Rate-Adaptive Joint Source Channel Coding Using Deep Block-based Compressed Sensing

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 introduces a novel Rate-Adaptive Compressed Sensing-based Joint Source-Channel Coding scheme, termed RACS-JSCC, which leverages deep block-based CS to dynamically adjust the encoding rate based on available channel bandwidth and input image statistics. RACS-JSCC selects the encoding rate using both local and global statistics of the input image, alongside channel state information, prior to the feature extraction stage. This approach eliminates the transmission of redundant features and ensures the input image is encoded at an optimal rate. By training a deep learning-based JSCC encoder-decoder pair to operate across multiple rates and channel conditions, the proposed method reduces the necessity for multiple models and enhances practical applicability in diverse communication environments. Our experimental results demonstrate that RACS-JSCC achieves superior performance in terms of image quality and robustness against varying channel conditions, making it a highly efficient solution for real-world wireless image transmission.**

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

I. Introduction

Modern wireless image transmission systems typically rely on a separate design of source and channel coding. This approach is currently being utilized for real-world applications due to its modularity and, hence, ease of optimisation. However, the optimality of the separation-based source and channel coding can only be guaranteed in the case of infinitely, or, in practice, very long code blocks which becomes an important consideration in applications requiring low latency such as autonomous driving, remote surgery and other mission critical scenarios. In this regard, joint source and channel coding (JSCC) becomes a powerful alternative to separate source and channel coding as it can jointly leverage the statistical properties of the sources and the channel characteristics to achieve an optimal trade-off between source compression and error resilience for finite code block lengths [1].

Recent advancements in machine learning, and in particular, deep learning (DL), have revolutionized the research in the domain of JSCC design for wireless image transmission [2]. A number of works have demonstrated remarkable results showing that DL-based JSCC algorithms are resilient to channel variations and exhibit superior performance under low signal-to-noise ratios (SNR) and limited bandwidth conditions compared to their separation-based coding counterparts, making DL-based JSCC a promising candidate for efficient wireless image transmission systems [3], [4]. Bourtsoulatze *et al.* have proposed the first DL-based JSCC algorithm for wireless image transmission (DeepJSCC) [5] which leverages convolutional neural networks (CNNs) to directly map image pixel values to channel symbols, effectively integrating the processes of compression and transmission. This was later extended by Kurka *et al.* who proposed DeepJSCC-I [6] and DeepJSCC-f [7], which adapt the original DeepJSCC model to sequential and feedback-based transmission scenarios, respectively. These methods showed improved performance in low channel SNR and limited bandwidth conditions. Yang *et al.* [8] further advanced the field by integrating DeepJSCC with Orthogonal Frequency Division Multiplexing (OFDM) for multipath fading channels, enhancing robustness against signal degradation in complex transmission environments.

Despite their advantages, most of DL-based JSCC algorithms in the literature face several challenges that hinder practical adoption. The over-parameterized nature of CNNs deployed to realise the encoding and decoding functions results in high computational costs and significant storage requirements. This, combined with the fact that DL-based JSCC models are often trained for specific channel conditions and, thus, necessitate multiple models for different scenarios, renders them impractical for real-world applications. In this context, rate- and SNR-adaptive JSCC schemes have emerged as a key area of research [9]. These approaches aim to dynamically adjust the encoding rate based on available channel bandwidth and varying SNR conditions, thereby improving the adaptability and efficiency of DL-based JSCC algorithms. The authors in [10] have introduced attention mechanisms into the DL-based JSCC algorithm to achieve optimal encoding over a range of channel SNR values with a single model. Ding *et al.* [11] have introduced SNR-adaptive JSCC algorithm, which dynamically adjusts to different SNR levels without requiring multiple pre-trained models. The authors in [12], [13] achieve rate adaptivity by using a policy network which selects the most important features to be transmitted over the channel based on feature characteristics. However, the challenge of reducing the model size and computational complexity without sacrificing performance persists. Techniques such as low-rank decomposition, particularly Tensor-Train (TT) decomposition [14], [15], explored to mitigate these issues by compressing the neural network parameters, thus making the models more suitable for deployment on resource-constrained devices.

