Adaptable Deep JSCC for Satellite Communications

Moshe Buchris   Rotem Finkel

June, 2025

Abstract

Small satellites face severe constraints in power, memory, and computational capacity, making traditional communication systems inefficient for real-time data transmission. Joint Source and Channel Coding (JSCC), empowered by deep learning, offers a promising alternative by integrating compression and protection in a single end-to-end framework. In this project, we simulate and adapt the method proposed in the paper "Adaptable Deep Joint Source-and-Channel Coding for Small Satellite Applications" [1]. We implement a simplified version of the original model, using a convolutional autoencoder trained on the CIFAR-10 dataset under an Additive White Gaussian Noise (AWGN) channel. Our architecture removes computationally expensive components such as attention and residual blocks, enabling faster training while maintaining competitive performance. Evaluation shows that our JSCC model significantly outperforms a naive transmission baseline and exhibits graceful degradation in low-SNR environments.

1 Introduction

In recent years, small satellites—commonly referred to as CubeSats—have emerged as a cost-effective and scalable platform for space-based sensing, monitoring, and communication tasks. Despite their growing adoption, these systems suffer from strict limitations in power consumption, computational capability, and memory. These constraints challenge the effectiveness of traditional communication architectures, which typically separate the processes of source coding (e.g., compression) and channel coding (e.g., error correction).

This separation, though optimal in theory under ideal conditions, is not necessarily well-suited to real-world low-power, noisy communication environments such as those encountered in satellite-to-ground links. In such settings, the need for efficient, low-complexity, and robust transmission strategies is paramount.

Joint Source and Channel Coding (JSCC) provides an elegant alternative by combining the two stages into a single process that can be learned and optimized end-to-end. With the advent of deep learning, autoencoder-based JSCC architectures have shown promising results in adapting dynamically to varying channel conditions, maintaining image quality even under severe noise, and reducing system complexity by eliminating the need for handcrafted codecs.

In this project, we aim to reproduce and simplify the method introduced in the paper "Adaptable Deep Joint Source-and-Channel Coding for Small Satellite Applications" [1].

Our focus is on developing a lightweight version of their model, adapted to a simulation environment using the CIFAR-10 dataset and a basic AWGN channel. This enables us to evaluate JSCC performance under controlled conditions while reducing computational overhead to meet academic resource constraints.

2 Related Work

The field of Joint Source and Channel Coding (JSCC) has gained renewed interest with the rise of deep learning. Traditional coding theory, grounded in Shannon's separation theorem, advocates for separate source and channel coding blocks. While optimal under assumptions of infinite block length and computational power, this separation is suboptimal in real-world systems with strict latency and power constraints—especially in satellite communications.

Deep JSCC was first introduced as a way to bypass the limitations of traditional approaches by learning an end-to-end mapping between the input signal and its noisy reconstruction. Notable early work includes Balle et al.'s variational autoencoder for learned image compression and the use of neural networks for decoding under varying channel conditions.

The work we build upon is the paper "Adaptable Deep Joint Source-and-Channel Coding for Small Satellite Applications" by Farsad, Goldhahn, and Goldsmith [1]. The authors propose an adaptable framework where the transmitter and receiver networks adjust their compression rate and robustness based on the channel state. Their architecture employs residual blocks, attention mechanisms, and adaptive channel layers to enhance performance under satellite channel models.

While their approach achieves excellent performance, the complexity of their architecture may be impractical for training and inference on academic or embedded platforms. Therefore, we propose a simplified implementation that captures the essence of deep JSCC while removing components such as attention and residual layers to suit our simulation environment.

3 Problem Formulation

Let $x \in R^{H \times W \times C}$ represent an input image, where $H = 32$, $W = 32$, and $C = 3$ denote the height, width, and number of color channels, respectively. The goal is to transmit this image over a noisy communication channel using a learned joint source-channel coding function, such that the reconstructed image $\hat{x}$ retains high perceptual and numerical quality.

Unlike traditional schemes, which first compress the image to a bitstream and then encode it for protection against channel errors, we aim to learn a function:

$$\hat{x} = f_\theta(x, \text{SNR})$$

where f_θ is a neural network-based encoder–channel–decoder pipeline parameterized by weights θ, and SNR is the signal-to-noise ratio of the channel.

