Bandwidth-Agile Image Transmission with Deep Joint Source-Channel Coding

David Burth Kurka and Deniz Gündüz
Information Processing and Communications Laboratory
Department of Electrical and Electronic Engineering
Imperial College London, London, UK
{d.kurka, d.gunduz}@imperial.ac.uk

Abstract

We propose deep learning based communication methods for adaptive-bandwidth transmission of images over wireless channels. We consider the scenario in which images are transmitted progressively in layers over time or frequency, and such layers can be aggregated by receivers in order to increase the quality of their reconstructions. We investigate two scenarios, one in which the layers are sent sequentially, and incrementally contribute to the refinement of a reconstruction, and another in which the layers are independent and can be retrieved in any order. Those scenarios correspond to the well known problems of *successive refinement* and *multiple descriptions*, respectively, in the context of joint source-channel coding (JSCC). We propose DeepJSCC-l, an innovative solution that uses convolutional autoencoders, and present three architectures with different complexity trade-offs. To the best of our knowledge, this is the first practical multiple-description JSCC scheme developed and tested for practical information sources and channels. Numerical results show that DeepJSCC-l can learn to transmit the source progressively with negligible losses in the end-to-end performance compared with a single transmission. Moreover, DeepJSCC-l has comparable performance with state of the art digital progressive transmission schemes in the challenging low signal-to-noise ratio (SNR) and small bandwidth regimes, with the additional advantage of graceful degradation with channel SNR.

Index Terms

Image transmission, joint source-channel coding, multiple description coding, successive refinement, wireless communication.

I. Introduction

We consider wireless transmission of images in multiple layers, each communicated over an independent noisy channel. The receiver receives the output of only a subset of the channels, and tries to reconstruct the original image at the best quality possible. We would like the image quality to increase as more layers are received. Such a scheme enables flexible transmission modes, where communication can be fulfilled with varying bandwidth availability. For example, these layers may be communicated over different frequency bands, and the receiver may be able to tune into only a subset of these bands. We would like the receiver to be able to reconstruct the underlying image no matter which subset of bands it can tune into. Alternatively, if the layers are transmitted sequentially in time, the receiver can stop receiving if it has reached a desired reconstruction quality, saving valuable time and energy resources. Concurrently, other receivers may continue to receive more layers, and can recover a better quality reconstruction by receiving additional symbols. Such a scheme results in bandwidth agile communication, and can be used in a variety of applications in which communication is either expensive, urgent, or limited. For example, in surveillance applications, it may be beneficial to quickly send a low-resolution image to detect a potential threat as soon as possible, while a higher resolution description can be received with additional delay for further evaluation or archival purposes. This approach can also benefit emergency systems, where urgent actions may need to be taken based on low resolution signals transmitted rapidly.

Our problem is the joint source-channel coding (JSCC) version of the well-known *multiple descriptions* problem [1]. The conventional multiple description problem focuses on the compression aspects, where the image is compressed into multiple layers, each at a different rate. In the multiple description problem, each layer is either received perfectly or not received at all, and the goal is to obtain the best possible reconstruction quality for any subset of received layers. A special case of this problem is the *successive refinement* problem, in which the layers are transmitted sequentially, starting from a base layer providing the main elements of the content being transmitted, followed by refinement layers used to enhance the image quality and add details to it. See Fig. 1 for an illustration of the two problems.

The rate-distortion region for both the multiple description [2]–[4] and the successive refinement problems [5]–[8] have been studied extensively from an information theoretic

perspective. While the optimal rate-distortion region for the multiple description problem remains open for general source distributions, optimal characterization is known for Gaussian sources [9]. A general single-letter characterization of the rate-distortion region is possible for the successive refinement problem [7]. Generating practical multiple description and successive refinement codes has also been studied. While the best practical source codes typically depend on the statistical properties of the underlying source distribution, researchers have studied how to achieve successive refinement or multiple descriptions through quantization [10]–[13]. Multiple descriptions can also be obtained through a pair of correlating transforms [14].

The JSCC version of the problem, however, has received considerably less attention. This may be partially due to the theoretical optimality of separation between the source and channel coding problems. A separation theorem is proven in [15] for the successive refinement JSCC problem when the layers are transmitted over independent channels. It is shown that it is optimal to compress the source into multiple layers using successive refinement source coding, where the rate of each layer is dictated by the capacity of the channel it is transmitted over. A similar result is proven for the multiple description problem for Gaussian sources in [16]. Note, however, that the optimality of separation holds only under the information theoretic assumption of ergodic sources and channels, and in the asymptotic limits of large source and channel blocklengths and unbounded complexity.

Here, following our previous work [17]–[19], we use deep learning (DL) methods, in particular, the autoencoder architecture [20], for the design of a practical end-to-end multiple description image transmission system. In [17], we introduced a novel end-to-end DL-based JSCC scheme for image transmission over wireless channels, called *DeepJSCC*, where encoding and decoding functions are parameterized by convolutional neural networks (CNNs) and the communication channel is incorporated into the neural network (NN) architecture as a non-trainable layer. This method achieves remarkable performance in low signal-to-noise ratio (SNR) and limited channel bandwidth, also showing resilience to mismatch between training and test channel conditions and channel variations, similarly to analog communications; that is, it does not suffer from the 'cliff effect' unlike digital communication schemes based on separate source and channel coding. JSCC of text has also been studied in [21]. Several recent works have considered variational autoencoders for JSCC [22]–[25]. Similarly to [24] and [25], Gaussian sources are considered in [26], but

