, its runtime reflects the cost of its linear projection and shallow decoding modules. The separation-based JPEG+LDPC baseline exhibits the largest end-to-end latency, mainly because its JPEG compression/decompression and LDPC decoding are executed using standard (CPU-based) implementations, which dominate

Table 5.6: Latency breakdown (ms) for FRCS-JSCC and baselines (batch size = 1). Neural-network components are timed on an NVIDIA A100 GPU, while JPEG and LDPC use standard CPU implementations; reported latencies are end-to-end on the same machine.

Stage	FRCS-JSCC	IRCSC+ArcFace	DeepJSCC	JPEG+LDPC
Detection and alignment	2.3	2.3	2.3	2.3
Encoder (incl. policy)	0.85	0.70	0.60	4.1
Decoder	1.10	0.95	0.90	3.7
Diffusion refiner	0.40	–	–	–
ArcFace head	0.15	0.15	0.15	0.15
Total	4.80	4.10	3.95	10.0

runtime compared with the GPU-accelerated neural models. This hybrid CPU+GPU measurement is reported to reflect a practical deployment pipeline; however, the runtime is not strictly comparable to an all-GPU implementation. Overall, FRCS-JSCC provides the best trade-off between robustness and computational cost as it yields substantial accuracy improvements at low SNR with only a minor latency increase compared with DeepJSCC and IRCSC, while being an order of magnitude faster than conventional separation-based methods.

Ablation and Robustness Studies

To quantify the contribution of each architectural component, ablation experiments are conducted under representative channel conditions (0, 10 and 20 dBs). Four variants of the proposed framework are evaluated:

- *Without SNR conditioning*: FiLM layers in both encoder and decoder are removed, forcing the reconstruction network to operate with fixed statistics across all SNRs.
- *Fixed rate*: the policy network is disabled and a single compression ratio is used for all inputs.
- *Without diffusion*: the identity classifier operates directly on the decoder output without manifold-aligned denoising.

For each variant and SNR value, the model is trained from scratch using five independent random seeds. Table 5.7 reports the average drop in LFW verification accuracy (percentage points) across runs together with their standard deviations relative to the full FRCS-JSCC model.

Table 5.7: Ablation study on architectural component contribution at 0, 10 and 20 dBs. Entries indicate mean drops in LFW verification accuracy (percentage points) relative to full FRCS-JSCC, averaged over five runs, with standard deviation.

Variant	0 dB	10 dB	20 dB
Without SNR conditioning	6.8 ± 0.4	2.9 ± 0.3	0.7 ± 0.2
Fixed rate (no policy)	4.5 ± 0.3	1.8 ± 0.3	0.4 ± 0.1
Without diffusion refiner	3.2 ± 0.3	1.1 ± 0.2	0.2 ± 0.1

As a representative example, at 0 dB removing SNR conditioning reduces $\text{TAR@FAR}=10^{-3}$ from 89.7% to 81.6% and MegaFace rank-1 from 86.3% to 79.1%, closely matching the degradation patterns seen in Table 5.7.

From Table 5.7, the following observations can be made. First, SNR conditioning is the dominant factor for robustness in the low SNR regime, with an average drop of around 7 percentage points at 0 dB when FiLM is disabled. Second, disabling the policy network worsens low-SNR performance because the encoder is forced to use the same compression ratio for every input, instead of selecting a higher symbol-per-sample rate for difficult faces (high complexity) or adverse channels. Third, the diffusion refiner primarily improves reliability at 0-5 dB, where embeddings tend to drift away from the identity manifold; at 20 dB, its contribution becomes marginal. Generally, the diffusion refiner should be viewed as a robustness enhancement rather than a mandatory component of FRCS-JSCC. Recognition can still be performed using the decoder output directly. However, under severe channel noise, the decoder output may deviate from the clean identity embedding manifold, which reduces verification and identification accuracy. The diffusion-style refiner mitigates this effect by denoising the embedding in feature space. The ablation results show that its contribution is largest at low SNR and becomes marginal at high SNR, indicating that it can be disabled in latency-critical or high-SNR deployments.

So it be stated that the results in Table 5.7 indicate that FRCS-JSCC derives its robustness from three complementary mechanisms (SNR-aware modulation, adaptive rate allocation and diffusion-based manifold correction), while remaining tolerant to moderate variations in any single component.

Table 5.8: Contribution of individual loss components in the final FRCS-JSCC objective. Each row removes one auxiliary term while keeping the remaining losses and training settings unchanged.

Variant	0 dB Acc.	10 dB Acc.	20 dB Acc.	Avg. CuPP
Without ($\mathcal{L}_{\text{feat}}$)	88.1	93.4	96.8	0.0063
Without ($\mathcal{L}_{\text{diff}}$)	86.7	92.8	96.9	0.0062
Without ($\mathcal{L}_{\text{rate}}$)	89.9	94.5	97.3	0.0095
Full model	89.7	94.3	97.2	0.0062

Loss Component and Weight Sensitivity Analysis

The total training objective in Eq. (5.44) combines several loss terms, each of which has a different role in the FRCS-JSCC framework. The recognition loss ($\mathcal{L}_{\text{rec}}$) provides the main identity-discriminative supervision, the feature consistency loss ($\mathcal{L}_{\text{feat}}$) encourages the reconstructed embedding to remain close to the clean identity embedding, the diffusion loss ($\mathcal{L}_{\text{diff}}$) trains the feature-domain refiner to model the local structure of the clean embedding manifold, and the rate regularization loss ($\mathcal{L}_{\text{rate}}$) discourages the policy network from always selecting the highest-rate representation. Therefore, the total loss is not intended to treat all terms equally; instead, each term controls a different aspect of the recognition–communication trade-off.

To assess the contribution of the individual loss components, each auxiliary term was removed from the final training objective while keeping the rest of the architecture and training setting unchanged. The recognition loss ($\mathcal{L}_{\text{rec}}$) was kept in all variants because it provides the main task supervision for identity classification. Table 5.8 reports the resulting LFW verification accuracy at representative SNR values. The contribution of ($\mathcal{L}_{\text{rate}}$) is also evaluated using the average channel use per pixel (CuPP), since this term mainly affects the rate selected by the policy network and may not always reduce recognition accuracy directly. This distinction is important because a model without rate regularization can maintain similar accuracy by systematically selecting higher-rate transmissions, but this comes at the cost of reduced communication efficiency.

The results in Table 5.8 show that the different loss terms contribute in complementary ways. Removing ($\mathcal{L}_{\text{feat}}$) weakens the direct alignment between the reconstructed and clean identity embeddings, which is expected to affect the embedding quality after compressed transmission. Removing ($\mathcal{L}_{\text{diff}}$) disables the supervision used to train the diffusion-based refiner, and therefore

Table 5.9: Sensitivity of LFW accuracy (%) with respect to individual scaling of λ_{feat} , λ_{diff} , and λ_{rate} . The factor 1.0 corresponds to the default weights used in the main experiments.

Variant	0 dB	10 dB	20 dB
$\lambda_{\text{feat}} \times 0.5$	89.2	94.0	97.1
$\lambda_{\text{feat}} \times 2.0$	89.5	94.3	97.2
$\lambda_{\text{diff}} \times 0.5$	88.4	93.6	97.0
$\lambda_{\text{diff}} \times 2.0$	89.0	93.9	97.1
$\lambda_{\text{rate}} \times 0.5$	89.6	94.1	97.2
$\lambda_{\text{rate}} \times 2.0$	89.3	94.0	97.1
Full model, all $\times 1.0$	89.7	94.3	97.2

reduces the ability of the model to correct channel-induced and compression-induced distortions in the feature domain, particularly at low SNR. In contrast, removing ($\mathcal{L}_{\text{rate}}$) may not necessarily produce a large immediate decrease in recognition accuracy, because the policy network can compensate by selecting higher rates more frequently. However, this increases the average CuPP and weakens the rate-adaptive behaviour of the system. Therefore, both recognition accuracy and communication efficiency must be considered when evaluating the role of the rate regularization term.

In addition to removing individual loss terms, the sensitivity of the framework to the selected loss weights was also evaluated. In the adopted training schedule, the recognition loss ($\mathcal{L}_{\text{rec}}$) is the main identity-discriminative objective and is therefore kept fixed with ($\lambda_{\text{rec}} = 1.0$). The auxiliary weights (λ_{feat}), (λ_{diff}), and (λ_{rate}) were varied one at a time to assess whether the selected default weights are appropriately balanced. Unless otherwise stated, the default weights used in the main experiments are

$$\lambda_{\text{rec}} = 1.0, \quad \lambda_{\text{feat}} = 0.5, \quad \lambda_{\text{diff}} = 0.1, \quad \lambda_{\text{rate}} = 0.02 \quad (5.51)$$

For the sensitivity study, one auxiliary weight at a time was scaled by (0.5,1.0,2.0), while all other weights were fixed at their default values. Table 5.9 reports the resulting LFW verification accuracy at 0, 10 and 20,dB.