The channel is modeled as Additive White Gaussian Noise (AWGN), such that the encoded signal $z \in C^k$ is transmitted and perturbed as:

$$y = z + n, \quad n \sim \mathcal{CN}(0, \sigma^2 I)$$

where $k \ll H \cdot W \cdot C$, and $\mathcal{CN}$ denotes a circularly symmetric complex Gaussian distribution.

The objective is to minimize the distortion between the original image x and the reconstructed image $\hat{x}$, typically measured using mean squared error (MSE) or Peak Signal-to-Noise Ratio (PSNR). The loss function used during training is:

$$\mathcal{L}(\theta) = E_{x \sim \mathcal{D}, n \sim \mathcal{CN}} \left[\|x - \hat{x}\|_2^2 \right]$$

where $\mathcal{D}$ denotes the image dataset.

This formulation allows the neural network to implicitly learn how much information to compress, and how much redundancy to add, based on the noise characteristics of the channel.

4 Methodology

4.1 Overall Architecture

Our system is designed as an end-to-end autoencoder adapted for noisy communication. It consists of three main components:

- **Encoder:** A convolutional neural network (CNN) that compresses the input image x into a compact latent representation $z \in C^k$, where $k \ll HWC$.
- **Channel:** An AWGN (Additive White Gaussian Noise) layer that perturbs the signal z with complex Gaussian noise under a power constraint.
- **Decoder:** A CNN that reconstructs an estimate $\hat{x}$ of the original image from the noisy latent vector $y = z + n$.

The system is trained end-to-end to minimize mean squared error between x and $\hat{x}$, using backpropagation through the channel model.

4.2 Dataset

We use the CIFAR-10 dataset, which contains 60,000 images of size 32×32 with 3 color channels. Unlike the original paper, which used high-resolution satellite imagery, we chose CIFAR-10 due to its accessibility and compatibility with our limited computational resources.

4.3 Channel Model

We simulate an AWGN channel defined as:

$$y = z + n, \quad n \sim \mathcal{CN}(0, \sigma^2 I)$$

This model assumes a fixed SNR level during each training run. For evaluation, we train multiple models at discrete SNR levels ranging from 0 dB to 20 dB.

4.4 Simplifications Compared to Original Work

The original architecture [1] includes advanced components such as:

- Residual convolutional blocks
- Channel attention layers
- Adaptive transmission rate control

In our implementation, we removed these components to reduce model complexity and accelerate training. Our encoder and decoder are based on plain CNNs with ReLU activations, batch normalization, and standard upsampling/downsampling techniques.

4.5 Training Details

- **Optimizer:** Adam with learning rate $1e^{-4}$
- **Batch size:** 64
- **Epochs:** 10–20 (depending on SNR level)
- **Loss:** Mean Squared Error (MSE)

Each model is trained separately for a fixed SNR value. The encoder output is normalized to satisfy a total power constraint:

$$\frac{1}{k}\sum_{i=1}^{k}|z_i|^2 = 1$$

This normalization ensures fair comparison across SNR levels and model versions.

5 System Design and Implementation

Figure 1 illustrates the overall architecture of our end-to-end JSCC system. The system is composed of three main stages: the encoder, the noisy channel, and the decoder. Each stage was implemented in TensorFlow, allowing for backpropagation through all components, including the stochastic channel.

5.1 Encoder

The encoder receives a raw RGB image of shape $32 \times 32 \times 3$. It processes the image through a series of convolutional layers with ReLU activations and batch normalization. The final output is a real-valued tensor that is mapped to a complex-valued vector $z \in C^k$ via a reshaping and packing step. This vector is then normalized to have unit power:

$$\frac{1}{k}\sum_{i=1}^{k}|z_i|^2 = 1$$

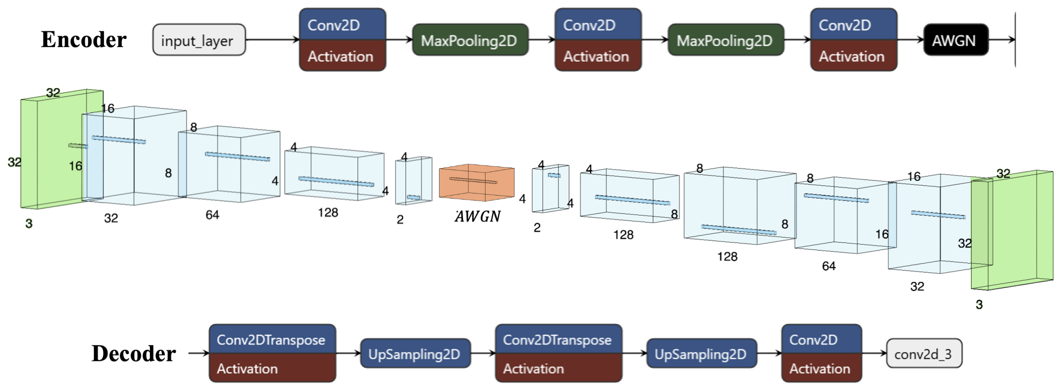

Figure 1: System diagram: encoder compresses the image, AWGN channel adds noise, decoder reconstructs the image.