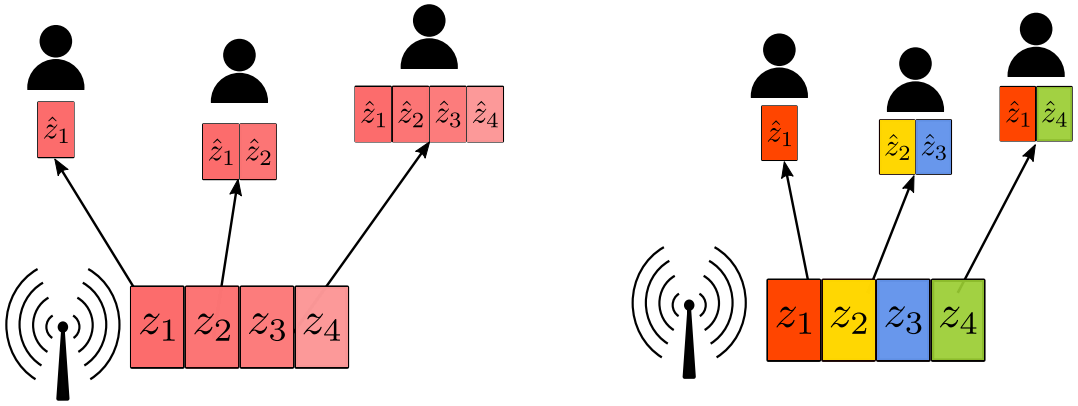


Fig. 1. Bandwidth-agile JSCC illustrating successive refinement (left) and multiple descriptions (right). Given an input image, the transmitter generates multiple codewords, denoted by $\boldsymbol{z}_i$, $i \in 1, \ldots, L$ (in the illustration, $L = 4$), and the receiver receives noisy versions of a subset of the codewords. Under the successive refinement scheme, layers are received sequentially, i.e., $\hat{\boldsymbol{z}}_1, \ldots, \hat{\boldsymbol{z}}_i$ for $i \leq L$. Under the more general multiple description scenario, any subset of noisy codewords may be received.

LSTM based autoencoder architecture is employed instead. Extension of [17] to a network of orthogonal links is considered in [27] with the focus on 'network coding' carried out by the intermediate nodes. Techniques and ideas from DeepJSCC have also been exploited for channel state information feedback [28], classification at the network edge [29], or wireless image retrieval [30].

In parallel, there have been significant efforts in the design of DL-based image compression schemes, in some cases outperforming current handcrafted codecs [31]–[35]. More recently, these efforts have also been extended to the multiple description problem [36]–[38]. In the source coding domain, an autoencoder is used for dimensionality reduction to efficiently represent the original source image. This is followed by quantization and entropy coding as in standard compression codecs. However, in the JSCC problem, a low dimensional representation of the source is not sufficient. The encoder must learn how to map the input to the transmitted channel input vectors. In principle, this transformation should map similar source signals to similar channel inputs, so that they can be reconstructed with minimal distortion despite channel noise.

We first tackle the successive refinement problem, and introduce a new strategy for progressive image transmission, called *DeepJSCC-l*. We show with extensive experimental

results that DeepJSCC-l can successfully learn to encode images into multiple channel codewords, each one successively refining the reconstruction at the receiver, and that the introduction of multiple codewords does not cause significant performance losses. In the context of source coding, a source is said to be "successively refinable" under a specified distortion measure when it is possible to achieve the single layer rate distortion performance at every stage of the successive refinement process. For example, Gaussian sources are successively refinable under squared-error distortion. Here, in the context of JSCC, our experimental results suggest that natural images transmitted with DeepJSCC are nearly 'successively refinable' over Gaussian channels. We also demonstrate how the problem of successive refinement can be approached with different implementations, by proposing three candidate solutions with different time-space complexity trade-offs. Finally, we further extend the solution and explore the more general multiple description problem, showing that our solution is able to learn independent codewords that have similar performance to a single layer transmission when sent separately, yet significantly improving the transmission performance when multiple parts are combined.

Despite the introduction of progressive transmission through successive refinement, all the properties present in single-layer transmission with DeepJSCC [17], such as graceful degradation, versatility in different channel models, and better or comparable performance compared to separate source and channel coding (JPEG2000 or BPG followed by high performance channel codes) are maintained. Thus, this work introduces, to the best of our knowledge, not only the first practical progressive and multiple-decription JSCC schemes for realistic information sources and channels, but also a solution that enables flexible and high-performance communication with adaptive bandwidth and uncertain channel quality; providing one more reason to explore its practical implementation in future communication systems.

In summary, the main contributions of this work are:

- The first practical scheme for the successive refinement and multiple description JSCC problems, achieved by a data-driven machine-learning approach;
- Introduction of a family of network architectures that are able to learn solutions with different complexity trade-offs;
- Outstanding performance at the task of image transmission when compared to digital schemes and negligible performance compromise due to multi-channel adaptation;

- Adaptability to different communication channel models (AWGN, Rayleigh fading), presenting graceful degradation over non-ergodic channels.