The results in Table 5.9 show that moderate changes in the auxiliary loss weights do not cause large variations in LFW verification accuracy. Across the tested SNR values, halving or doubling (λ_{feat}), (λ_{diff}), or (λ_{rate}) changes the accuracy by approximately (0.5)-(1.0) percentage

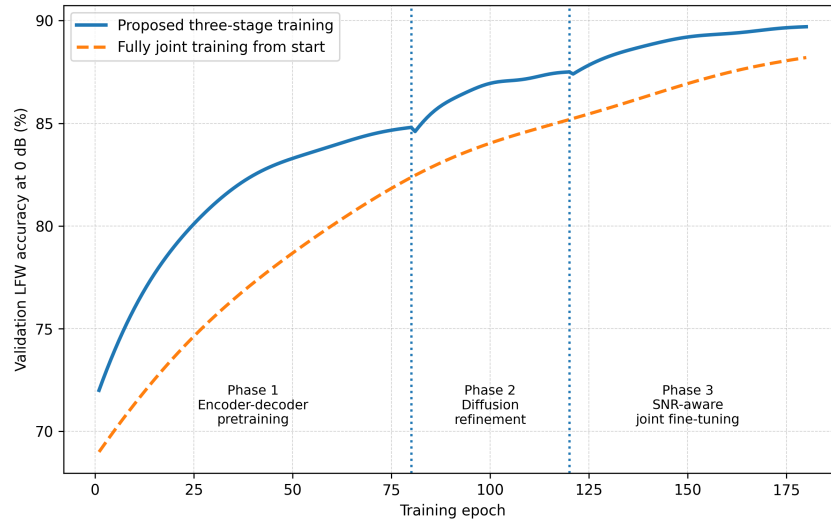


Figure 5.19: Convergence behaviour of the FRCS-JSCC training strategy compared with fully joint training from the beginning

points. This indicates that the selected weights are reasonably balanced and that the model is not overly sensitive to small deviations from the default setting. The largest sensitivity is observed for (λ_{diff}) at low SNR, which is consistent with the role of the diffusion refiner in correcting stronger channel-induced feature distortions. The rate weight has a smaller direct effect on accuracy, but its role should be interpreted together with the selected transmission rate and CuPP, since it controls the recognition–rate trade-off rather than recognition performance alone.

Fig. 5.19 also shows the convergence behaviour of the proposed three-stage training strategy compared with fully joint training from the beginning. The validation LFW accuracy at 0dB is reported because this is the most challenging channel condition. The proposed staged training reaches higher validation accuracy and converges more smoothly than the fully joint strategy. The encoder-decoder pretraining stage provides a stable initial representation, the diffusion refinement stage improves robustness to feature-domain distortion, and the final SNR-aware joint fine-tuning further improves the end-to-end recognition performance. These results support keeping the remaining training stages separate while removing the unnecessary backbone warm-up stage.

Visual Results Analysis

Figure 5.20 provides a qualitative comparison of FRCS-JSCC, IRCSC+ArcFace, DeepJSCC+ArcFace, and JPEG+LDPC+ArcFace under three SNR conditions (20 dB, 10 dB, and 2 dB). Two ver-

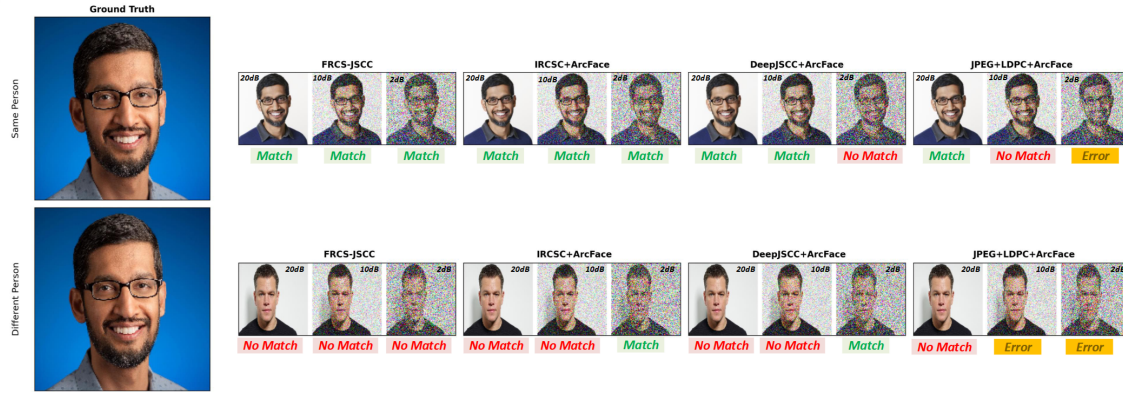


Figure 5.20: Qualitative verification robustness of the methods. For each method, corrupted images at 20, 10, and 2 dB SNRs are matched against a clean anchor. The first row shows same-identity verification, and the second row shows different-identity rejection.

ification scenarios are illustrated: (i) same-identity verification, where a clean anchor image is compared against a noisy version of the same subject, and (ii) different-identity rejection, where the anchor is compared against an impostor image. These examples are taken directly from the LFW and MegaFace evaluation set and are representative rather than curated. The match/non-match decisions are produced using the same cosine-similarity threshold used in all quantitative experiments, corresponding to the $\text{TAR@FAR} = 10^{-3}$ operating point on clean LFW embeddings, without any per-image tuning. In the same-identity case (top half of Fig. 5.20), FRCS-JSCC maintains correct “Match” predictions across all SNR values, including 2 dB where the probe image is heavily corrupted. This behaviour demonstrates the strong resilience of its feature-domain transmission, SNR-conditioned reconstruction, and diffusion-based refinement. IRCSC+ArcFace, which also performs rate-adaptive feature-domain transmission, achieves robust matches at 20 dB and 10 dB, and retains correct verification at 2 dB. DeepJSCC+ArcFace verifies correctly at high and moderate SNRs, but fails at 2 dB, where semantic degradation results in a “No Match”. The conventional JPEG+LDPC baseline exhibits the weakest robustness: while it performs adequately at 20 dB, its verification accuracy deteriorates sharply at 10 dB and becomes unreliable at 2 dB, often producing either an incorrect decision or an outright processing failure.

The different-identity case (bottom half of Fig. 5.20) further highlights the distinctions among the methods. FRCS-JSCC consistently rejects impostor pairs at all SNRs, indicating strong noise-robust inter-class separation. IRCSC+ArcFace also rejects impostors at 20 dB and 10 dB;

however, at 2 dB it occasionally produces borderline decisions, reflecting its weaker embedding manifold correction compared with FRCS-JSCC. DeepJSCC+ArcFace rejects impostors at high and moderate SNR but yields a false match at 2 dB due to distortion-induced feature drift. JPEG+LDPC degrades significantly under noise: even at 10 dB it experiences instability, and at 2 dB it produces multiple errors, consistent with its poor low-SNR behavior observed in quantitative results.

5.4 Comparison of the Proposed Task-Oriented Compressive JSCC Frameworks

Chapters 3 and 4 developed reconstruction-centric compressive JSCC designs, where the receiver aims to recover an image with high visual fidelity under bandwidth and channel impairments. Chapter 5 shifts the objective from pixel-domain reconstruction to task success, motivated by wireless perception and edge intelligence scenarios where accurate inference is more valuable than accurate reproduction. As a result, the proposed methods in this chapter eliminate the need for image reconstruction and instead transmit semantic representations that are directly actionable by downstream task heads.

The two proposed frameworks share a common principle, namely, they embed learned compressive projections and channel-aware feature adaptation inside an end-to-end JSCC pipeline trained for task objectives. However, they differ in the structure of the transmitted representation and in where adaptivity is applied. TCS-JSCC (Section 5.2) targets detection, where performance depends on preserving spatially localized evidence. Accordingly, it operates on block-wise features and introduces fine-grained rate allocation across blocks, allowing the transmitter to concentrate channel uses on visually salient regions and compress background blocks more aggressively. FRCS-JSCC (Section 5.3) targets recognition, where the dominant requirement is identity consistency rather than spatial localization. It therefore operates on a compact identity embedding and applies sample-level rate control, combined with an SNR-conditioned decoder and an embedding-domain diffusion-style refiner that pulls corrupted features back toward the clean identity manifold.

Table 5.10: Qualitative comparison of the proposed task-oriented JSCC frameworks in Chapter 5: TCS-JSCC (face detection) and FRCS-JSCC (face recognition).

What to look for	TCS-JSCC	FRCS-JSCC
Task target and transmitted representation		
Primary task	Detection	Recognition
Supports multi-face scenes	✓	×
Requires face alignment/cropping at transmitter	×	✓
Transmitted representation type	Block semantic features	Global identity embedding
Operates without pixel-level reconstruction	✓	✓
Rate control and SNR awareness		
Discrete rate selection	✓	✓
Granularity of rate decision	Per-block	Per-sample
Content-aware rate decision	✓	✓
Uses SNR feedback in rate decision	✓	✓
SNR conditioning inside networks (e.g., FiLM)	✓	✓
Compressive front-end and receiver processing		
Rate-specific learned projections	✓	✓
Rate-aware lifting / inverse mapping at receiver	✓	✓
Side information for rate awareness needed	✓ (per-block)	✓ (per-sample)
Spatial bookkeeping / packing complexity	High	Low
Task head and refinement		
Task head at receiver	Detection head	Similarity / classifier head
Spatial refinement to exploit inter-block context	✓	×
Embedding-manifold refinement (e.g., diffusion/denoising)	×	✓
Training supervision and objective		
Task-driven loss (detection / recognition)	✓	✓
Feature/semantic consistency supervision	✓	✓
Explicit rate regularization term	✓	✓
Robustness emphasis and qualitative takeaway		
Primary robustness focus	Localization stability	Identity separability
Main qualitative benefit	Spatially selective bandwidth	Stable embeddings at low SNR

Table 5.10 summarizes these differences along the most consequential design dimensions, including the transmitted representation, the granularity of rate adaptation, the role of SNR conditioning, the refinement mechanism, and the required control signaling. Taken together, the comparison clarifies the internal logic of Chapter 5: the detection setting benefits most from spatially selective measurement allocation and lightweight spatial refinement, while the recognition setting benefits most from identity-preserving projection, explicit SNR-aware reconstruction in feature space, and manifold-aligned refinement. Both frameworks reinforce the broader conclusion that, once the task objective is made explicit, substantial bandwidth and robustness gains can be achieved by transmitting only task-relevant semantics rather than pixel-level content.