In our previous work [16], we have demonstrated that the integration of compressed sensing (CS) [17] into JSCC algorithm can significantly improve the performance. Specif-

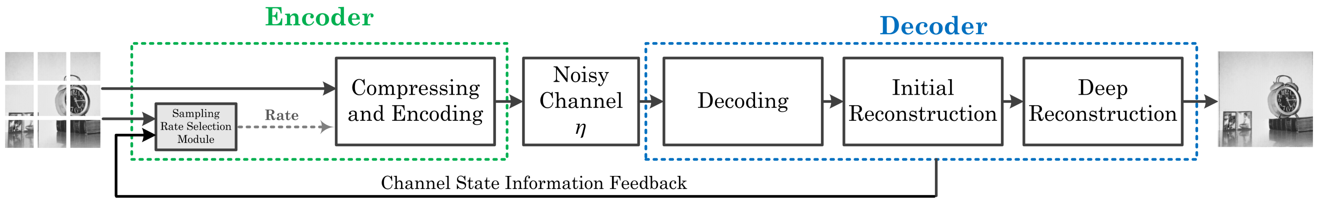

Fig. 1. Key components of the system model.

ically, we have introduced an CS-based sampling module, which leverages the sparsity of natural images and reduces the amount of input information prior to feature extraction and encoding by a CNN-based encoder. The addition of the CS module facilitates subsequent feature extraction modules to learn and encode features with high semantic content.

In this paper, we capitalise on the success of our CS-based JSCC algorithm and propose a novel rate-adaptive JSCC scheme, termed RACS-JSCC, that employs a deep block-based CS technique, designed to dynamically adjust the encoding rate, based on the available channel bandwidth and the statistics of the input images. Unlike existing works, the encoding rate is selected using both local/global statistics of the input signal and the channel state information prior to feature extraction. This eliminates the learning of redundant features that are never transmitted and enables the encoding of the input signal at an optimal rate. By training a DL-based JSCC encoder-decoder pair to handle multiple rates and channel conditions, our method not only reduces the need for multiple models but also enhances practical applicability in diverse communication environments.

II. System Model

Consider a point-to-point image transmission system illustrated in Fig. 1. The encoder encodes an input image of dimensions $H \times W \times C$, where H is the height, W is the width, and C is the number of channels, represented as a vector $\mathbf{x} \in \mathbb{R}^n$ with $n = H \times W \times C$. The encoded image is then transmitted over a wireless communication channel and reconstructed at the decoder.

The joint source-channel encoder comprises two parts: a sampling rate selection network and an encoding network. The optimal sampling rate is chosen based on the channel state information (CSI), which is estimated at the decoder and fed back in the form of the channel SNR, and the input image statistics. Subsequently, the encoder, represented by the encoding function $f_\theta : \mathbb{R}^n \to \mathbb{C}^k$, maps the input image $\mathbf{x}$ to complex-valued channel input symbols $\mathbf{z} \in \mathbb{C}^k$. Mathematically, this operation is:

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

where k denotes the number of encoded symbols, θ denotes the encoder parameters, and $\mathbb{C}$ is the set of complex numbers.

The wireless communication channel introduces noise and distortion to the encoded symbols $\mathbf{z}$. In this work, we consider an additive white Gaussian noise (AWGN) channel model, where the channel function $\eta : \mathbb{C}^k \to \mathbb{C}^k$ is represented as:

$$\hat{\mathbf{z}} = \eta(\mathbf{z}) = \mathbf{z} + \mathbf{w}. \tag{2}$$

Here, $\mathbf{w} \in \mathbb{C}^k$ represents i.i.d samples with a circularly symmetric complex Gaussian distribution $\mathcal{CN}(0, \sigma^2 I)$, where σ^2 denotes the noise power.