II. System Model

We consider wireless transmission of images over L parallel channels. Let $\boldsymbol{z}_i \in \mathbb{C}^{k_i}$ denote the complex channel input vector and $\hat{\boldsymbol{z}}_i \in \mathbb{C}^{k_i}$ the corresponding channel output vector for the i-th channel, $i \in [L] \triangleq [1, \ldots, L]$. We assume that the transmissions of $\boldsymbol{z}_i$ sequences are done through independent realizations of a noisy communication channel represented by the transfer function $\hat{\boldsymbol{z}}_i = \eta(\boldsymbol{z}_i)$, and consider in this work two widely used channel models: (a) the additive white Gaussian noise (AWGN) channel, and (b) the slow fading channel. The transfer function of the Gaussian channel is $\eta_n(\boldsymbol{z}_i) = \boldsymbol{z}_i + \boldsymbol{n}$, where the vector $\boldsymbol{n} \in \mathbb{C}^{k_i}$ consists of independent identically distributed (i.i.d.) samples from a circularly symmetric complex Gaussian distribution, i.e., $\boldsymbol{n} \sim \mathcal{CN}(0, \sigma^2 \boldsymbol{I}_{k_i})$, where σ^2 is the average noise power. In the case of a slow fading channel, we adopt the commonly used Rayleigh slow fading model. The multiplicative effect of the channel gain is captured by the channel transfer function $\eta_h(\boldsymbol{z}_i) = h\boldsymbol{z}_i$, where $h \sim \mathcal{CN}(0, H_c)$ is a complex normal random variable.

Let $\boldsymbol{x} \in \mathbb{R}^n$ denote the image to be transmitted. The receiver obtains a subset $\mathscr{S} \subseteq [L]$ of the channel output vectors, and creates a reconstruction $\hat{\boldsymbol{x}}_{\mathscr{S}} \in \mathbb{R}^n$. We consider two kinds of subsets: in the *successive refinement* problem, the receiver obtains channel output vectors corresponding to sequential and consecutive channels, i.e., $\mathscr{S} = [i]$ for some $1 \leq i \leq L$; in the *multiple description* problem, the receiver obtains channel output vectors from arbitrary combinations of channels. As different channel output subsets have different sizes, we achieve agile bandwidth in the sense that the same image can be transmitted and reconstructed with the use of different amounts of bandwidth.

We will call the image dimension n as the *source bandwidth*, and the dimension k_i of the i-th channel as the *channel bandwidth*. We will refer to the ratio k_i/n as the *bandwidth ratio* for the i-th channel.

An average power constraint is imposed on the transmitted signal at every channel, i.e,

$$\frac{1}{k_i}\mathbb{E}[\boldsymbol{z}_i{}^*\boldsymbol{z}_i] \leq P, \qquad \forall i \in [L], \tag{1}$$

where the average signal-to-noise ratio (SNR) given by: $\mathrm{SNR} = 10\log_{10}\frac{P}{\sigma^2}$ (dB).

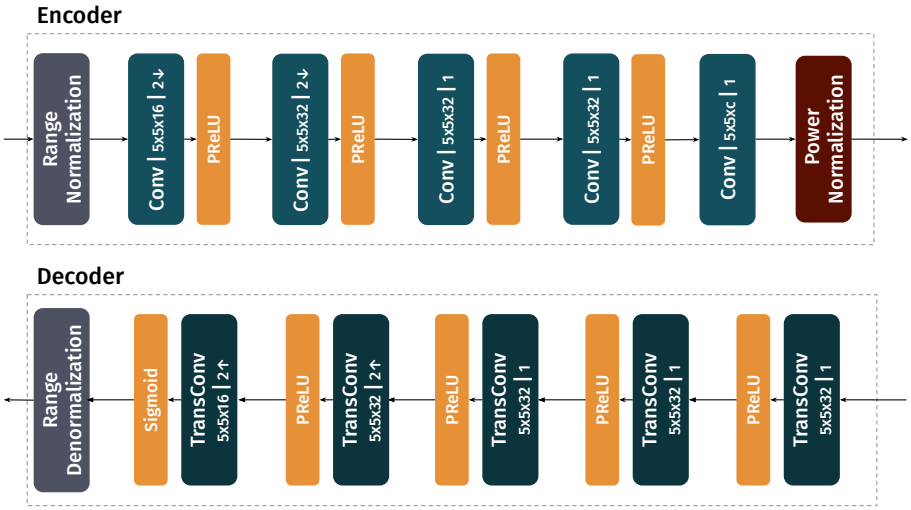


Fig. 2. The encoder and decoder components used in this paper, introduced in [17]. The notation $k \times k \times d/s$ refers to kernel size k, depth d and stride s, while c defines the encoder's compression rate.

Performance is evaluated by the peak signal to noise ratio (PSNR_j) between the input image $\boldsymbol{x}$ and a reconstruction $\hat{\boldsymbol{x}}_j$. The PSNR is inversely proportional to the mean square error (MSE), and both are defined as:

$$\text{MSE}_{\mathscr{S}} = \frac{1}{n}||\boldsymbol{x} - \hat{\boldsymbol{x}}_{\mathscr{S}}||^2 \tag{2}$$

$$\text{PSNR}_{\mathscr{S}} = 10\log_{10}\frac{\text{MAX}^2}{\text{MSE}_{\mathscr{S}}}, \tag{3}$$

where MAX is the maximum value a pixel can take, which is 255 in our case (we consider RGB images, with 8 bits per pixel per color channel).

III. DeepJSCC-l

Inspired by the success of [17], we propose the use of CNNs to represent both JSCC encoder and decoder, and add the channel to the model as a differentiable yet non-trainable layer, producing random values at every realization. All neural network components are trained jointly and the performance is optimized on realizations of an end-to-end communication system, forming an autoencoder architecture. Thus, our proposed architecture, called DeepJSCC-l, is built using neural encoders and decoders as basic components. We will primarily investigate the model with one encoder and multiple decoders (as can be seen

in Fig. 3 for the case of successive refinement with $L = 2$), but alternative models are also considered. The name DeepJSCC-l refers to the family of all the different architectures and solutions considered.