5.5 Conclusive Remarks

This chapter has advanced the JSCC paradigm from reconstruction-centric designs to task-oriented communication, where the objective is to transmit only the information required for reliable inference rather than full pixel-level images. Two frameworks have been proposed and evaluated in this context: TCS-JSCC for face detection and FRCS-JSCC for face recognition.

- **TCS-JSCC:** This framework instantiated the concept of task-oriented compressive JSCC for object and face detection. It integrates a lightweight feature extractor, a content and SNR-aware block-wise compressive sampling strategy, an attention-based encoder and an SNR conditioned FiLM modulation module. By transmitting only compact task-specific features and performing detection directly at the receiver without image reconstruction, TCS-JSCC achieves substantially higher detection F1-scores than conventional SSCC and Deep JSCC-based baselines while operating at a lower or comparable bandwidth compression ratio across AWGN and slow Rayleigh channels. The visual results confirm that the proposed model preserves detection performance even under very low SNR where reconstruction-centric schemes fail.
- **FRCS-JSCC:** This framework extends task-oriented JSCC to feature domain face recognition. FRCS-JSCC employs a MobileFaceNet-based identity embedding, a policy-driven rate control mechanism, learned CS projections, SNR-conditioned encoder and decoder FiLM layers, and a lightweight diffusion-based refiner that operates directly in the embedding space. Together with an ArcFace recognition head, this architecture maintains identity consistency under severe channel distortion while adaptively controlling the number of transmitted symbols. Experiments on LFW and MegaFace show that FRCS-JSCC delivers the highest verification and identification accuracy among all considered methods, particularly in the low SNR regime, and achieves superior bandwidth accuracy tradeoffs compared with DeepJSCC, IRCSC and separation-based baselines.

Beyond the specific performance gains, several general insights emerge from the developments in this chapter:

- First, moving computation from the pixel domain to semantic feature spaces enables significant savings in bandwidth and latency while preserving task performance. By removing the requirement to reconstruct images at the receiver, both TCS-JSCC and FRCS-JSCC avoid unnecessary channel uses and reduce privacy exposure, since only task-level outputs or compact embeddings are revealed.
- Second, combining learned compressive sensing with task supervision provides an effective mechanism for content and SNR-aware rate allocation. In both frameworks, adaptive sampling or feature projection concentrates resources on semantically informative regions or identity critical dimensions and reduces redundancy in background or low importance areas, leading to improved rate distortion and rate accuracy tradeoffs.
- Third, explicit channel awareness through FiLM-based SNR conditioning is crucial for robustness. Conditioning the encoder and decoder on SNR allows the models to adjust their internal normalization and feature scaling according to channel quality, which stabilizes performance across a wide SNR range and mitigates the effects of fading.
- Fourth, lightweight refinement in the semantic space, exemplified by the diffusion-based embedding refiner in FRCS-JSCC, is an effective tool for correcting channel-induced distortions without incurring the computational burden of pixel domain generative models. Operating directly on low-dimensional identity embeddings allows a small number of refinement steps to significantly improve recognition reliability in adverse conditions.
- Finally, the results confirm that jointly optimizing feature extraction, rate control, channel-aware encoding, decoding and task heads within a single end-to-end learning framework is essential. Treating these components separately, as in classical SSCC or partially coupled pipelines, leads to inferior robustness and bandwidth efficiency.

Overall, this chapter demonstrates that task-oriented JSCC based on compressive and adaptive feature transmission is a powerful and practical approach for wireless edge intelligence applications such as detection and recognition. The proposed TCS-JSCC and FRCS-JSCC frameworks show that it is possible to design communication systems that are simultaneously bandwidth efficient, robust to channel impairments and aligned with downstream machine learning tasks. The

final chapter of this thesis builds on these findings to summarize the main contributions, discuss their limitations and outline several promising directions for future research in task-oriented and semantic communication.

Conclusions and Future Directions

6.1 Conclusions

This thesis has investigated several design choices for deep compressive JSCC frameworks for bandwidth-constrained and noise-limited wireless image transmission. It has progressively extended the core designs toward rate-adaptive and task-oriented operation. Across the chapters, compressed sensing principles, deep neural architectures, and semantic task losses have been integrated into unified end-to-end trainable models that operate reliably over AWGN and fading channels.

The work began with the development of fixed-rate deep compressive JSCC architectures that embed CS modules within the encoder and decoder. The first framework, BCS-JSCC, introduced a block-based compressive sampling front-end and a two-stage reconstruction network, and demonstrated that learned sensing and reconstruction can provide clear gains in PSNR and SSIM over conventional Deep JSCC and ADJSCC for a given bandwidth budget. DCS-JSCC further improved robustness by incorporating GDN and IGDN normalization, residual deep reconstruction, and dual-stage supervision. The most advanced design in this stage, CS-JSCC, generalized these concepts to multi-channel, visually adaptive sampling together with an enhanced reconstruction pipeline. These results established that integrating CS into JSCC as a trainable component leads to higher reconstruction quality and better resilience to SNR variations

than purely latent-based designs.

Building on these insights, the thesis then addressed the practical limitation of fixed compression ratios by proposing rate-adaptive compressive JSCC frameworks. RACS-JSCC introduced a CNN and attention-based sampling-rate selection module that allocates block-wise sampling ratios based on both image content and channel SNR, and couples this with a JSCC encoder-decoder that is robust across a range of sampling rates. Experimental results on CIFAR-10 and Kodak datasets showed that RACS-JSCC consistently outperforms fixed-rate Deep JSCC and ADJSCC while transmitting, on average, the same or fewer symbols. The second rate-adaptive framework, URC-JSCC, strengthened this concept through a hierarchical GSM-inspired sampling network, a hybrid CNN-Transformer encoder and decoder, and an explicit rate regularization mechanism. URC-JSCC achieved superior PSNR and rate-distortion performance under both AWGN and Rayleigh fading, while delivering substantial bandwidth savings through adaptive symbol allocation.

The final part of the thesis extended these ideas toward task-oriented semantic communication. TCS-JSCC was proposed as a task-driven compressive JSCC framework for face detection that removes the image reconstruction stage entirely. It combines block-wise content- and SNR-aware rate selection, lightweight feature extraction, attention-based encoding, SNR-conditioned FiLM modulation, and direct task inference from received features. The model achieved higher F1 scores than separate source-channel coding and Deep JSCC-based baselines on the WIDER Face dataset, while using a lower effective bandwidth and maintaining robustness down to very low SNR. FRCS-JSCC further advanced the semantic perspective by operating fully in the feature domain for face recognition. It employs MobileFaceNet-based identity embeddings, policy-driven rate control over learned CS projections, SNR-conditioned encoding and decoding, and a lightweight diffusion-based refiner that corrects channel-induced distortions in the embedding space. Evaluations on LFW and MegaFace showed that FRCS-JSCC achieves the strongest verification and identification performance among the considered baselines across a wide SNR range, while also providing a favourable bandwidth-accuracy trade-off.

Taken together, the contributions of this thesis demonstrate that: (i) CS, when incorporated as a trainable module inside JSCC, enables more efficient use of channel resources than conventional

latent compression; (ii) adaptive rate control that responds to both content and SNR is important for practical deployment under fluctuating bandwidth and channel conditions; (iii) explicit SNR awareness through conditioning mechanisms such as FiLM significantly enhances robustness; and (iv) shifting the design objective from pixel-level reconstruction to task-oriented feature preservation can improve bandwidth efficiency and latency for inference-driven applications, while also reducing the need to expose or reconstruct raw visual content. These findings position deep compressive and task-oriented JSCC as promising foundations for future wireless edge intelligence systems.

6.2 Future Work

The results presented in this thesis open several promising directions for further research and practical deployment.

- **Extension to video and temporal tasks:** The current frameworks focus on still images and frame-level tasks. Extending compressive and task-oriented JSCC to video transmission would require temporal models that exploit motion and inter-frame redundancy while adapting rates over time. Applications include action recognition, tracking, and streaming in dynamic bandwidth environments.
- **Multiuser and network-level scenarios:** All proposed architectures operate in a point-to-point setting. Future work can investigate multiuser, broadcast, and relay networks where multiple receivers with heterogeneous SNR values and task requirements share limited resources. Combining the proposed rate-adaptive CS mechanisms with resource allocation, scheduling, and interference management is an important step toward system-level deployment.
- **Robustness to imperfect CSI and feedback:** The rate adaptation and SNR conditioning modules in this thesis assume accurate SNR estimates and reliable feedback links. In practice, channel state information can be noisy, delayed, or quantized. Designing JSCC schemes that remain robust under CSI uncertainty, limited feedback, or partial knowledge of channel statistics is a key open challenge.

