DCS-JSCC: Leveraging Deep Compressed Sensing into JSCC for Wireless Image Transmission

Mohammad Amin Jarrahi
School of Computer Science and Electronic Engineering (CSEE)
University of Essex
Colchester, United Kingdom
m.jarrahi@essex.ac.uk

Eirina Bourtsoulatze
School of Computer Science and Electronic Engineering (CSEE)
University of Essex
Colchester, United Kingdom
e.bourtsoulatze@essex.ac.uk

Vahid Abolghasemi
School of Computer Science and Electronic Engineering (CSEE)
University of Essex
Colchester, United Kingdom
v.abolghasemi@essex.ac.uk

***Abstract*—This paper presents a novel approach that integrates deep compressed sensing (DCS) into joint source-channel coding (JSCC) for efficient image transmission. Leveraging the capabilities of DCS, the proposed method offers enhanced compression and resilience to channel noise in wireless image transmission systems. A key component of the method is the utilization of a convolutional neural network (CNN) structure to implement a block-based DCS technique for image compression. The proposed encoder utilizes a well-designed CNN-based structure to capture structural information and then map it to complex-valued signals. The proposed decoder deals with channel noise and reconstructs the original image. Using the CNN-based sampling matrices and reconstruction capabilities helps the proposed algorithm enhance image compression and reconstruction in wireless transmission systems. The CIFAR-10 and Kodak datasets are used to evaluate the performance of the suggested technique, showing a significant improvement in terms of Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), across different channel Signal-to-Noise Ratios (SNRs) and channel bandwidth values in comparison with state-of-art JSCC frameworks. Experimental evaluations demonstrate the effectiveness of the proposed method in achieving superior compression ratios and maintaining image quality under varying channel conditions.**

***Index Terms*—Wireless image transmission, joint source-channel coding, compressed sensing, deep learning**

I. Introduction

Wireless image transmission faces challenges in compression, transmission resilience, and quality preservation [1], [2]. Conventional methods employ separate source and channel coding, but there is a need for alternatives to improve performance in noisy and bandwidth-limited scenarios [3]. Joint source-channel coding (JSCC) integrates statistical image properties with channel characteristics to enhance compression efficiency and resilience to noise [4]. Incorporating compressed sensing (CS) into JSCC offers an opportunity to further enhance wireless image transmission [5], [6]. While CS has shown promising results in recovering sparse signals, its reliance on sparsity assumptions and expensive reconstruction process prompts the need for more efficient methods. Deep learning (DL) provides a solution by addressing these limitations and improving efficiency [7].

Various techniques and frameworks have been proposed for JSCC, offering innovative wireless image transmission solutions. DL-based methods introduce encoder and decoder network models that learn from input images, with the encoder output forming the transmitted code-word and the decoder aiming to reconstruct the source image from the received noisy code-word [8], [9]. Early contributions to DL-based JSCC [10], built the foundation for employing neural networks to address the challenges of encoding and decoding information in the presence of additive white Gaussian noise (AWGN). Inspired by advancements in variational learning and gradient estimation, Zhang et al. [11] proposed a JSCC model that leverages discrete latent variable models to enhance the efficiency and robustness of wireless communication systems. Building upon these foundations, Kurka et al. introduced a JSCC scheme offering unique advantages such as graceful degradation with varying signal-to-noise ratios (SNR) and utilisation of channel output feedback [12].

Applying CS to wireless image transmission offers practical advantages, evident in solutions like SoftCast and SparseCast [13], [14]. SoftCast uses a Discrete Cosine Transform (DCT) on images and transmits coefficients directly through a dense constellation, while SparseCast optimizes bandwidth with frequency domain sparsity [13], [14]. Additionally, Song et al. proposed a distributed CS for scalable cloud-based image transmission, improving reconstruction using cloud resources and enhancing resistance to channel impairments [15]. However, these methods may be sensitive to channel changes and are subject to potential disruptions, impacting image quality and transmission errors. Despite these challenges, the integration of CS and DL-based methods into JSCC frameworks holds promise for addressing the demands of wireless image transmission.