The encoder is a CNN and is represented by the deterministic function $f^{\boldsymbol{\theta}}$ parameterized by vector $\boldsymbol{\theta}$. It receives as input the source image $\boldsymbol{x}$, and outputs at once all the channel input symbols $\boldsymbol{z}$, i.e., we have $\boldsymbol{z} = f^{\boldsymbol{\theta}}(\boldsymbol{x})$ with $\boldsymbol{z} \in \mathbb{C}^k$, where k is the total bandwidth, i.e., $k = \sum_{i=1}^{L} k_i$. The channel input is $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_L)$, where $\boldsymbol{z}_i$ is transmitted over the i-th channel.

We consider that, for each valid subset $\mathscr{S}$ of channel output vectors received, a different decoder is employed to transform the noisy symbols into reconstruction $\hat{\boldsymbol{x}}_{\mathscr{S}}$. Thus, the decoder is a CNN represented by $g_{\mathscr{S}}^{\boldsymbol{\phi}_{\mathscr{S}}}$, where $\boldsymbol{\phi}_{\mathscr{S}}$ is the learned parameter vector. We denote the concatenation of all channel outputs for subset $\mathscr{S}$ by $\hat{\boldsymbol{Z}}_{\mathscr{S}}$, i.e., $\hat{\boldsymbol{Z}}_{\mathscr{S}} = \bigcup_{i \in \mathscr{S}} \hat{\boldsymbol{z}}_i$. The corresponding reconstruction is given by $\hat{\boldsymbol{x}}_{\mathscr{S}} = g_{\mathscr{S}}^{\boldsymbol{\phi}_{\mathscr{S}}}(\hat{\boldsymbol{Z}}_{\mathscr{S}})$.

We optimize all the parameters jointly to minimize the average distortion between the input image $\boldsymbol{x}$ and a partial reconstructions $\hat{\boldsymbol{x}}_{\mathscr{S}}$:

$$(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*) = \sum_{\mathscr{S}} \underset{\theta, \phi_{\mathscr{S}}}{\arg\min}\ \mathbb{E}_{p(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\mathscr{S}})}[d(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\mathscr{S}})], \tag{4}$$

where $\boldsymbol{\phi}$ is the collection of of all decoders' parameter vectors, $d(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\mathscr{S}})$ is a given distortion measure, and $p(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\mathscr{S}})$ the joint probability distribution of the original and reconstructed images, which depends on the channel and input image statistics, as well on the encoder and decoder parameters. Note that this is a multi-objective problem, as multiple reconstructions are considered and the parameter vector $\boldsymbol{\theta}$ is common in the optimization of every reconstruction. We address this problem by performing a joint training the sum of all the objectives, or by greedily training them sequentially (Section IV-E2).

Fig. 2 presents the NN architectures used for the encoder and decoder components. The encoder and decoder are symmetric, containing the same number of convolutional layers and trainable weights. The convolutional layers are responsible for feature extraction and downsampling (at the encoder) or upsampling (at the decoder) through stride and varying the depth of the output space. The last layer of the encoder is parameterized by depth c, which defines the total bandwidth ratio of all the layers combined: $k/n = (H/4 \times W/4 \times c)/(H \times W \times 3) = c/48$, where H and W are the height and width of an image with 3 color channels. After each convolution, we use the parametric ReLU (PReLU) [39] activation

function, or a sigmoid in the last block of the decoder to produce outputs in the range $[0, 1]$. Normalization at the beginning of the encoder, and denormalization at the end of the decoder convert values from range $[0, 255]$ to $[0, 1]$ (and vice versa). At the end of the encoder, the output of the last convolution layer is normalized as in Eq. (1) so the average power of each layer is constrained to $P = 1$. Note that the total channel bandwidth k used by DeepJSCC-l depends on the input dimension n; that is, the same model can output different channel bandwidths for different input sizes, keeping the bandwidth ratio constant. Thus, we will consider the bandwidth ratio (k/n) when presenting and comparing results.

The architecture in Fig. 2 is based on [17], where it is employed for single-layer transmission, and is shown to achieve competitive results with the state-of-the-art separation-based digital schemes. While we studied improved and slightly more complex versions of this architecture in a separate work [19], we use the original DeepJSCC architecture for most of the simulations in this paper, as our focus is to show that the single-layer model can be extended to multi-layer transmission. A more robust architecture is presented in Section IV-D.

All simulations are implemented on TensorFlow [40], using the Adam algorithm [41] for stochastic gradient descent, learning rate of 10^{-4}, and 64 images per training batch. As the model is fully convolutional, a trained model can accept images of arbitrary dimensions, but the results presented in this paper are for models trained and evaluated on the CIFAR-10 dataset [42] containing 50000 training images and 10000 test images with dimension $n = 32 \times 32 \times 3$. Results are presented in terms of the average PSNR calculated over the whole CIFAR-10 test dataset, each image transmitted over 10 independent realizations of the noisy channel.

IV. Successive Refinement JSCC

We start with the successive refinement problem, in which the decoder receives the outputs of the first i channels for some $1 \leq i \leq L$. We refer to the symbols transmitted over the first channel as the *base layer*, and the following channels as the *refinement layers*.