The noisy symbols are decoded at the decoder which comprises three networks: the decoding network, the initial reconstruction network, and the deep reconstruction network. The decoding function g_φ maps the received symbols $\hat{\mathbf{z}}$ back to an intermediate compressed representation $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}} = g_\varphi(\hat{\mathbf{z}}) = g_\varphi(\eta(f_\theta(\mathbf{x}))) \tag{3}$$

The initial reconstruction network produces an initial estimate of the transmitted image from $\tilde{\mathbf{x}}$:

$$\hat{\mathbf{x}}_{\text{init}} = h_\psi(\tilde{\mathbf{x}}) \tag{4}$$

where h_ψ represents the initial reconstruction function parameterized by ψ. Finally, the deep reconstruction step employs a deep learning-based approach to enhance $\hat{\mathbf{x}}_{\text{init}}$ further, leveraging the advantages of compressed sensing and deep learning techniques. This step aims to refine the initial estimate $\hat{\mathbf{x}}_{\text{init}}$ into the final reconstructed image:

$$\hat{\mathbf{x}} = d_\chi(\hat{\mathbf{x}}_{\text{init}}) \tag{5}$$

where d_χ is the deep reconstruction function parameterized by χ.

III. Proposed RACS-JSCC

Our proposed RACS-JSCC framework (Fig. 2) introduces a strategy to optimize the image transmission process by dynamically adjusting the sampling rate based on image content and transmission channel conditions. Our adaptive approach ensures efficient use of bandwidth while maintaining high image quality.

A. Encoder Design

The overall architecture of the RACS-JSCC encoder is shown in Fig. 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 inputted 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 BCS 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

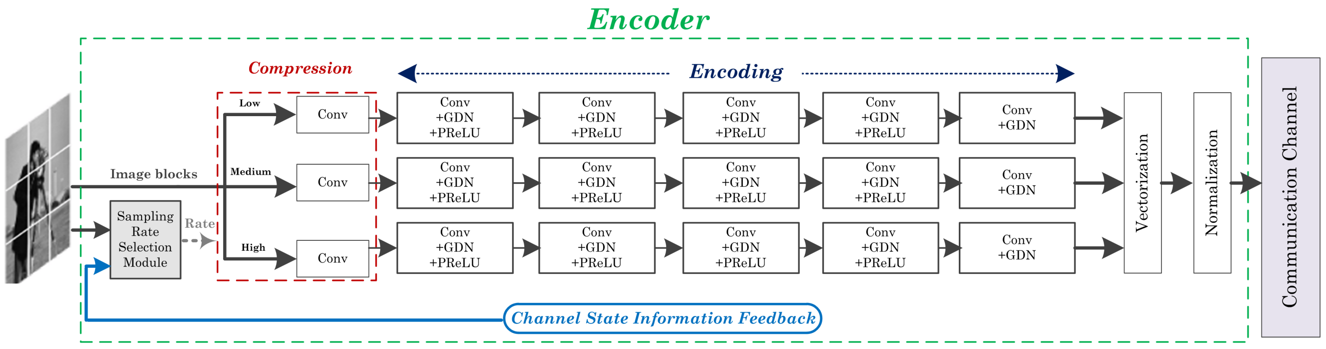

Fig. 2. Architecture of the proposed encoder.

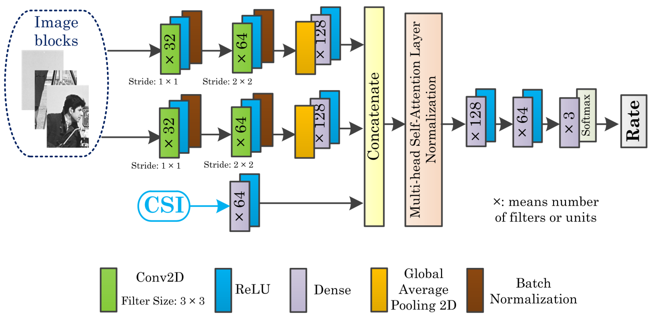
Image blocks
Stride: 1 × 1
Stride: 2 × 2
× 32
× 64
× 128
CSI
Concatenate
Multi-head Self-Attention Layer Normalization
× 3
Softmax
Rate
×: means number of filters or units
Conv2D Filter Size: 3 × 3
ReLU
Dense
Global Average Pooling 2D
Batch Normalization

Fig. 3. Architecture of the sampling rate selection module.

input image such as spatial frequency content, overall contrast, edges and local texture variations. This is then followed by a combination of fully connected layers and attention mechanisms which combine the image features and the channel state information to determine the optimal sampling rate for each block. Specifically, the sampling rate selection process involves the following key steps:

- *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.
- *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. 3.

The rate selection process involves directing each block to one of three sampling modules with high, medium and low sampling rate (Fig. 2). The high-rate module samples input blocks at 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.