- **Hardware constraints and on-device implementation:** Although the proposed models are lightweight relative to many deep architectures, further work is needed to tailor them for embedded and mobile hardware. This includes model compression, quantization, pruning, and hardware-aware architecture search, as well as exploring analog and mixed-signal implementations of learned CS sampling.
- **Security, privacy, and explainability:** Task-oriented communication can reduce exposure of raw visual content, but systematic analysis of privacy leakage through transmitted features remains limited. Future research could integrate formal privacy guarantees, adversarial robustness, and explainable feature visualizations to better understand and control what information is revealed to the receiver or intermediate nodes.
- **Theoretical analysis and performance bounds:** While empirical results show clear advantages over conventional schemes, there is a need for theoretical frameworks that characterize the rate-distortion-reliability behaviour of deep compressive and semantic JSCC. Potential directions include information-theoretic bounds for task-based distortion measures, analysis of learned sensing matrices, and convergence properties of SNR-conditioned training.
- **Generalization to other modalities and tasks:** The TCS-JSCC and FRCS-JSCC frameworks can be extended to other vision tasks such as segmentation, multi-object detection, tracking, and multimodal applications that combine images with text or audio. Adapting the proposed ideas to non-visual modalities, such as sensor networks, LiDAR point clouds, or biomedical signals, is another promising direction.
- **Adaptive learning:** In many practical scenarios, data distributions, tasks, and channel statistics evolve over time. Integrating continual learning, meta-learning, or online adaptation mechanisms into compressive and task-oriented JSCC frameworks would allow systems to update their models in the field without exhaustive retraining.

Addressing these directions will strengthen the theoretical foundations and practical relevance of compressive and task-oriented JSCC. The methodologies developed in this thesis provide a

starting point for building future communication systems that are deeply integrated with machine learning workloads, adapt their behaviour to both channel and task conditions, and make efficient use of scarce wireless resources.

Bibliography

- [1] M. Chandra, D. Agarwal, and A. Bansal, “Image Transmission Through Wireless Channel: A Review,” in *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, IEEE, 2016, pp. 1–4.
- [2] S. Weber, J. G. Andrews, and N. Jindal, “An Overview of the Transmission Capacity of Wireless Networks,” *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3593–3604, 2010.
- [3] N. Thomos, N. V. Boulgouris, and M. G. Strintzis, “Wireless Image Transmission Using Turbo Codes and Optimal Unequal Error Protection,” *IEEE Trans. Image Process.*, vol. 14, no. 11, pp. 1890–1901, 2005.
- [4] F. Zhai, Y. Eisenberg, and A. K. Katsaggelos, “Joint Source-Channel Coding for Video Communications,” *Handbook of Image and Video Processing*, vol. 2, 2005.
- [5] M. Fresia, F. Pérez-Cruz, H. V. Poor, and S. Verdú, “Joint Source and Channel Coding,” *IEEE Signal Process. Mag.*, vol. 27, no. 6, pp. 104–113, 2010.
- [6] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, “Deep Joint Source-Channel Coding for Wireless Image Transmission,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 3, pp. 567–579, 2019.
- [7] D. B. Kurka and D. Gündüz, “DeepJSCC-F: Deep Joint Source-Channel Coding of Images With Feedback,” *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 178–193, 2020.
- [8] D. Gündüz, M. A. Wigger, T.-Y. Tung, P. Zhang, and Y. Xiao, “Joint Source–Channel Coding: Fundamentals and Recent Progress in Practical Designs,” *Proc. IEEE*, 2024.
- [9] L. X. Nguyen, A. D. Raha, P. S. Aung, D. Niyato, Z. Han, and C. S. Hong, “A Contemporary Survey on Semantic Communications: Theory of Mind, Generative AI, and Deep Joint Source-Channel Coding,” *IEEE Commun. Surveys Tuts.*, 2025.

- [10] D. L. Donoho, “Compressed Sensing,” *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [11] V. Abolghasemi, S. Ferdowsi, and S. Sanei, “A Gradient-Based Alternating Minimization Approach for Optimization of the Measurement Matrix in Compressive Sensing,” *Signal Processing*, vol. 92, no. 4, pp. 999–1009, 2012.
- [12] M. Rani, S. B. Dhok, and R. B. Deshmukh, “A Systematic Review of Compressive Sensing: Concepts, Implementations and Applications,” *IEEE Access*, vol. 6, pp. 4875–4894, 2018.
- [13] A. L. Machidon and V. Pejović, “Deep Learning for Compressive Sensing: A Ubiquitous Systems Perspective,” *Artificial Intelligence Review*, vol. 56, no. 4, pp. 3619–3658, 2023.
- [14] S. Jakubczak and D. Katabi, “SoftCast: One-Size-Fits-All Wireless Video,” in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 449–450.
- [15] T.-Y. Tung and D. Gündüz, “SparseCast: Hybrid Digital-Analog Wireless Image Transmission Exploiting Frequency-Domain Sparsity,” *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2451–2454, 2018.
- [16] J. Chen et al., “LDPC Coded Compressive Sensing for Joint Source-Channel Coding in Wireless Sensor Networks,” *IEEE Trans. Veh. Technol.*, vol. 72, no. 2, pp. 2145–2160, 2022.
- [17] J. Chen et al., “Tree-Search Techniques for Joint Iterative Compressive Sensing and LDPC Decoding in Wireless Sensor Networks,” *IEEE Sensors J.*, vol. 23, no. 1, pp. 434–451, 2022.
- [18] X. Song, X. Peng, J. Xu, G. Shi, and F. Wu, “Distributed Compressive Sensing for Cloud-Based Wireless Image Transmission,” *IEEE Trans. Multimedia*, vol. 19, no. 6, pp. 1351–1364, 2017.
- [19] S. Feizi and M. Médard, “A Power Efficient Sensing/Communication Scheme: Joint Source-Channel-Network Coding by Using Compressive Sensing,” in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2011, pp. 1048–1054.

- [20] L. Zheng, T. Zhu, and X. Ma, “Block Compressed Sensing-Based Joint Source-Channel Coding for Wireless Image Transmission,” in *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*, IEEE, 2020, pp. 13–18.
- [21] A. Shirazinia, S. Chatterjee, and M. Skoglund, “Joint Source-Channel Vector Quantization for Compressed Sensing,” *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3667–3681, 2014.
- [22] M. U. Lokumarambage, V. S. S. Gowrisetty, H. Rezaei, T. Sivalingam, N. Rajatheva, and A. Fernando, “Wireless End-to-End Image Transmission System Using Semantic Communications,” *IEEE Access*, vol. 11, pp. 37 149–37 163, 2023.
- [23] Y. Chen, B. Tan, J. Wu, and D. Hu, “PJSCC: A Puncturing-Based Joint Source Channel Coding Scheme With Hierarchical Down-Sampling Layer,” in *ICASSP 2024 IEEE Int. Conf. on Acoust., Speech and Signal Process. (ICASSP)*, IEEE, 2024, pp. 8976–8980.
- [24] C. Karamanli, T.-Y. Tung, and D. Guenduez, “Model-Driven Deep Joint Source-Channel Coding Over Time-Varying Channels,” in *WSA & SCC 2023; 26th International ITG Workshop on Smart Antennas and 13th Conference on Systems, Communications, and Coding*, VDE, 2023, pp. 1–6.
- [25] K. Yang, S. Wang, J. Dai, X. Qin, K. Niu, and P. Zhang, “SwinJSCC: Taming Swin Transformer for Deep Joint Source-Channel Coding,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 11, no. 1, pp. 90–104, 2024.
- [26] Z. Lyu, G. Zhu, J. Xu, B. Ai, and S. Cui, “Semantic Communications for Image Recovery and Classification via Deep Joint Source and Channel Coding,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 8, pp. 8388–8404, 2024.
- [27] X. Han et al., “SCSC: A Novel Standards-Compatible Semantic Communication Framework for Image Transmission,” *IEEE Trans. Commun.*, vol. 73, no. 8, pp. 5682–5698, 2025.
- [28] R. Hemalatha, S. Radha, and S. Sudharsan, “Energy-Efficient Image Transmission in Wireless Multimedia Sensor Networks Using Block-Based Compressive Sensing,” *Computers & Electrical Engineering*, vol. 44, pp. 67–79, 2015.

- [29] J. Zhang, Q. Xiang, Y. Yin, C. Chen, and X. Luo, “Adaptive Compressed Sensing for Wireless Image Sensor Networks,” *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 4227–4242, 2017.
- [30] S. Chen, Y. Liu, X. Gao, and Z. Han, “MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices,” in *Chinese Conf. on Biometric Recognition*, Springer, 2018, pp. 428–438.
- [31] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition,” in *Proc. IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, 2019, pp. 4690–4699.
- [32] G. Xin, P. Fan, and K. B. Letaief, “Semantic Communication: A Survey of Its Theoretical Development,” *Entropy*, vol. 26, no. 2, p. 102, 2024.
- [33] B. Zhang, Z. Qin, and G. Y. Li, “Compression Ratio Learning and Semantic Communications for Video Imaging,” *IEEE J. Sel. Top. Signal Process.*, vol. 18, no. 3, pp. 312–324, 2024.
- [34] Z. Xie, L. Liu, and Z. Chen, “Image Compressed Sensing: From Deep Learning to Adaptive Learning,” *Knowledge-Based Systems*, vol. 293, p. 111 659, 2024.
- [35] T. M. Getu, G. Kaddoum, and M. Bennis, “A Survey on Goal-Oriented Semantic Communication: Techniques, Challenges, and Future Directions,” *IEEE Access*, vol. 12, pp. 51 223–51 274, 2024.
- [36] T. M. Getu, G. Kaddoum, and M. Bennis, “Semantic Communication: A Survey on Research Landscape, Challenges, and Future Directions,” *Proc. IEEE*, vol. 112, no. 11, pp. 1649–1685, 2025.
- [37] R. E. Van Dyck and D. J. Miller, “Transport of Wireless Video Using Separate, Concatenated, and Joint Source-Channel Coding,” *Proc. IEEE*, vol. 87, no. 10, pp. 1734–1750, 2002.
- [38] J. Dai, P. Zhang, K. Niu, S. Wang, Z. Si, and X. Qin, “Communication Beyond Transmitting Bits: Semantics-Guided Source and Channel Coding,” *IEEE Wireless Commun.*, vol. 30, no. 4, pp. 170–177, 2022.