The first solution is based on an architecture consisting of a single encoder NN and L independent decoder NNs, as illustrated in Fig. 3 for $L = 2$. The whole system is modeled as an autoencoder and all the layers are trained jointly, with the loss function defined as:

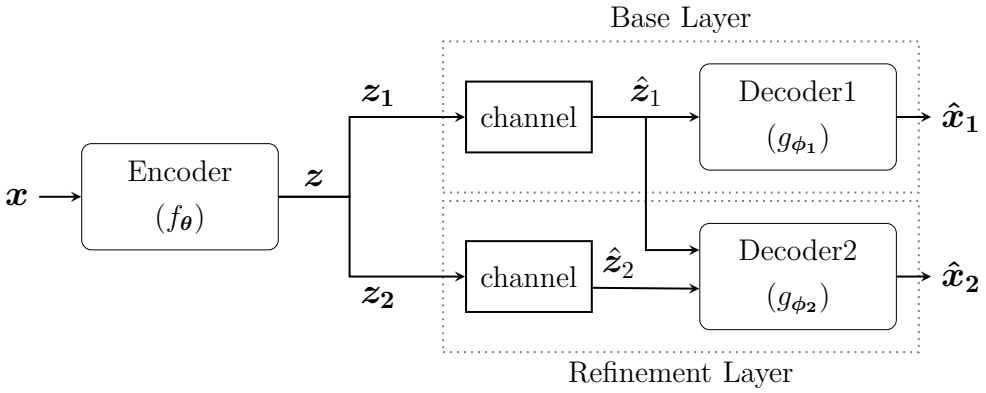


Fig. 3. DeepJSCC-l architecture for progressive wireless image transmission with two layers, performing successive refinement. An input image is encoded into layers $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, each of them transmitted over different realizations of the noisy channel.

$$\mathcal{L} = \frac{1}{L}\frac{1}{N}\sum_{j=1}^{L}\sum_{i=1}^{N} d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}_j^i), \tag{5}$$

where $d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}_j^i)$ is the MSE distortion between the original image $\boldsymbol{x}^i$ and its reconstruction at decoder j, $\hat{\boldsymbol{x}}_j^i$, for the i-th sample of the training dataset, and N is the number of training samples. Note that the loss function in (5) puts equal weights on the distortions of all the L decoders. Although a more general loss function could be formulated with different weights per distortion achieved by different decoders, experimental results showed that this has marginal impact on the performance. For more details, please see Appendix A.

A. Two-layer Model

Our first set of results focus on the $L = 2$ layers scenario, which requires the training of only one encoder and two decoders. We consider $k_1/n = k_2/n = 1/12$, and the AWGN channel. In Fig. 4, we present the results for different channel SNRs, where each point in the figure is achieved by training a distinct encoder-decoder pair. As a comparison baseline, we also present the performance achieved by the DeepJSCC scheme with a single layer using the same bandwidth as $\hat{\boldsymbol{Z}}_1$ and $\hat{\boldsymbol{Z}}_2$ ($k/n = 1/12$ and $k/n = 1/6$), respectively.

For all the channel conditions, the average PSNR_2 is consistently higher than PSNR_1 by 2 to 3 dB, showing the contribution of the refinement layer. The results also demonstrate that DeepJSCC-l can learn to transmit a sequential representation of the input images,

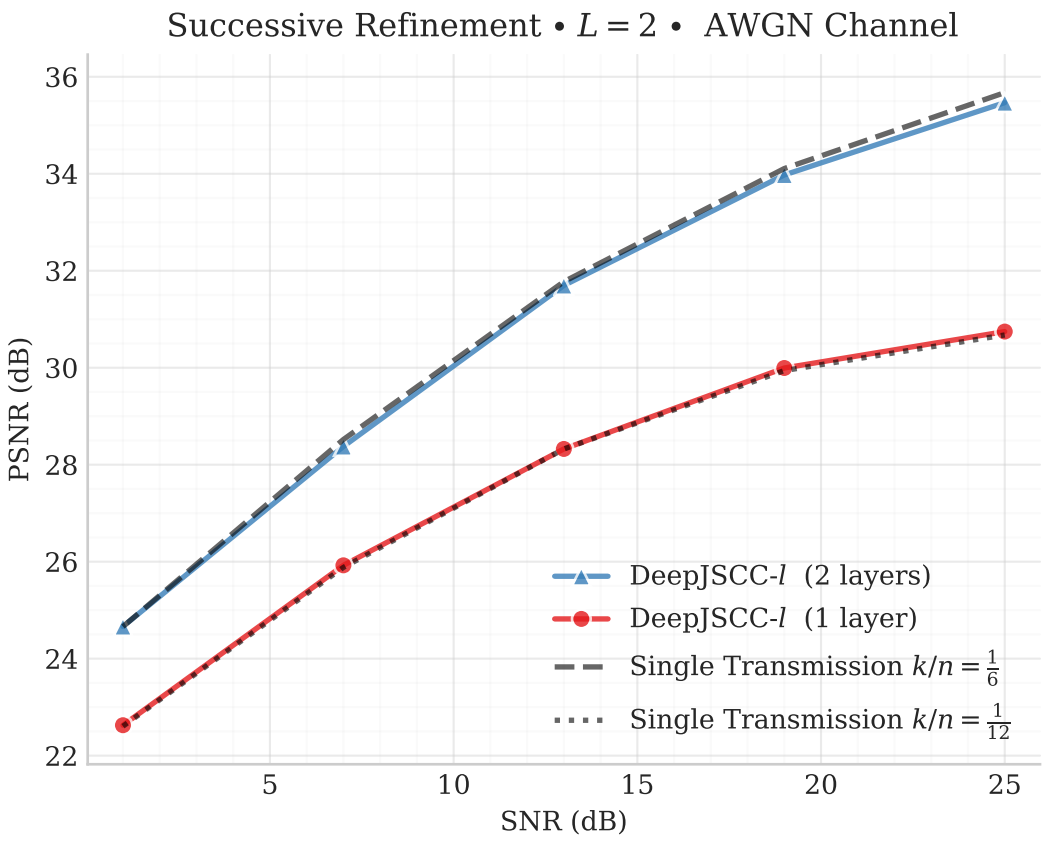


Fig. 4. DeepJSCC-l performance for successive refinement with $L = 2$ layers over a wide range of SNRs, for $k_1/n = k_2/n = 1/12$. Colored curves show the performance of reconstructions using both subsets of channel outputs ($\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$). Black dashed lines plot the performance of the single transmission model with equivalent bandwidth. Our results show that the loss due to layering is negligible.

while maintaining the performance close to the baseline curves. The fact that the performance loss compared to the baseline is negligible implies that DeepJSCC-l is able to find a nearly successively refinable representation over Gaussian channels; that is, the flexibility of allowing a decoder to reconstruct the imaged based only on the base layer, or both the layers comes at almost no cost in performance.