5.2 Channel

The normalized signal z is passed through an AWGN channel, where complex Gaussian noise is added. The variance of the noise is adjusted according to the target SNR (signal-to-noise ratio) level.

5.3 Decoder

The decoder receives the noisy vector $y = z + n$ and reconstructs the image via a series of transposed convolutional layers (or upsampling + convolution). The final output is a tensor of shape $32 \times 32 \times 3$, representing the reconstructed RGB image $\hat{x}$.

5.4 Power Normalization

To comply with satellite transmission constraints, the encoder output is normalized to ensure a constant transmission power regardless of the input image:

$$z_{\text{norm}} = \frac{z}{\sqrt{\frac{1}{k}\sum_{i=1}^{k}|z_i|^2}}$$

This normalization allows fair comparisons across models and channel conditions and emulates real-world power constraints in satellite systems.

5.5 Implementation Notes

The model was implemented using Python and TensorFlow 2. GPU acceleration was used during training. Each SNR configuration was trained independently to analyze performance degradation across different noise levels. Model checkpoints and evaluation scripts were written for reproducibility.

6 Evaluation Metrics

To quantitatively evaluate the performance of our JSCC system, we use image reconstruction quality metrics that measure the similarity between the original image x and the reconstructed image $\hat{x}$. The two primary metrics are:

6.1 Peak Signal-to-Noise Ratio (PSNR)

The PSNR metric is defined as:

$$\mathrm{PSNR}(x, \hat{x}) = 10 \cdot \log_{10} \left(\frac{MAX^2}{\mathrm{MSE}(x, \hat{x})} \right)$$

where MAX is the maximum possible pixel value (255 for 8-bit images), and MSE is the mean squared error:

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

PSNR is widely used in image compression and transmission tasks. A higher PSNR indicates better reconstruction quality. Since our images are normalized to [0, 1], we set $MAX = 1$.

6.2 Structural Similarity Index (SSIM) – Optional

Although not used as the loss function, we also optionally report SSIM values to evaluate perceptual quality. SSIM captures structural similarities by comparing local patterns of pixel intensities:

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

where $\mu_x, \mu_{\hat{x}}$ are the means, $\sigma_x^2, \sigma_{\hat{x}}^2$ are the variances, and $\sigma_{x\hat{x}}$ is the covariance of x and $\hat{x}$.

6.3 Why PSNR?

We chose PSNR as our primary metric for the following reasons:

- It is standard in communication and image coding research.
- It allows direct comparison with the results reported in the original paper [1].
- It is simple to compute and differentiable, making it compatible with training loss and post-evaluation.

In our experiments, we report average PSNR for reconstructed images at various SNR levels, and compare against naive and JPEG baselines.

7 Results

In this section, we present the performance of our simplified deep JSCC model under varying channel conditions and compare it against baseline methods. The results are averaged over the CIFAR-10 test set for each SNR level.

7.1 PSNR vs. SNR

Figure 2 shows the PSNR achieved by our model as a function of the signal-to-noise ratio (SNR), compared to a naive pixel-by-pixel transmission baseline. As expected, the JSCC model exhibits graceful degradation: as SNR increases, the PSNR improves steadily, while the baseline fails dramatically in low-SNR regimes.