- [39] Y. Zhong, F. Alajaji, and L. L. Campbell, “On the Joint Source-Channel Coding Error Exponent for Discrete Memoryless Systems,” *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1450–1468, 2006.
- [40] V. Kostina and S. Verdú, “Lossy Joint Source-Channel Coding in the Finite Blocklength Regime,” *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 2545–2575, 2013.
- [41] R. C. Jain, “An Introduction to Joint Source and Channel Coding,” *IETE J. Educ.*, vol. 46, no. 3, pp. 121–127, 2005.
- [42] P. Duhamel and M. Kieffer, *Joint Source-Channel Decoding: A Cross-Layer Perspective With Applications in Video Broadcasting*. Academic Press, 2009.
- [43] T.-Y. Tung, D. B. Kurka, M. Jankowski, and D. Gündüz, “DeepJSCC-Q: Constellation Constrained Deep Joint Source-Channel Coding,” *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 4, pp. 720–731, 2022.
- [44] M. Jankowski, D. Gündüz, and K. Mikolajczyk, “Deep Joint Source-Channel Coding for Wireless Image Retrieval,” in *ICASSP 2020 IEEE Int. Conf. on Acoust., Speech and Signal Process. (ICASSP)*, IEEE, 2020, pp. 5070–5074.
- [45] D. B. Kurka and D. Gündüz, “Bandwidth-Agile Image Transmission With Deep Joint Source-Channel Coding,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 8081–8095, 2021.
- [46] C. Bian, Y. Shao, H. Wu, E. Ozfatura, and D. Gündüz, “Process-and-Forward: Deep Joint Source-Channel Coding Over Cooperative Relay Networks,” *IEEE J. Sel. Areas Commun.*, vol. 43, no. 4, pp. 1118–1134, 2025.
- [47] S. Xie et al., “Deep Learning-Based Adaptive Joint Source-Channel Coding Using Hypernetworks,” in *2024 IEEE Int. Mediterranean Conf. on Commun. and Netw. (MeditCom)*, IEEE, 2024, pp. 191–196.
- [48] L. Luo, Z. He, J. Wu, H. Guo, and C. Zhu, “Adaptive Deep Joint Source-Channel Coding for One-to-Many Wireless Image Transmission,” *IEEE Trans. Broadcast.*, vol. 71, no. 3, pp. 914–929, 2025.

- [49] G. He, S. Zhang, and T. Jiang, “Residual Vector Quantized Diffusion Model-Based Joint Source-Channel Coding for Task-Oriented Agent Communication,” *IEEE Trans. Wireless Commun.*, vol. 24, no. 6, pp. 4663–4676, 2025.
- [50] J. Feng, G. Cao, J. Liu, Z. Wang, Z. Liu, and Y. Ding, “Deep Joint Source-Channel Coding for Wireless Image Transmission With Auxiliary Semantics,” *Phys. Commun.*, p. 102 836, 2025.
- [51] J. Xu, B. Ai, W. Chen, A. Yang, P. Sun, and M. Rodrigues, “Wireless Image Transmission Using Deep Source Channel Coding With Attention Modules,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 2315–2328, 2021.
- [52] H. Chang, L. Ma, and D. Qin, “SNR Adaptive Attention Joint Source Channel Coding for Enhanced Image Transmission in Wireless Vehicular System,” *IEEE Trans. Veh. Technol.*, 2025.
- [53] W. Jiang, W. Chen, and B. Ai, “Deep Joint Source Channel Coding With Attention Modules Over MIMO Channels,” in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, IEEE, 2024, pp. 1–5.
- [54] H. Zhang and M. Tao, “SNR-EQ-JSCC: Joint Source-Channel Coding With SNR-Based Embedding and Query,” *IEEE Wireless Commun. Lett.*, vol. 14, no. 3, pp. 881–885, 2025.
- [55] H. Zhang, M. Wu, L. Qiao, L. Liu, Z. Han, and Z. Gao, “Residual Cross-Attention Transformer-Based Multi-User CSI Feedback With Deep Joint Source-Channel Coding,” *IEEE Wireless Commun. Lett.*, vol. 14, no. 8, pp. 2481–2485, 2025.
- [56] H. Wu, Y. Shao, E. Ozfatura, K. Mikolajczyk, and D. Gündüz, “Transformer-Aided Wireless Image Transmission With Channel Feedback,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 11 904–11 919, 2024.
- [57] D. Persson, J. Kron, M. Skoglund, and E. G. Larsson, “Joint Source-Channel Coding for the MIMO Broadcast Channel,” *IEEE Trans. Signal Process.*, vol. 60, no. 4, pp. 2085–2090, 2011.

- [58] C. Bian, Y. Shao, H. Wu, and D. Gündüz, “Space-Time Design for Deep Joint Source Channel Coding of Images Over MIMO Channels,” in *2023 IEEE 24th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, IEEE, 2023, pp. 616–620.
- [59] T. Holliday, A. J. Goldsmith, and H. V. Poor, “Joint Source and Channel Coding for MIMO Systems: Is It Better to Be Robust or Quick?” *IEEE Trans. Inf. Theory*, vol. 54, no. 4, pp. 1393–1405, 2008.
- [60] H. Wu, Y. Shao, C. Bian, K. Mikołajczyk, and D. Gündüz, “Deep Joint Source-Channel Coding for Adaptive Image Transmission Over MIMO Channels,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 10, pp. 15 002–15 017, 2024.
- [61] M. Yang, C. Bian, and H.-S. Kim, “OFDM-Guided Deep Joint Source Channel Coding for Wireless Multipath Fading Channels,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 2, pp. 584–599, 2022.
- [62] B. Dong, B. Cao, G. Gui, and Q. Zhang, “Robust Deep Joint Source-Channel Coding Enabled Distributed Image Transmission With Imperfect Channel State Information,” *IEEE Trans. Wireless Commun.*, 2025.
- [63] J. Xu, B. Ai, N. Wang, and W. Chen, “Deep Joint Source-Channel Coding for CSI Feedback: An End-to-End Approach,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 260–273, 2022.
- [64] T. Wu, Z. Chen, M. Tao, X. Xu, W. Zhang, and P. Zhang, “MambaJSCC: Deep Joint Source-Channel Coding With Visual State Space Model,” in *GLOBECOM 2024 IEEE Global Communications Conference*, IEEE, 2024, pp. 1677–1682.
- [65] M. Ding, J. Li, M. Ma, and X. Fan, “SNR-Adaptive Deep Joint Source-Channel Coding for Wireless Image Transmission,” in *ICASSP 2021 IEEE Int. Conf. on Acoust., Speech and Signal Process. (ICASSP)*, IEEE, 2021, pp. 1555–1559.
- [66] X. Bao, M. Jiang, and H. Zhang, “ADJSCC-L: SNR-Adaptive JSCC Networks for Multi-Layer Wireless Image Transmission,” in *2021 7th Int. Conf. on Computer and Communications (ICCC)*, IEEE, 2021, pp. 1812–1816.

- [67] B. Chen and J. Zhang, "Content-Aware Scalable Deep Compressed Sensing," *IEEE Trans. Image Process.*, vol. 31, pp. 5412–5426, 2022.
- [68] Z. Gao et al., "Integrated Sensing and Communication With mmWave Massive MIMO: A Compressed Sampling Perspective," *IEEE Trans. Wireless Commun.*, vol. 22, no. 3, pp. 1745–1762, 2022.
- [69] Y. Chen, Y. C. Eldar, and A. J. Goldsmith, "Shannon meets Nyquist: Capacity of Sampled Gaussian Channels," *IEEE Trans. on Inf. Th.*, vol. 59, no. 8, pp. 4889–4914, 2013.
- [70] M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4053–4085, 2011.
- [71] H. Rauhut, J. Romberg, and J. Tropp, "Restricted isometries for partial random circulant matrices," *Applied and Computational Harmonic Analysis*, vol. 32, no. 2, pp. 242–254, 2012.
- [72] H. Du et al., "Semantic Communications for Wireless Sensing: RIS-Aided Encoding and Self-Supervised Decoding," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 8, pp. 2547–2562, 2023.
- [73] W. Shi, F. Jiang, S. Liu, and D. Zhao, "Image Compressed Sensing Using Convolutional Neural Network," *IEEE Trans. Image Process.*, vol. 29, pp. 375–388, 2019.
- [74] A. Majumdar, "An Autoencoder Based Formulation for Compressed Sensing Reconstruction," *Magn. Reson. Imaging*, vol. 52, pp. 62–68, 2018.
- [75] R. Nan, G. Sun, Z. Wang, and X. Ren, "Research on Image Reconstruction of Compressed Sensing Based on a Multi-Feature Residual Network," *Sensors*, vol. 20, no. 15, p. 4202, 2020.
- [76] M. Shen, H. Gan, C. Ning, Y. Hua, and T. Zhang, "TransCS: A Transformer-Based Hybrid Architecture for Image Compressed Sensing," *IEEE Trans. Image Process.*, vol. 31, pp. 6991–7005, 2022.