This paper presents a novel JSCC algorithm combining DL-based block-based CS (BCS) for improved compression rates and resilience to channel noise. Using a CNN-based structure, it integrates a BCS module into a DL-based source and channel encoder, capturing structural information and mapping it to complex-valued signals. At the receiver, a CNN-based decoder handles channel noise and reconstructs the original image. Leveraging DL-based sampling matrices and reconstruction capabilities, the proposed algorithm enhances image compression and reconstruction in wireless transmission systems. Numerical evaluations show that the proposed

scheme significantly outperforms existing DL-based JSCC methods such as Deep JSCC (DJSCC) [8] and Attention DL based JSCC (ADJSCC) [16] with respect to various metrics.

II. System Model

Let us consider a point-to-point image transmission system depicted in Fig. 1. Here, an input image of dimensions H (height) W (width) C (number of channels) is denoted by a vector $x \in \mathbb{R}^n$, where $n = H \quad W \quad C$, and $\mathbb{R}$ represents the set of real numbers. The joint source-channel encoder operates by encoding x through the function $f \ : \mathbb{R}^n \quad ! \ \mathbb{C}^k$, resulting in a vector of complex-valued channel input symbols $z \in \mathbb{C}^k$. This encoding process is mathematically expressed as:

$$z = f \ (x) \in \mathbb{C}^k \tag{1}$$

where k signifies the number of channel input symbols, represents the parameter set of the joint source-channel encoder, and $\mathbb{C}$ denotes the set of complex numbers.

To adhere to the average power constraint at the joint source-channel encoder, an additional condition $\frac{1}{k}E\,(zz^{\ }) \quad P$ is enforced, where z denotes the complex conjugate of z and P stands for the average power constraint. These encoded symbols z are transmitted across a noisy channel, which is modelled by the function $\ : \mathbb{C}^k \ ! \ \mathbb{C}^k$. Our work primarily considers the presence of additive white Gaussian noise (AWGN). Consequently, the channel output symbols $\hat{z} \in \mathbb{C}^k$ received by the joint source-channel decoder are described by:

$$\hat{z} = \ (z) = z + ! \tag{2}$$

where $! \in \mathbb{C}^k$ comprises independent and identically distributed $(i.i.d)$ samples with the distribution $\mathcal{CN}\ 0; \ ^2 I$. Here, ^{2} represents the average noise power, and $\mathcal{CN}(:;:)$ signifies a circularly symmetric complex Gaussian distribution. Moreover, our proposed method can be adapted to other channel models that can be represented by a differentiable transfer function. For decoding, the joint source-channel decoder utilizes a function $g_{\ '} : \mathbb{C}^k \ ! \ \mathbb{R}^n$ to map $\hat{z}$ to an estimation of the original image, given as:

$$\hat{x} = g_{\ '}(\hat{z}) = g_{\ '}(\ \ (f \ (x))) \tag{3}$$

where $\hat{x} \in \mathbb{R}^n$ is the reconstructed image at the receiver, and $'$ represents the parameter set of the joint source-channel decoder. In this study, we propose a novel approach to model f and $g_{\ '}$ through a DL-based structure, considering the advantages of CS, with further details described in the following section.

III. DCS-JSCC

Fig. 2 illustrates the end-to-end architecture of the encoder and decoder networks of the proposed DCS-JSCC. This model consists of a sampling network, integrated with an encoding network as the encoder, and a decoding network, integrated with the initial and deep reconstruction network as the decoder which are introduced next.

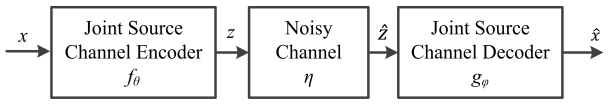

Fig. 1. Components of the system model [8]