B. Adaptability to Varying Channels

A common issue in real systems is the mismatch of conditions between design and deployment stages. Often a system is designed having a specific target communication channel condition, but when deployed the channel conditions may have changed. Also, most practical systems rely on imperfect channel estimation and feedback, which results

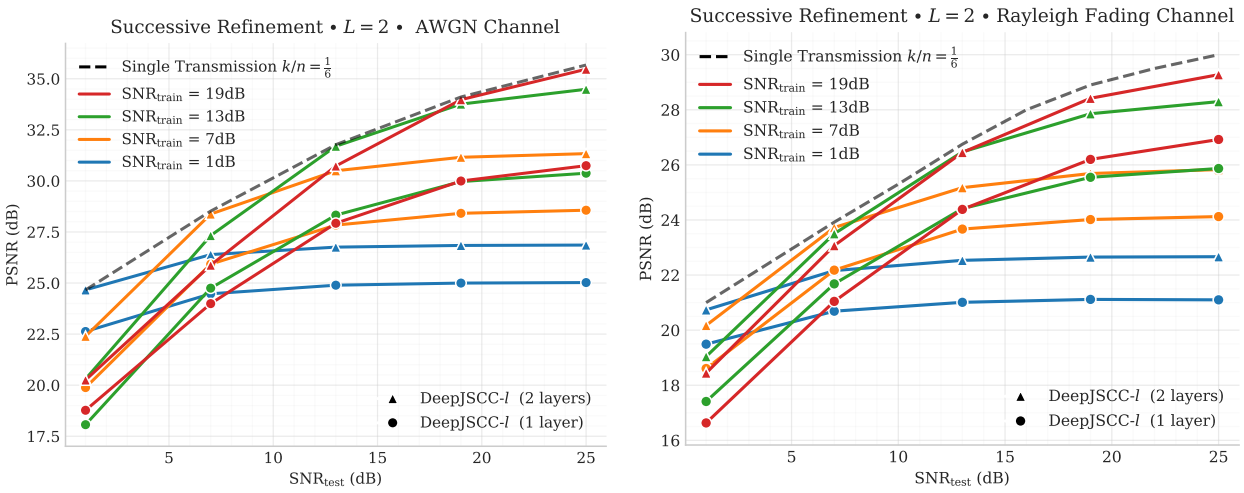


Fig. 5. DeepJSCC-l performance on successive refinement when there is disparity between training and test channel conditions, typical from multi-user communication. Each color represents the performance over a range of SNR for a DeepJSCC-l model trained for a specific SNR; triangle markers correspond to receivers using $k_1/n = 1/12$ bandwidth ratio (base layer), while circle markers correspond to receivers using $k_1/n + k_2/n = 1/6$ bandwidth ratio (base+refinement layers). Two channel models are considered: (a) AWGN channel and (b) slow Rayleigh fading channel.

in a mismatch between the channel state assumed at the transmitter (and used for picking the rate for compression and channel coding) and the real channel condition. This can be a serious issue for digital systems, as significant mismatch can lead to a total loss of information at the receiver, known as the *cliff effect*.

To simulate such a scenario, we consider the performance of DeepJSCC-l trained on a specific target SNR, but evaluated at a range of different channel conditions. Fig. 5a shows the performance results for both the base layer transmission and the base+refinement layers, where each color represents the performance of a model trained for a specific SNR, with the curve with circle markers corresponding to the performance of the decoder receiving only the base layer, while the one with triangle markers the decoder receiving both layers. As reference, we also plot with the black dashed line the best performance at different SNRs of a single-layer DeepJSCC.

The results show that the performance of DeepJSCC-l deteriorates gradually (yet not abruptly) for both reconstructions when the test SNR is lower than the trained SNR, showing that it is robust against SNR mismatch. Similarly, unlike in digital systems, the performance

of DeepJSCC-l for both reconstructions improves gradually with the channel SNR. This shows that DeepJSCC-l does not suffer from the cliff effect but instead presents *graceful degradation*. Note that this is a behavior typical for analog systems and was already observed in the single layer case in [17]. We also observe that the performance gap between the layers tend to remain constant when the test SNR is higher than the trained SNR, but the gap reduces as the test SNR falls below the training SNR. This is because the benefit from the refinement layer degrades as the test SNR decreases since the reconstructed base layer becomes significantly different from what the encoder expects based on the training data.

We also train DeepJSCC-l over a slow Rayleigh fading channel, when the channel realization remains constant for the duration of the transmission of each layer, but takes an independent value for the transmission of each image. This scenario can also represent a multi-user multicasting scenario, in which a different "virtual" receiver corresponds to each realization of the channel.

Fig. 5b shows the results for the same model architecture as in Fig. 5a, where the x-axis denotes the average SNR in the test phase. We see that, although the PSNR values are lower than those in the AWGN case due to channel uncertainty, the properties of graceful degradation and limited loss with respect to the single-layer baseline are preserved.