- [77] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, “ReconNet: Non-Iterative Reconstruction of Images From Compressively Sensed Measurements,” in *Proc. IEEE Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, 2016, pp. 449–458.
- [78] H. Yao, F. Dai, S. Zhang, Y. Zhang, Q. Tian, and C. Xu, “DR2-Net: Deep Residual Reconstruction Network for Image Compressive Sensing,” *Neurocomputing*, vol. 359, pp. 483–493, 2019.
- [79] Z. Zhang, Y. Liu, J. Liu, F. Wen, and C. Zhu, “AMP-Net: Denoising-Based Deep Unfolding for Compressive Image Sensing,” *IEEE Trans. Image Process.*, vol. 30, pp. 1487–1500, 2020.
- [80] J. Zhang and B. Ghanem, “ISTA-Net: Interpretable Optimization-Inspired Deep Network for Image Compressive Sensing,” in *Proc. IEEE Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, 2018, pp. 1828–1837.
- [81] J. Sun, H. Li, Z. Xu, et al., “Deep ADMM-Net for Compressive Sensing MRI,” *Adv. Neural Inf. Process. Syst.*, vol. 29, 2016.
- [82] J. W. Choi, B. Shim, Y. Ding, B. Rao, and D. I. Kim, “Compressed Sensing for Wireless Communications: Useful Tips and Tricks,” *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1527–1550, 2017.
- [83] K. Hayashi, M. Nagahara, and T. Tanaka, “A User’s Guide to Compressed Sensing for Communications Systems,” *IEICE Trans. Commun.*, vol. 96, no. 3, pp. 685–712, 2013.
- [84] Z. Gao, L. Dai, S. Han, Z. Wang, L. Hanzo, et al., “Compressive Sensing Techniques for Next-Generation Wireless Communications,” *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 144–153, 2018.
- [85] Y. Jiang et al., “High Quality Compression and Transmission of Remote Sensing Images Based on Semantic Communication,” *IEEE Trans. Sustainable Comput.*, vol. 10, no. 5, pp. 843–857, 2025.
- [86] C. Liu, C. Guo, Y. Yang, and N. Jiang, “Adaptable Semantic Compression and Resource Allocation for Task-Oriented Communications,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 10, no. 3, pp. 769–782, 2023.

- [87] J. Guo, B. Sun, J. Zhu, W. Feng, S. He, and W. Xue, “SCSC: The Super Compressed Semantic Communication Transmission Method for Images and Video Frames,” *IEEE Internet Things J.*, vol. 12, no. 20, pp. 43 731–43 741, 2025.
- [88] J. Dai et al., “Joint Communication and Computation Design for Secure Integrated Sensing and Semantic Communication System,” *Sci. China Inf. Sci.*, vol. 68, no. 3, p. 132 301, 2025.
- [89] J. Huang, D. Li, C. Huang, X. Qin, and W. Zhang, “Joint Task and Data-Oriented Semantic Communications: A Deep Separate Source-Channel Coding Scheme,” *IEEE Internet Things J.*, vol. 11, no. 2, pp. 2255–2272, 2023.
- [90] Z. Zhao et al., “Compression Ratio Allocation for Probabilistic Semantic Communication With RSMA,” *IEEE Trans. Commun.*, vol. 73, no. 9, pp. 7304–7318, 2025.
- [91] J. Wang, W. Wang, and J. Chen, “Adaptive Rate Block Compressive Sensing Based on Statistical Characteristics Estimation,” *IEEE Trans. Image Process.*, vol. 31, pp. 734–747, 2021.
- [92] Y. Huang, B. Chen, N. Li, B. An, S.-T. Xia, and Y. Wang, “MB-RACS: Measurement-Bounds-Based Rate-Adaptive Image Compressed Sensing Network,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 7, pp. 5460–5474, 2025.
- [93] Z. Hong, Q. Yan, Z. Li, T. Zhan, and Y. Wang, “Photon-counting Underwater Optical Wireless Communication for Reliable Video Transmission Using Joint Source-Channel Coding Based on Distributed Compressive Sensing,” *Sensors*, vol. 19, no. 5, p. 1042, 2019.
- [94] H. Wang, W. Zhang, Y. Liang, and Y. Liu, “Efficient Reconstruction Architecture of Compressed Sensing and Integrated Source-Channel Decoder based on Reed Solomon Code,” *IEEE Communications Letters*, vol. 24, no. 2, pp. 239–243, 2019.
- [95] A. HesamMohseni, M. Babaie-Zadeh, and C. Jutten, “Inflating Compressed Samples: A Joint Source-Channel Coding Approach for Noise-Resistant Compressed Sensing,” in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2009, pp. 2957–2960.

- [96] A. Moslemi, “Sparse Representation Learning Using ℓ_{1-2} Compressed Sensing and Rank-Revealing QR Factorization,” *Eng. Appl. Artif. Intell.*, vol. 125, p. 106 663, 2023.
- [97] S. Guo et al., “A Survey on Semantic Communication Networks: Architecture, Security, and Privacy,” *IEEE Commun. Surveys Tuts.*, 2024.
- [98] A. Nosratinia, J. Lu, and B. Aazhang, “Source-Channel Rate Allocation for Progressive Transmission of Images,” *IEEE Trans. Commun.*, vol. 51, no. 2, pp. 186–196, 2003.
- [99] V. Joseph, S. Borst, and M. I. Reiman, “Optimal Rate Allocation for Adaptive Wireless Video Streaming in Networks With User Dynamics,” in *IEEE INFOCOM 2014 IEEE Conf. on Comput. Commun.*, IEEE, 2014, pp. 406–414.
- [100] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, “An Experimental Evaluation of Rate-Adaptive Video Players Over HTTP,” *Signal Process. Image Commun.*, vol. 27, no. 4, pp. 271–287, 2012.
- [101] K. Sayood, H. H. Otu, and N. Demir, “Joint Source/Channel Coding for Variable Length Codes,” *IEEE Trans. Commun.*, vol. 48, no. 5, pp. 787–794, 2002.
- [102] M. Rao, N. Farsad, and A. Goldsmith, “Variable Length Joint Source-Channel Coding of Text Using Deep Neural Networks,” in *2018 IEEE 19th Int. Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, IEEE, 2018, pp. 1–5.
- [103] M. Song, N. Ma, C. Dong, X. Xu, and P. Zhang, “Deep Joint Source-Channel Coding for Wireless Image Transmission With Adaptive Models,” *Electronics*, vol. 12, no. 22, p. 4637, 2023.
- [104] X. Sun, J. Yu, C. Wu, Y. Li, and H. Zhang, “Hierarchical Multi-Granularity Joint Source-Channel Coding for Image Semantic Transmission,” *IEEE Wir. Comms. Lts.*, vol. 13, no. 12, pp. 3325–3329, 2024.
- [105] C. Bian, Y. Shao, and D. Gündüz, “DeepJSCC-L++: Robust and Bandwidth-Adaptive Wireless Image Transmission,” in *GLOBECOM 2023 IEEE Global Communications Conference*, IEEE, 2023, pp. 3148–3154.

- [106] H. Wu, Y. Shao, K. Mikolajczyk, and D. Gündüz, “Channel-Adaptive Wireless Image Transmission With OFDM,” *IEEE Wireless Commun. Lett.*, vol. 11, no. 11, pp. 2400–2404, 2022.
- [107] H. Li, G. Zhang, K. Zhou, Y. Cai, and G. Yu, “Coarse-to-Fine: A Dual-Phase Channel-Adaptive Method for Wireless Image Transmission,” *IEEE Trans. Wireless Commun.*, 2025.
- [108] H. Choi and D. Seo, “Feature Importance-Aware Deep Joint Source-Channel Coding for Computationally Efficient and Adjustable Image Transmission,” *IEEE Internet of Things Journal*, vol. 13, no. 12, pp. 27 893–27 911, 2026.
- [109] C. Li, Y. Zeng, Z. Ye, Q. Peng, Y. Sun, and C. Wang, “Adaptive Nonlinear Deep Coding using Hybrid Attention for Wireless Image Transmission,” *Signal, Image and Video Processing*, vol. 19, no. 3, p. 251, 2025.
- [110] A. D. Raha, A. Adhikary, M. Gain, Y. Park, W. Saad, and C. S. Hong, “DD-JSCC: Dynamic Deep Joint Source-Channel Coding for Semantic Communications,” in *Int. Conf. on Comms. (ICC)*, IEEE, 2025, pp. 3754–3759.
- [111] P. Yang, G. Zhang, and Y. Cai, “Rate-Adaptive Generative Semantic Communication Using Conditional Diffusion Models,” *IEEE Wireless Commun. Lett.*, vol. 14, no. 2, pp. 539–543, 2024.
- [112] N. Guler and Z. B. Hazem, “Semantic Communication-Based Convolutional Neural Network for Enhanced Image Classification,” *Franklin Open*, vol. 9, p. 100 192, 2024.
- [113] E. Kutay and A. Yener, “Classification-Oriented Semantic Wireless Communications,” in *ICASSP 2024 IEEE Int. Conf. on Acoust., Speech and Signal Process. (ICASSP)*, IEEE, 2024, pp. 9096–9100.
- [114] V. Kouvakis, S. E. Trevlakis, and A.-A. A. Boulogeorgos, “Semantic Communications for Image-Based Sign Language Transmission,” *IEEE Open J. Commun. Soc.*, vol. 5, pp. 1088–1100, 2024.