A. Encoder Design

The encoder shown in Fig. 2 comprises a BCS sampling network, followed by an array of processing blocks, which encode the image by further compressing it and adding resilience to noise. The CS sampling method, outlined in [17], begins by partitioning the image into non-overlapping blocks of size $B \quad B \quad C$, where C represents the number of color channels and B denotes the block size. Compressed measurements are obtained using a sampling matrix $'_B$ with dimensions $n_B \quad CB^2$, or $sampling \quad ratio \quad B$ in scenarios where a sampling ratio like 0:1 is applied (see [17] for further details). The sampling process is mathematically expressed as $y_j = \ '_B x_j$, meaning that each row of jth measurement in $'_B$ 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 $B \quad B$. No biases or activation functions are used, resulting in n_B feature maps, each containing n_B measurements 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, the data flow proceeds through a series of convolutional layers, PReLU activation functions, and a Generalized Divisive Normalization (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 convolutional layers 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 signal space, enabling the network to capture complex relationships within the data. Also, GDN layer employs local divisive normalization, which is highly effective for tasks like image compression by capturing statistical dependencies within the image.

As the final step within the encoder, the output of the last convolutional layer is subjected to a normalization process as follows:

$$z = \ ^{p\overline{kP}\ p\frac{z^{*}}{z^{*}\ z}} \tag{4}$$

where z^{*} is the conjugate transpose of z, such that the channel input z satisfies the average transmit power constraint P.

After encoding, the joint source-channel coded sequence is transmitted over the communication channel by directly send-

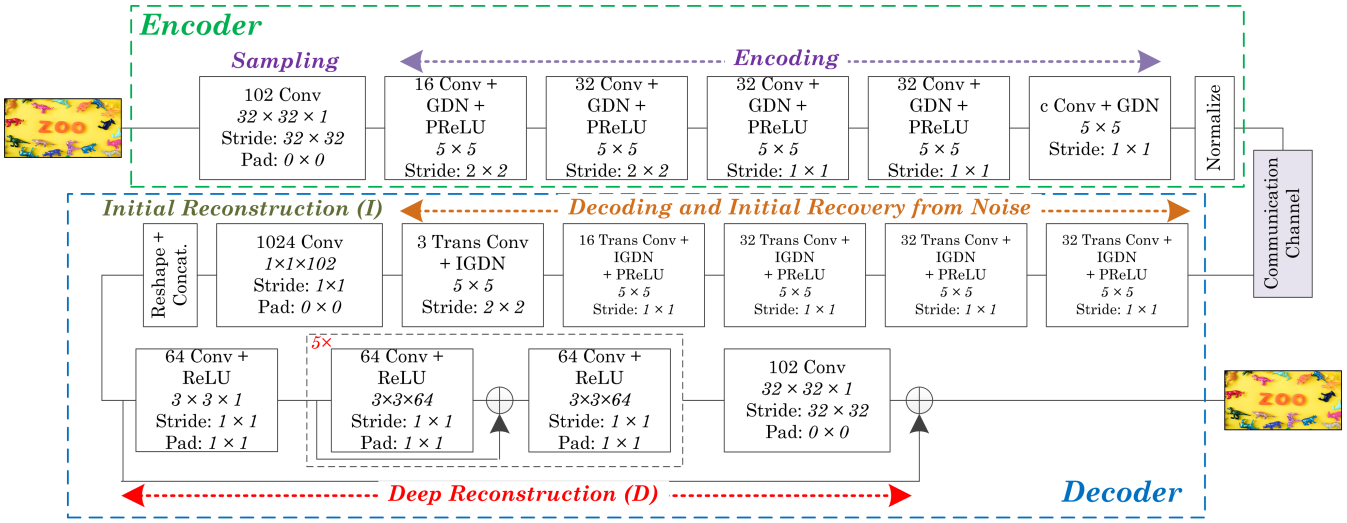

Fig. 2. Architecture of the proposed model

ing the real and imaginary parts of the channel input samples via the I and Q components of the digital signal. The channel introduces random corruption to the transmitted symbols. To optimize the end-to-end wireless image transmission system, the communication channel is included in the architecture as a non-trainable layer, represented by the transfer function in Eq. (2).

B. Decoder Design

As shown in Fig. 2, the designed decoder has three consecutive stages: decoding and initial recovery from noises, initial reconstruction and deep reconstruction. In the first stage, the decoder maps the signals 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 transpose convolutional layers with PReLU activation functions and Inverse GDN (IGDN).