We highlight here that DeepJSCC-l does not exploit explicit pilot signals or channel estimation, yet it is able to adapt to the channel uncertainty. All the models presented in this paper exhibit similar behavior of graceful degradation and capacity to learn over fading channels. In the remainder of this paper, we only present the highest PSNR obtained for each channel SNR value, and only consider transmissions over an AWGN channel.

Remark 1. *We remark here that, due to the analog nature of DeepJSCC-l, the reconstruction at the receiver based on the first l layers is not fixed, and depends on the realization of the random channel. Therefore, unlike in digital systems, the exact reconstruction at the decoder cannot be known by the encoder in advance; and hence, the second layer cannot simply transmit the residual information. It is remarkable that DeepJSCC-l can learn to refine the previous reconstructions despite this uncertainty, even in the case of a fading channel.*

C. Multiple Layers

Next, we extend the model to more than two layers. Fig. 6a shows the results for $L = 5$ layers, each transmitted with a bandwidth ratio equal to 1/12. The results show that

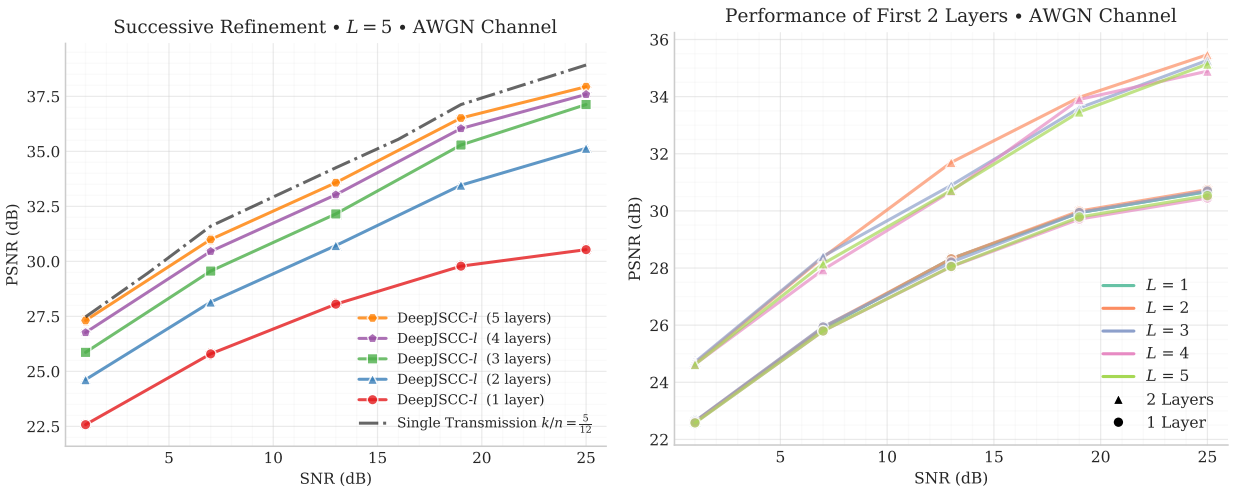
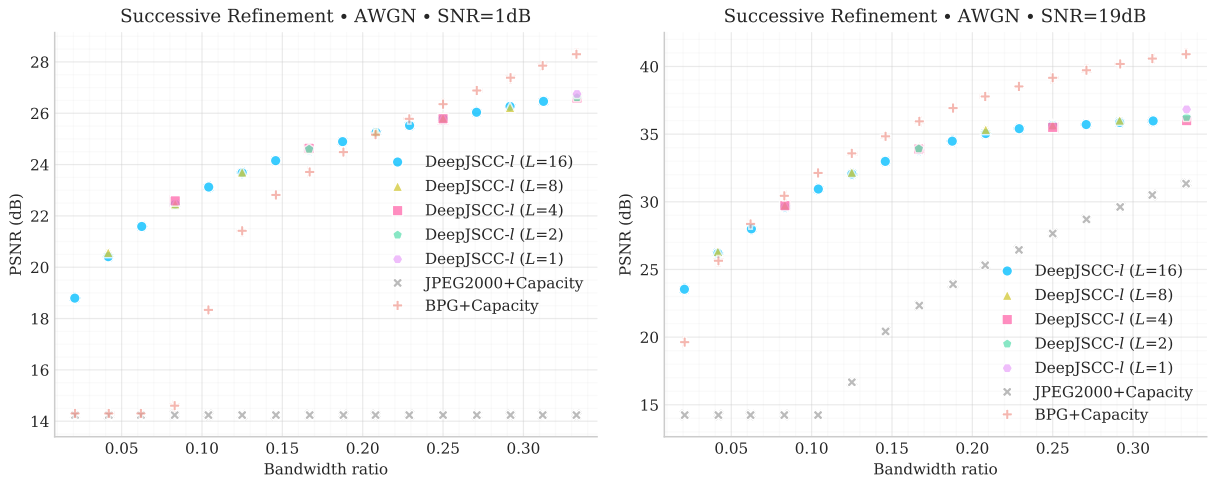


Fig. 6. (a) Performance of DeepJSCC-l using $L = 5$ layers over different SNRs. Note that the increase in performance with each refinement layer gradually decreases. (b) Performance of the two first layers ($\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$) for DeepJSCC-l trained with different values of L. Note that despite the increase in the number of layers, the performances of the first two layers remain relatively stable. In both plots, $k_i/n = 1/12,\ \forall i \in \{1 \dots 5\}$.

the addition of new layers increases the overall quality of the transmitted image at every step; although the amount of improvement is diminishing, as the model is able to transmit the main image features with the lower layers, leaving only marginal contributions to the additional layers.