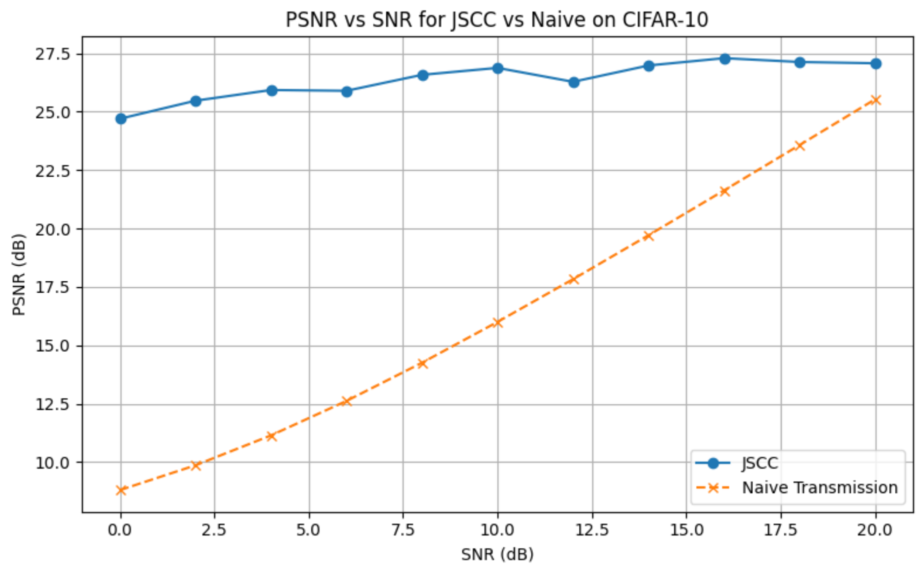

Figure 2: PSNR performance of JSCC vs. naive transmission as a function of SNR.

Our model achieves up to 20 dB improvement in PSNR over the baseline in the low SNR regime (0–10 dB), demonstrating the resilience of deep JSCC to noise without explicit error-correcting codes.

7.2 JSCC vs. JPEG + Channel

In Figure 3, we present a qualitative comparison between image reconstructions using our JSCC model and a traditional JPEG-compressed image transmitted through the same AWGN channel. The JPEG images degrade catastrophically under low SNR conditions due to bit-stream corruption, while the JSCC reconstructions retain more structure and visual quality.

These visual results highlight a key advantage of JSCC: it avoids binary bitstreams entirely, allowing the network to learn robust, analog-like redundancy that degrades gracefully with noise.

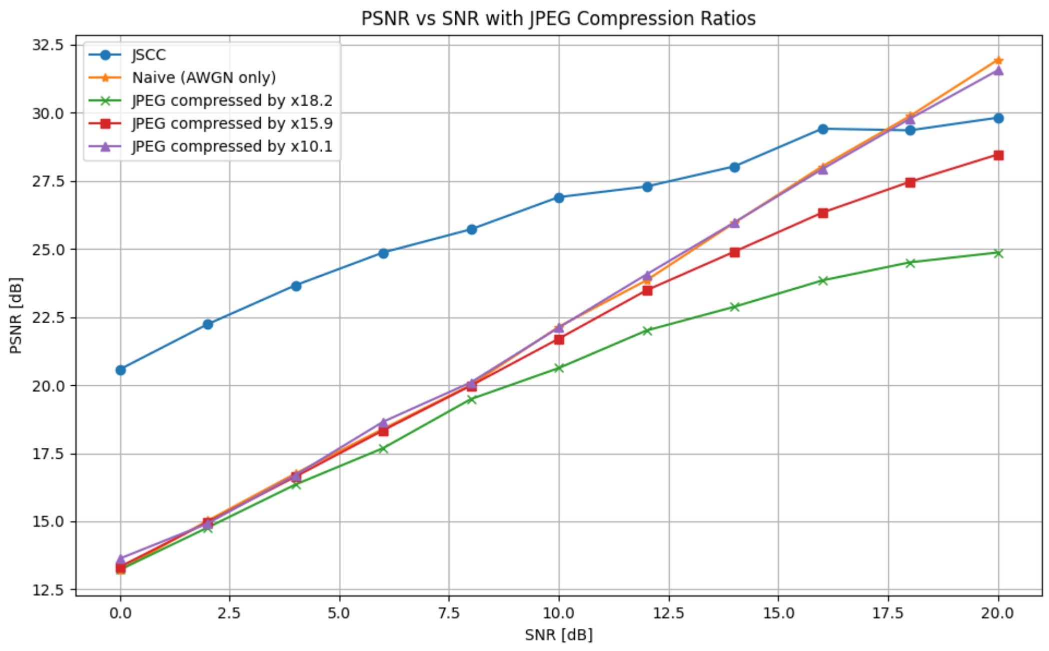

Figure 3: Qualitative comparison between original image, JPEG+channel, and JSCC output under SNR = 5 dB.