The reconstruction network comprises initial I and deep networks D, ensuring accurate recovery of the original image from the CS encoded measurements. For the initial reconstruction stage, similar to the compressive sampling process, a convolutional layer is employed with a specific kernel size and stride. As can be seen in Fig. 2, IB^2 convolution filters of dimensions $1 \quad 1 \quad n_B$ are applied to generate each initial reconstructed block. 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 $1 \quad 1 \quad IB^2$ reconstructed vector into a $B \quad B \quad I$ block, followed by concatenating all blocks to form the initial reconstructed image. This initial phase allows considering reconstruction of the entire image rather than individual blocks, enabling comprehensive utilization of both intra-block and inter-block information for improved reconstruction. As there is no activation layer in the initial reconstruction network, it functions as a linear signal reconstruction network.

Following the initial reconstruction, a deep network based on residual learning is employed to enhance the non-linear signal reconstruction process for improved performance. This network encompasses three main operations: feature extraction, non-linear mapping, and feature aggregation. The feature extraction operation utilizes a convolutional layer followed by a ReLU activation layer to generate high-dimensional features from the local receptive field. Subsequently, the deep reconstruction network alternates between residual blocks, convolutional layers, and activation layers, augmenting the network's non-linearity and expanding its receptive field. To produce the final output, a feature aggregation operation is employed to reconstruct the image from the high-dimensional features. Additionally, a skip connection is incorporated between the initial reconstructed image and the output of the deep reconstruction network to expedite network convergence.

C. Loss Function

The proposed encoder and decoder networks are optimized jointly in an end-to-end manner. Given the input image x, the goal is to obtain z by using the encoding network f , and then recover the original input image x accurately from $\hat{x}$ by using the decoding network g . The mean square error is adopted as the cost function of the model. To do so, two objectives are considered to be minimized: the *initial* and *final* versions of the reconstructed image. For the initial reconstruction, the loss function would be:

$$L_{\text{int}}(\ ; \ ') = \frac{1}{K} \sum_{i=1}^{K} \| I (\ (f \ (x_i))) \quad x_i \|_2^2 \qquad (5)$$

where and are the parameters of the encoding and decoding networks to be trained, respectively. Also, K represents the number of samples or data points in the dataset. Moreover $I (\ (f \ (x_i))$ is the initial reconstructed output with respect to image x_i. For the final output $g (\ (f \ (x_i))$, the following function is considered:

$$L_{\text{deep}}(\ ;\ ') = \frac{1}{K}\sum_{i=1}^{K} kg \cdot (\ \ (f\ (x_i))) \quad x_i k_2^2 : \qquad (6)$$

Training is carried out by optimizing the mentioned functions, simultaneously using a summation between them considering the same weights. It should be noted that we train the encoder and decoder networks, jointly, but they can be utilized in the model, independently.

IV. Results and Discussions

The proposed model is implemented using TensorFlow and optimized utilizing the Adam algorithm. The compression ratio, denoted as $\frac{k}{n}$, where k represents the channel bandwidth and n signifies the source bandwidth, is varied across experiments, ranging from 0.05 to 0.45. Additionally, the channel signal-to-noise ratio (SNR), as formulated in [8], is adjusted during different trials. The performance of the algorithm is evaluated based on the peak signal-to-noise ratio (PSNR) of the reconstructed images. PSNR is calculated as the ratio of the peak signal power to the mean squared error between the original and reconstructed images, as mentioned in [8]. To train the proposed DCS-JSCC architecture, CIFAR-10 and Imagenet datasets are chosen [8], [18]. Then, the proposed model is tested on CIFAR-10 and Kodak datasets [19], and the results are compared with state-of-the-art DL-based JSCC methods, namely DJSCC [8] and ADJSCC [16].

A. Evaluation on CIFAR-10 Dataset

The training dataset consists of 60,000 images, each with dimensions of 32 32 3, alongside randomly generated realizations of the communication channel. To evaluate the effectiveness of our technique, it is tested on a separate set of 10,000 images from the CIFAR-10 dataset, distinct from those used in training. 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 observed on the test dataset. The experiments for this dataset consider values of $B = 8$ and $I = 3$. Also, The parameter c in the encoder's last CNN layer is responsible for defining the dimension of the channel input. It is noteworthy to mention that the test set images are not utilized for tuning network hyperparameters. To address the impact of channel-induced randomness, each image is transmitted 10 times during performance evaluation. The study assesses the performance of the proposed algorithm within an AWGN environment, adjusting the SNR to varying levels.