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 BCS sampling network, outlined in [18], generates compressed measurements using a sampling matrix φ_B with dimensions $n_B \times CB^2$. The sampling process of the j-th block can be mathematically expressed as $\mathbf{y}^j = \varphi_B \mathbf{x}^j$, 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 n_B feature maps, with each column of output encapsulating n_B 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 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 convolution layers, shown in Fig. 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

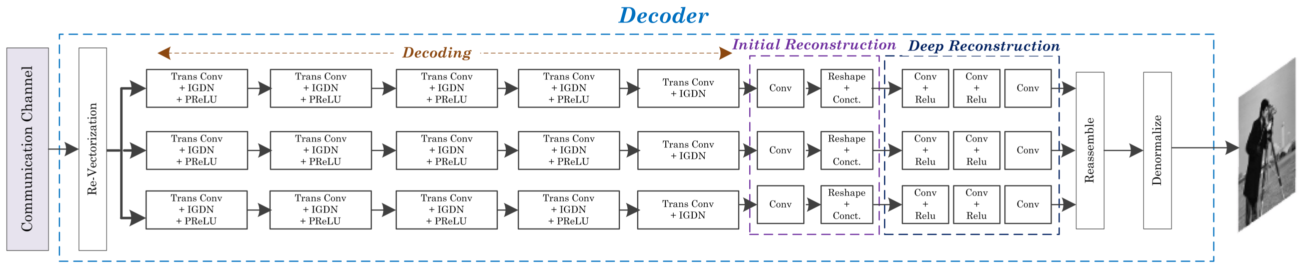

Fig. 4. Architecture of the proposed decoder

within the image. The details of the implementation of these layers can be found in [16].

After the image blocks are compressed and encoded, 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 a final step, the output of the encoder, which is a vector comprised of compressed encoded image blocks, is subjected to a normalization process as follows:

$$\mathbf{z} = \sqrt{kP}\frac{\tilde{\mathbf{z}}}{\sqrt{\tilde{\mathbf{z}}^{*}\tilde{\mathbf{z}}}} \tag{6}$$

where $\tilde{\mathbf{z}}$ is the vector of concatenated encoded blocks and $\tilde{\mathbf{z}}^{*}$ is the conjugate transpose of $\tilde{\mathbf{z}}$. The above normalization ensures that the channel input $\mathbf{z}$ satisfies the average transmit power constraint P. After encoding, the joint source-channel coded sequence is transmitted over the communication channel by directly sending the real and imaginary parts of the channel input samples via the I and Q components of the transmitted 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. 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 Parametric ReLU (PReLU) activation functions and Inverse Generalized Divisive Normalization (IGDN). The details of these layers can be found in [16]. 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 $1 \times 1 \times n_B$ and stride 1×1 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 $1 \times 1 \times CB^2$ 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 considering 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 subnetwork, 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 $f \times f \times 1$. The last layer, which outputs the final image estimation, 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.

C. 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

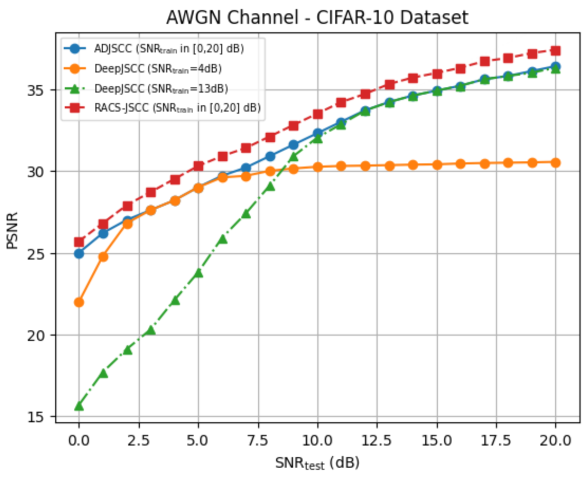

Fig. 5. CIFAR-10 dataset: performance of different methods with compression ratio $k/n = 1/6$, versus varying channel SNRs over an AWGN channel