- [115] J. Guo et al., “Multi-Scale Semantic Communication for Object Detection: Single and Cross-Domain Scenarios,” *IEEE Trans. Wireless Commun.*, vol. 24, no. 7, pp. 6195–6210, 2025.
- [116] X. Song et al., “Knowledge Graph Driven UAV Cognitive Semantic Communication Systems for Efficient Object Detection,” in *ICC 2024 IEEE Int. Conf. on Commun. (ICC)*, IEEE, 2024, pp. 1685–1690.
- [117] Z. Lu et al., “Semantics-Empowered communications: A Tutorial-Cum-Survey,” *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 41–79, 2023.
- [118] N. Islam and S. Shin, “Deep Learning in Physical Layer: Review on Data Driven End-to-End Communication Systems and Their Enabling Semantic Applications,” *IEEE Open Journal of the Communications Society*, vol. 5, pp. 4207–4240, 2024.
- [119] A. Li, X. Liu, G. Wang, and P. Zhang, “Domain Knowledge Driven Semantic Communication for Image Transmission Over Wireless Channels,” *IEEE Wireless Commun. Lett.*, vol. 12, no. 1, pp. 55–59, 2022.
- [120] S. Min, H. Yao, H. Xie, Z.-J. Zha, and Y. Zhang, “Domain-Oriented Semantic Embedding for Zero-Shot Learning,” *IEEE Trans. Multimedia*, vol. 23, pp. 3919–3930, 2020.
- [121] K. Zhou, G. Zhang, Y. Cai, Q. Hu, G. Yu, and A. L. Swindlehurst, “Feature Allocation for Semantic Communication With Space-Time Importance Awareness,” *IEEE Trans. Wireless Commun.*, vol. 24, no. 10, pp. 8847–8862, 2025.
- [122] G. Shi, Y. Xiao, Y. Li, and X. Xie, “From Semantic Communication to Semantic-Aware Networking: Model, Architecture, and Open Problems,” *IEEE Commun. Mag.*, vol. 59, no. 8, pp. 44–50, 2021.
- [123] S. Tang et al., “Retrieval-Augmented Generation for GenAI-Enabled Semantic Communications,” *IEEE Wireless Commun.*, 2025.
- [124] M. Shen et al., “Secure Semantic Communications: Challenges, Approaches, and Opportunities,” *IEEE Netw.*, vol. 38, no. 4, pp. 197–206, 2023.

- [125] M. Gong, S. Wang, F. Ye, and S. Bi, “Compression Before Fusion: Broadcast Semantic Communication System for Heterogeneous Tasks,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 12, pp. 19 428–19 443, 2024.
- [126] L. Wang, W. Wu, F. Zhou, Z. Yang, Z. Qin, and Q. Wu, “Adaptive Resource Allocation for Semantic Communication Networks,” *IEEE Trans. Commun.*, vol. 72, no. 11, pp. 6900–6916, 2024.
- [127] Y. Wang, M. Chen, W. Saad, T. Luo, S. Cui, and H. V. Poor, “Performance Optimization for Semantic Communications: An Attention-Based Learning Approach,” in *2021 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2021, pp. 1–6.
- [128] Z. Yang, M. Chen, Z. Zhang, and C. Huang, “Energy Efficient Semantic Communication Over Wireless Networks With Rate Splitting,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 5, pp. 1484–1495, 2023.
- [129] A. Krizhevsky, “Learning Multiple Layers of Features From Tiny Images,” University of Toronto, Tech. Rep., 2009. [Online]. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- [130] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *2009 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2009, pp. 248–255.
- [131] Kodak, *Kodak Color Management*, <https://r0k.us/graphics/kodak/>, Accessed: January 8, 2013.
- [132] S. Barbarossa, D. Communiello, E. Grassucci, F. Pezone, S. Sardellitti, and P. Di Lorenzo, “Semantic Communications Based on Adaptive Generative Models and Information Bottleneck,” *IEEE Commun. Mag.*, vol. 61, no. 11, pp. 36–41, 2023.
- [133] S. Yang, P. Luo, C.-C. Loy, and X. Tang, “WIDER FACE: A Face Detection Benchmark,” in *Proc. IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, 2016, pp. 5525–5533.

- [134] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments,” University of Massachusetts, Amherst, Technical Report 07-49, 2007. [Online]. Available: <https://vis-www.cs.umass.edu/lfw/lfw.pdf>.
- [135] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, “The MegaFace Benchmark: 1 Million Faces for Recognition at Scale,” in *Proc. IEEE Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, 2016, pp. 4873–4882.
- [136] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Learning Face Representation From Scratch,” in *Proc. IEEE Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, IEEE, 2014, pp. 1–10.
- [137] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “VGGFace2: A Dataset for Recognising Faces Across Pose and Age,” in *2018 13th IEEE Int. Conf. on Automatic Face & Gesture Recognition (FG 2018)*, IEEE, 2018, pp. 67–74.
- [138] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, “MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition,” in *Eur. Conf. on Computer Vision (ECCV)*, Springer, 2016, pp. 87–102.
- [139] I. DeAndres-Tame et al., “FRCSyn Challenge at CVPR 2024: Face Recognition Challenge in the Era of Synthetic Data,” in *Proc. IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, 2024, pp. 3173–3183.
- [140] T.-Y. Lin et al., “Microsoft Coco: Common Objects in Context,” in *European conference on computer vision*, Springer, 2014, pp. 740–755.
- [141] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (voc) Challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [142] M. Cordts et al., “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- [143] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene Parsing Through ADE20K Dataset,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.
- [144] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision Meets Robotics: The KITTI Dataset,” *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. DOI: [10.1177/0278364913491297](https://doi.org/10.1177/0278364913491297).
- [145] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [146] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.
- [147] M. Yang, B. Liu, B. Wang, and H.-S. Kim, “Diffusion-Aided Joint Source Channel Coding for High Realism Wireless Image Transmission,” *IEEE Trans. on Mach. Lea. in Comms. and Net.*, vol. 3, pp. 1227–1243, 2025.
- [148] W. Shi, F. Jiang, S. Zhang, and D. Zhao, “Deep Networks for Compressed Image Sensing,” in *2017 IEEE Int. Conf. on Multimedia and Expo (ICME)*, IEEE, 2017, pp. 877–882.
- [149] J. Ballé, V. Laparra, and E. P. Simoncelli, “Density Modeling of Images Using a Generalized Normalization Transformation,” *arXiv preprint arXiv:1511.06281*, 2015.
- [150] S. Zhou, Y. He, Y. Liu, C. Li, and J. Zhang, “Multi-Channel Deep Networks for Block-Based Image Compressive Sensing,” *IEEE Trans. Multimedia*, vol. 23, pp. 2627–2640, 2020.
- [151] Y. Yu, B. Wang, and L. Zhang, “Saliency-Based Compressive Sampling for Image Signals,” *IEEE Signal Process. Lett.*, vol. 17, no. 11, pp. 973–976, 2010.
- [152] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, “A Comprehensive Survey on Model Compression and Acceleration,” *Artificial Intelligence Review*, vol. 53, no. 7, pp. 5113–5155, 2020.

- [153] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising,” *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, 2017.
- [154] J. Ballé et al., “Nonlinear Transform Coding,” *IEEE J. Sel. Top. Signal Process.*, vol. 15, no. 2, pp. 339–353, 2020.
- [155] S. Wang, J. Dai, X. Qin, Z. Si, K. Niu, and P. Zhang, “Improved Nonlinear Transform Source-Channel Coding to Catalyze Semantic Communications,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 5, pp. 1022–1037, 2023.
- [156] M. Xu, Y. Liang, C.-T. Lam, B. Ng, and S.-K. Im, “Low Complexity Rate-Adaptive Deep Joint Source Channel Coding for Wireless Image Transmission Using Tensor-Train Decomposition,” in *2022 7th Int. Conf. on Signal and Image Processing (ICSIP)*, IEEE, 2022, pp. 707–712.
- [157] M. Xu, C.-T. Lam, Y. Liang, B. Ng, and S.-K. Im, “Low-Rank Decomposition for Rate-Adaptive Deep Joint Source-Channel Coding,” in *2022 IEEE 8th Int. Conf. on Computer and Communications (ICCC)*, IEEE, 2022, pp. 58–64.
- [158] M. Xu, C.-T. Lam, Y. Liang, B. Ng, X. Yuan, and S.-K. Im, “TDJSCC: Low Complexity and Bandwidth Efficient Deep Joint Source-Channel Coding With OFDM,” *IEEE Access*, vol. 13, pp. 150 244–150 257, 2025.
- [159] S. F. Yilmaz, C. Karamanlı, and D. Gündüz, “Distributed Deep Joint Source-Channel Coding Over a Multiple Access Channel,” in *ICC 2023 IEEE Int. Conf. on Commun. (ICC)*, IEEE, 2023, pp. 1400–1405.
- [160] H. Gao, G. Yu, and Y. Cai, “Rate Adaptive Mechanism for Semantic Communication Systems: A Robustness Verification Approach,” *IEEE Netw.*, vol. 38, no. 4, pp. 216–223, 2023.
- [161] Y. Jia, G. Chen, and K. Qi, “CTRCNet: A CNN–Transformer Semantic Communication System for Image Transmission With Adaptive Rate Control,” *J. Electron. Imaging*, vol. 34, no. 2, pp. 023 052–023 052, 2025.