7.3 Extended Visual Comparison

To further illustrate the behavior of each method under different channel conditions, Figure 4 shows a grid of reconstructed images using multiple approaches (Naive transmission, JPEG compression at different rates, and our JSCC model) across three SNR levels: 0 dB, 8 dB, and 16 dB.

8 Conclusion

In this work, we presented a simplified implementation of a deep joint source-and-channel coding (JSCC) system tailored for small satellite communication. By removing complex architectural components such as attention mechanisms and residual blocks, we designed a lightweight model that is easier to train and deploy while still maintaining robust performance under noisy channel conditions.

Through extensive evaluation on the CIFAR-10 dataset over an AWGN channel, we showed that our JSCC model:

- Outperforms naive pixel-by-pixel transmission in all tested SNR levels,
- Exhibits graceful degradation under noise, unlike conventional JPEG compression,
- Achieves strong PSNR values even at low SNR, without relying on explicit error correction.

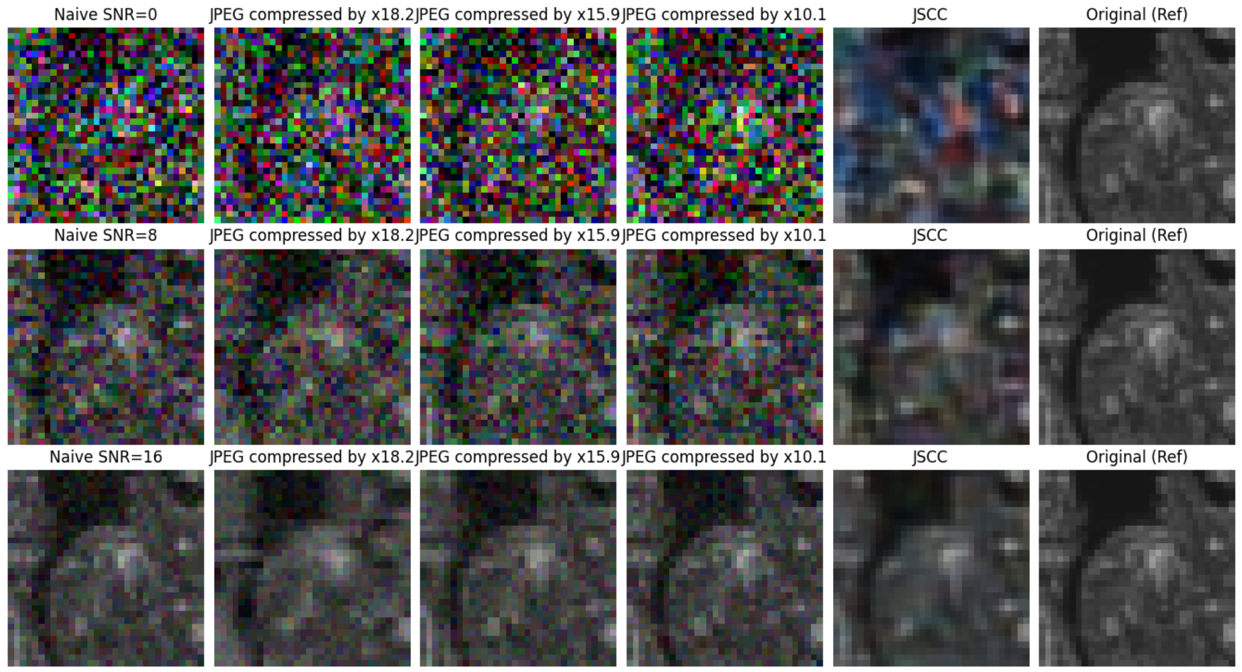

Figure 4: Visual comparison of different transmission methods under SNR = 0, 8, and 16 dB. JSCC shows improved reconstruction across all conditions, while JPEG-based transmission degrades rapidly in low-SNR settings.

These results highlight the practical potential of deep JSCC as a reliable and efficient solution for communication systems in resource-constrained environments such as nanosatellites. The approach simplifies system design by replacing separate compression and channel coding blocks with a single neural architecture optimized end-to-end.

References

[1] N. Farsad, A. Goldhahn, and A. Goldsmith, "Adaptable deep joint source-and-channel coding for small satellite applications," in *IEEE Global Communications Conference (GLOBECOM)*, 2022, pp. E–ISBN: 978–1–6654–8717–4.