Figure 3 illustrates the PSNR of the reconstructed images plotted against the SNR of the channel while maintaining a fixed compression ratio of 1=6. Each curve in the plot represents the performance of the proposed end-to-end system trained at a specific channel SNR value, denoted as SNR_{train}. Subsequently, the learned encoder/decoder parameters are evaluated using test images under various SNR conditions, labeled as SNR_{test}. Essentially, each curve demonstrates how well the proposed approach performs when optimized for a particular channel SNR (SNR_{train}) and tested under different channel conditions (SNR_{test}). These results provide insights into the algorithm behavior in varying channel conditions, showcasing its resilience to changes in channel quality. The findings reveal that the proposed method consistently outperforms the trained DJSCC approach. Additionally, both the proposed method and ADJSCC demonstrate adaptability to changing SNR levels, as evidenced by their gradual performance degradation with decreasing SNR. Notably, the proposed method exhibits superiority over ADJSCC, demonstrating better performance as SNR_{test} increases, surpassing ADJSCC ($SNR_{\text{train}} = 13$dB) by up to 1.2dB.

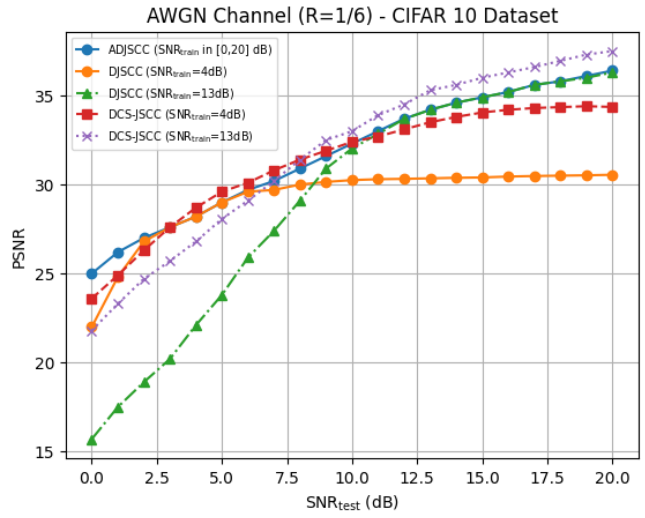

Fig. 3. CIFAR-10 dataset: performance of different methods with compression ratio 1=6, versus varying channel SNRs over an AWGN channel

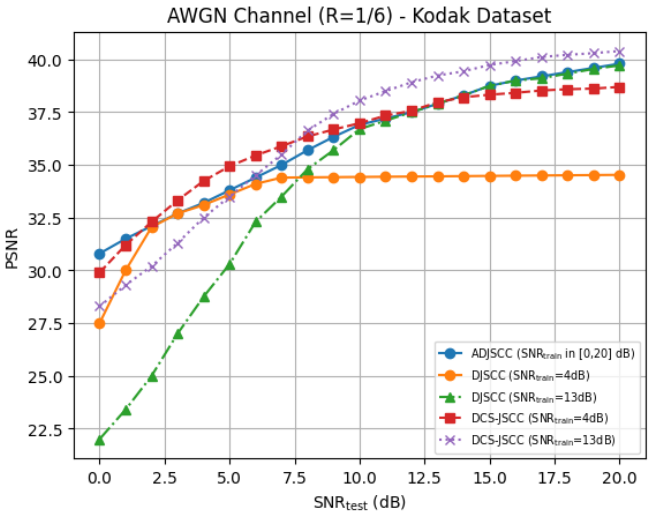

Fig. 4. Kodak dataset: performance of different methods with compression ratio 1=6, versus varying channel SNRs over an AWGN channel

B. Evaluation on Kodak Dataset

With almost 1.2 million images, the Imagenet dataset is a well-known dataset in our field of study. The images in our study are cropped to produce patches with dimensions