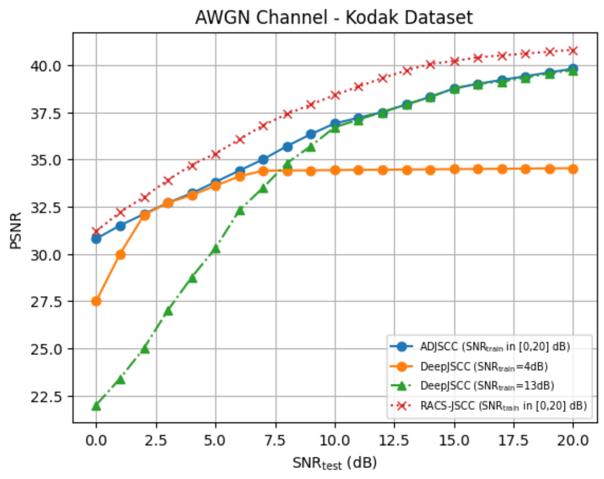

Fig. 6. Kodak dataset: performance of different methods with compression ratio $k/n = 1/6$, versus varying channel SNRs over an AWGN channel

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. Following most DL-based methods, the mean square error is adopted as the cost function of the proposed network. The optimization objective is formulated as:

$$\min_{\theta,\varphi,\psi,\chi} \frac{1}{N} \sum_{i=1}^{N} \| d_\chi \left(h_\psi \left(g_\varphi \left(\eta \left(f_\theta(\mathbf{x}_i) \right) \right) \right) \right) - \mathbf{x}_i \|_2^2 \qquad (7)$$

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 $d_\chi \left(h_\psi \left(g_\varphi \left(\eta \left(f_\theta(\mathbf{x}_i) \right) \right) \right) \right)$ is the final reconstructed output $\hat{\mathbf{x}}$. It should be noted that we train the encoder network and the decoder network jointly, but they can be utilized independently.

IV. Results

The proposed model is implemented using TensorFlow and optimized by the Adam algorithm. The performance of the RACS-JSCC 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. 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 [19], and the results are compared with state-of-the-art DL-based JSCC methods, namely DeepJSCC [5] and ADJSCC [10]. To achieve robustness to varying channel conditions, our 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 DeepJSCC 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, our proposed RACS-JSCC algorithm is rate-adaptive. Hence, for comparison purposes, we allow our algorithm to generate at most k channel input symbols, meaning that in practice due to the adaptive BCS sampling, the number of actual channel input symbols may be less than k.

A. Evaluation on CIFAR-10 Dataset

The training dataset consists of $60,000$ CIFAR-10 images, each with dimensions of $32 \times 32 \times 3$, alongside randomly generated realizations of the communication channel. To evaluate the performance of our algorithm, we test on a separate set of $10,000$ images from the CIFAR-10 dataset, distinct from those used for 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 of the loss function observed on the test dataset. In the experiments for this dataset we set $B = 8$. 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. The study assesses the performance of the proposed algorithm for an AWGN channel by varying the channel SNR.

Fig. 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$_{\text{test}}$). We can see that our algorithm is robust to changes in channel quality and can operate under varying channel conditions. Furthermore, the results clearly reveal that the proposed method consistently outperforms the DeepJSCC 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.

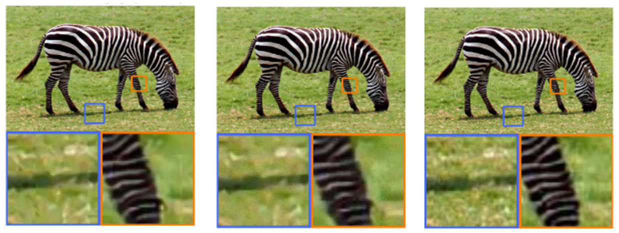
DeepJSCC / 23.42 dB / 0.7821
ADJSCC / 26.34 dB / 0.8412
RACS-JSCC / 28.81 dB / 0.895

Fig. 7. Reconstruction quality of an exemplary image (PSNR/SSIM) with different methods under the compression ratio $k/n = 1/6$ and SNR=$4dB$

However, RACS-JSCC surpasses ADJSCC, demonstrating better performance as SNR$_{\text{test}}$ 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.

B. Evaluation on Kodak Dataset

To demonstrate the validity of our RACS-JSCC algorithm, we further train our model 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, we set $B = 32$. The training process is carried out until convergence is attained, with the model learning rate fixed at 10^{-4}. We use SNR$_{\text{train}}$ values ranging from $0dB$ to $20dB$ 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. As previously, we consider transmission over an AWGN channel.