- [162] T. Guo, S. Gu, Y. Wu, Q. Zhang, and W. Xiang, “A Novel Lightweight Deep Joint Source-Channel Coding Framework: Using 1D-CNN for SNR and Compression Rate Adaptation,” in *2025 IEEE/CIC Int. Conf. on Commun. in China (ICCC)*, IEEE, 2025, pp. 1–6.
- [163] F. Wang, X. Chen, and X. Deng, “Wireless Adaptive Image Transmission Over OFDM Channels Based on Entropy Model,” *IEEE Wireless Commun. Lett.*, vol. 13, no. 10, pp. 2902–2906, 2024.
- [164] Z. He, J. Cai, and C. W. Chen, “Joint Source Channel Rate-Distortion Analysis for Adaptive Mode Selection and Rate Control in Wireless Video Coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 6, pp. 511–523, 2002.
- [165] J. C. Kwon and J.-K. Kim, “Adaptive Code Rate Decision of Joint Source-Channel Coding for Wireless Video,” *Electron. Lett.*, vol. 38, no. 25, pp. 1752–1754, 2002.
- [166] F. Liu, Z. Sun, Y. Yang, C. Guo, and S. Zhao, “Rate-Adaptable Multitask-Oriented Semantic Communication: An Extended Rate–Distortion Theory-Based Scheme,” *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15 557–15 570, 2024.
- [167] H. Wu, Y. Shao, C. Bian, K. Mikolajczyk, and D. Gündüz, “Vision Transformer for Adaptive Image Transmission Over MIMO Channels,” in *ICC 2023 IEEE Int. Conf. on Commun. (ICC)*, IEEE, 2023, pp. 3702–3707.
- [168] H. Yang et al., “An End-to-End Video Coding Method Via Adaptive Vision Transformer,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 38, no. 01, p. 2 354 023, 2024.
- [169] K. Ramchandran, M. Vetterli, and C. Herley, “Wavelets, Subband Coding, and Best Bases,” *Proceedings of the IEEE*, vol. 84, no. 4, pp. 541–560, 1996.
- [170] Z.-E. Fan, F. Lian, and J.-N. Quan, “Global Sensing and Measurements Reuse for Image Compressed Sensing,” in *Proc. IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, 2022, pp. 8954–8963.
- [171] D. Gündüz et al., “Beyond Transmitting Bits: Context, Semantics, and Task-Oriented Communications,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 5–41, 2022.

- [172] C.-H. Wu and W.-B. Liang, “Emotion Recognition of Affective Speech Based on Multiple Classifiers Using Acoustic-Prosodic Information and Semantic Labels,” *IEEE Trans. Affective Comput.*, vol. 2, no. 1, pp. 10–21, 2010.
- [173] Y. Tong, W. Liao, and Q. Ji, “Facial Action Unit Recognition by Exploiting Their Dynamic and Semantic Relationships,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1683–1699, 2007.
- [174] J. Vogel and B. Schiele, “Semantic Modeling of Natural Scenes for Content-Based Image Retrieval,” *Int. J. Comput. Vis.*, vol. 72, no. 2, pp. 133–157, 2007.
- [175] S. Dasiopoulou, V. Mezaris, I. Kompatsiaris, V.-K. Papastathis, and M. G. Strintzis, “Knowledge-Assisted Semantic Video Object Detection,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 10, pp. 1210–1224, 2005.
- [176] R. Wang, Y. Zhou, Y. Xiao, and Y. Li, “Multi-View Semantic-Aware Communication: An Information Bottleneck Perspective,” in *ICC 2025 IEEE Int. Conf. on Commun. (ICC)*, IEEE, 2025, pp. 4148–4153.
- [177] S. Zhang, X. Zheng, Y. Zhang, S. Li, and T. Jiang, “Bridging Theory and Practice: A Review of Task-Oriented Communication for Embodied Agent Societies,” *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 6, pp. 4480–4497, 2025.
- [178] A. Mostaani, T. X. Vu, H. Habibi, S. Chatzinotas, and B. Ottersten, “Task-Oriented Communication Design at Scale,” *IEEE Trans. Commun.*, vol. 73, no. 1, pp. 378–393, 2024.
- [179] D. Wen, Y. Zhou, X. Li, Y. Shi, K. Huang, and K. B. Letaief, “A Survey on Integrated Sensing, Communication, and Computation,” *IEEE Commun. Surveys Tuts.*, vol. 27, no. 5, pp. 3058–3098, 2024.
- [180] Y. Wang et al., “Feature Importance-Aware Task-Oriented Semantic Transmission and Optimization,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 10, no. 4, pp. 1175–1189, 2024.
- [181] Z. Sun, S. Ma, and S. Li, “Task-Oriented Semantic Communication With Importance-Aware Rate Control,” *IEEE Commun. Lett.*, vol. 29, no. 7, pp. 1520–1524, 2025.

- [182] C. Guo, J. Xi, Z. He, J. Liu, and J. Yang, “A Knowledge Base Driven Task-Oriented Image Semantic Communication Scheme,” *Remote Sensing*, vol. 16, no. 21, pp. 4044–4060, 2024.
- [183] A. Wijesinghe, S. Zhang, S. Wanninayaka, W. Wang, and Z. Ding, “Diff-Go+: An Efficient Diffusion Goal-Oriented Communication System With Local Feedback,” *IEEE Trans. Wireless Commun.*, vol. 24, no. 8, pp. 6535–6550, 2025.
- [184] Z. Weng, Z. Qin, X. Tao, C. Pan, G. Liu, and G. Y. Li, “Deep Learning Enabled Semantic Communications With Speech Recognition and Synthesis,” *IEEE Trans. Wireless Commun.*, vol. 22, no. 9, pp. 6227–6240, 2023.
- [185] S. Xie, S. Ma, M. Ding, Y. Shi, M. Tang, and Y. Wu, “Robust Information Bottleneck for Task-Oriented Communication With Digital Modulation,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 8, pp. 2577–2591, 2023.
- [186] H. Yang, Y. Wu, D. Wen, Y. Zhou, and Y. Shi, “Structured IB: Improving Information Bottleneck With Structured Feature Learning,” in *Proc. AAAI Conf. on Artif. Intell.*, vol. 39, 2025, pp. 21 922–21 928.
- [187] S. Lyu et al., “Improving Channel Resilience for Task-Oriented Semantic Communications: A Unified Information Bottleneck Approach,” *IEEE Commun. Lett.*, vol. 28, no. 11, pp. 2623–2627, 2024.
- [188] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in *Proc. IEEE/CVF Conf. on Comput. Vis. and Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520.
- [189] M. Yang and H.-S. Kim, “Deep Joint Source-Channel Coding for Wireless Image Transmission With Adaptive Rate Control,” in *ICASSP 2022 IEEE Int. Conf. on Acoust., Speech and Signal Process. (ICASSP)*, IEEE, 2022, pp. 5193–5197.
- [190] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “FiLM: Visual Reasoning With a General Conditioning Layer,” in *Proc. 32nd AAAI Conf. on Artif. Intell.*, New Orleans, Louisiana, USA, 2018.

- [191] M. Khan, S. Chakraborty, R. Astya, and S. Khepra, "Face Detection and Recognition Using OpenCV," in *Proc. IEEE ICCIS*, 2019, pp. 116–119.
- [192] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, 2016.
- [193] Z. Xu, W. Yang, A. Meng, N. Lu, and H. Huang, "Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 255–271.
- [194] G.-S. Hsu, J.-C. Chen, and Y.-C. Chung, "Application-Oriented License Plate Recognition," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 2, pp. 552–561, 2013. DOI: [10.1109/TVT.2012.2226218](https://doi.org/10.1109/TVT.2012.2226218).
- [195] Z. Yang, W. Xu, Z. Wang, X. He, F. Yang, and Z. Yin, "Combining YOLOv3-Tiny Model With DropBlock for Tiny-Face Detection," in *2019 IEEE 19th Int. Conf. on Communication Technology (ICCT)*, IEEE, 2019, pp. 1673–1677.
- [196] M. Jaderberg, K. Simonyan, A. Zisserman, et al., "Spatial Transformer Networks," *Adv. in Neu. Info. Proc. Sys.*, vol. 28, 2015.
- [197] S. Chen, Y. Liu, X. Gao, and Z. Han, "MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices," in *Chinese Conf. on Biometric Recognition*, Springer, 2018, pp. 428–438.
- [198] M. Abdul-Al, G. K. Kyeremeh, R. Qahwaji, N. T. Ali, and R. A. Abd-Alhameed, "The Evolution of Biometric Authentication: A Deep Dive Into Multi-Modal Facial Recognition: A Review Case Study," *IEEE Access*, 2024.
- [199] E. Jang, S. Gu, and B. Poole, "Categorical Reparameterization With Gumbel-Softmax," *arXiv preprint arXiv:1611.01144*, 2016.
- [200] X. Yi, H. Xu, H. Zhang, L. Tang, and J. Ma, "Diff-Retinex: Rethinking Low-Light Image Enhancement With a Generative Diffusion Model," in *Proc. IEEE/CVF Int. Conf. on Comput. Vis. (ICCV)*, 2023, pp. 12 302–12 311.