Fig. 6 illustrates the comparison of average PSNR against SNR for a compression ratio of $1/6$ with DeepJSCC and ADJSCC. The performance depicted in Fig. 6 reveals that our approach surpasses DeepJSCC 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.

In Fig. 7, a visual comparison of the reconstructed images is presented, showcasing the performance of RACS-JSCC trained on Imagenet in AWGN channels, contrasted with DeepJSCC and ADJSCC. For each reconstruction, both PSNR and Structural Similarity Index Measure (SSIM) values were calculated. The results indicate that the proposed method demonstrates superior visual reconstruction capabilities by accurately restoring the details of the original image while achieving a high compression ratio.

V. Conclusion

In this paper, we introduced RACS-JSCC, a rate-adaptive JSCC scheme equipped with deep block-based CS to dynamically adjust encoding rates based on channel bandwidth and image statistics. This innovative approach optimizes rate selection by utilizing local and global image statistics along with channel state information, thereby eliminating the learning of redundant features. By training deep learning-based JSCC encoder-decoder pairs to manage various rates and channel conditions, RACS-JSCC reduces the need for multiple models and enhances its applicability in diverse communication environments. Our experimental results demonstrate superior image quality and robustness, confirming the effectiveness of RACS-JSCC for adaptive and efficient wireless image transmission.

References

[1] J. Xu, B. Ai, W. Chen, N. Wang, and M. Rodrigues, "Deep joint source-channel coding for image transmission with visual protection," *IEEE Trans. on Cognitive Commun. and Netw.*, vol. 9, no. 6, pp. 1399–1411, 2023.

[2] N. Thomos, T. Maugey, and L. Toni, "Machine learning for multimedia communications," *Sensors*, vol. 22, no. 3, 2022.

[3] L. Sun, Y. Yang, M. Chen, C. Guo, W. Saad, and H. V. Poor, "Adaptive information bottleneck guided joint source and channel coding for image transmission," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 8, pp. 2628–2644, 2023.

[4] 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.

[5] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Trans. Cognitive Commun. and Netw.*, vol. 5, no. 3, pp. 567–579, 2019.

[6] 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.

[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] M. Yang, C. Bian, and H.-S. Kim, "OFDM-guided deep joint source channel coding for wireless multipath fading channels," *IEEE Trans. Cognitive Commun. and Netw.*, vol. 8, no. 2, pp. 584–599, 2022.

[9] S. F. Yilmaz, C. Karamanlı, and D. Gündüz, "Distributed deep joint source-channel coding over a multiple access channel," in *Proc of. IEEE Int. Conf. Commun.*, 2023, pp. 1400–1405.

[10] 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.

[11] M. Ding, J. Li, M. Ma, and X. Fan, "SNR-adaptive deep joint source-channel coding for wireless image transmission," in *Proc. of IEEE Int. Conf. Acoust., Speech and Signal Process.*, 2021, pp. 1555–1559.

[12] W. Chen, Y. Chen, Q. Yang, C. Huang, Q. Wang, and Z. Zhang, "Deep joint source-channel coding for wireless image transmission with entropy-aware adaptive rate control," in *Proc. of IEEE GLOBECOM*, 2023, pp. 2239–2244.

[13] M. Yang and H.-S. Kim, "Deep joint source-channel coding for wireless image transmission with adaptive rate control," in *Proc. of IEEE ICASSP*, 2022, pp. 5193–5197.

[14] 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 *Proc. of IEEE Int. Conf. Signal and Image Process.*, 2022, pp. 707–712.

[15] 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 *Proc. of IEEE Int. Conf. Computer and Commun.*, 2022, pp. 58–64.

[16] M. A. Jarrahi, E. Bourtsoulatze, and V. Abolghasemi, "Joint source-channel coding for wireless image transmission: A deep compressed-sensing based method," in *Proc of IEEE Wireless Commun. and Netw. Conf.*, 2024.

[17] D. Donoho, "Compressed sensing," *IEEE Trans. on Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[18] 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.

[19] Kodak, "Kodak color management," https://r0k.us/graphics/kodak/, accessed: January 8, 2013.