We also notice that the introduction of additional layers in the training model has very low impact on the performance of the first layers, compared to models with smaller values of L. This can be seen in Fig. 6b, which compares the performance of the first and second layers for models trained with $L \in \{2, 3, 4, 5\}$, showing that the loss due to the addition of new layers is negligible. This is rather surprising, given that the code of the first layer is shared by all the layers and is optimized to be maximally useful in combination with a number of refinement layers. The results, therefore, suggest that there is almost performance independence between layers, justifying the use of as many layers as desired, as long as there are available resources.



(a) (b)

Fig. 7. PSNR vs. bandwidth ratio comparison for $L = 1, 2, 4, 8$ and 16 layers at (a) SNR = 1dB and (b) SNR = 19dB. DeepJSCC-l presents superior performance for the first layers when compared to a separation-based scheme using JPEG2000 (with 16 layers) or BPG for compression, and an ideal capacity-achieving code.

D. Comparison with Digital Transmission

Finally, we consider an experiment in which a fixed bandwidth k is divided into L layers of equal size. Fig. 7 shows the results of five different cases corresponding to $L = 1, 2, 4, 8, 16$, and total bandwidth ratio $k/n = 1/3$ for SNR = 1dB (Fig. 7a) and SNR = 19dB (Fig. 7b). The performance of all the reconstructions for each model are shown. We observe that there is almost no loss in performance by dividing the transmission into many layers, as many as $L = 16$, while this provides additional flexibility, i.e., a receiver may stop receiving after having received a certain number of layers if it has reached a certain target quality, and may use the bandwidth and processing power for other tasks.

For comparison, we also consider digital transmission employing separate source and channel codes. In particular, we consider both JPEG2000[1] and BPG[2] as the source encoder, followed by a capacity-achieving channel code. JPEG2000 is chosen as it can to generate layered representations at different bit rates; the choice of BPG is motivated by its high performance in image compression. The capacity-achieving channel code is an ideal

[1]https://jpeg.org/jpeg2000/index.html

[2]https://bellard.org/bpg/

formulation assuming that bits can be transmitted without errors at the channel capacity [43]. Although near capacity-achieving channel codes exists for the AWGN channel, what we are assuming here is not feasible in practice for the blocklengths considered here. Thus, this scheme would serve as an upper bound on the performance of any separation based digital scheme employing JPEG2000 or BPG for compression.

The digital scheme works as follows. For a given bandwidth ratio k_i/n, source dimension n, and the channel capacity at each SNR, a bit budget b_i is determined as the maximum number of bits that can be transmitted over k_i channel uses. When using JPEG2000 as source code, we compress images into L layers, each using at most b_i bits. For BPG, as the official encoder does not allow layered compression, we produce independent compressions with the best possible quality for each b_i target. For fair comparison we discard the bits dedicated to header, so only the compressed pixels are transmitted.

The results show that DeepJSCC-l can achieve performance superior than or comparable with the state-of-the-art separation-based schemes using JPEG2000 and BPG codecs, particularly for the lower layers. The superior performance is particularly noticeable in Fig. 7a where the low SNR (1dB) and the constrained bandwidth ratio limits the channel capacity so much that the JPEG2000 and BPG codecs are unable to compress images to the level supported by the channel, resulting in the flat curve regions displayed in the graph.

We have also experimented extending DeepJSCC-l with different layers and hyperparameters, and training the network on larger datasets. In particular, we experimented using the network hyperparameters in [19] for the encoder and decoder, which uses bigger convolutional filters and apply generalized normalization transformations [44] inspired by the sate-of-the-art for neural image compression models in [33]–[35]. The models were trained on the ImageNet dataset, consisting of more and larger resolution images than CIFAR-10. The model is then evaluated on the Kodak dataset and the PSNR curves compared to the single-layer model for $L = 2$ is provided in Fig. 8, while examples of reconstructions are presented in Fig. 9. Although these more complex network architectures produce remarkable performance (see [19] for a detailed comparison with many codecs), they also require significantly larger training time, particularly for multi-layer compression tasks, which is why we have limited our numerical results for more layers to the simpler architecture of [17].

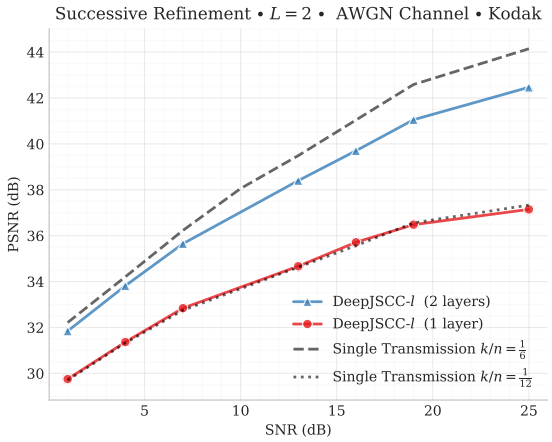


Fig. 8. Performance results with encoder/decoder architecture introduced in [19]. Model was trained on ImageNet dataset and evaluated on Kodak dataset.

(a) Original image

(b) Base reconstruction, $\hat{\boldsymbol{x}}_1$
PSNR: 31.90 / MS-SSIM: 0.9749

(c) Refined reconstruction, $\hat{\boldsymbol{x}}_2$
PSNR: 34.45 / MS-SSIM: 0.9861

(d) Highlight $\hat{\boldsymbol{x}}_1$

(e) Highlight $\hat{\boldsymbol{x}}_2$

Fig. 9. Examples of reconstructions of successive refinement model. Note how flower details are enhanced between Figures (d) and (e).

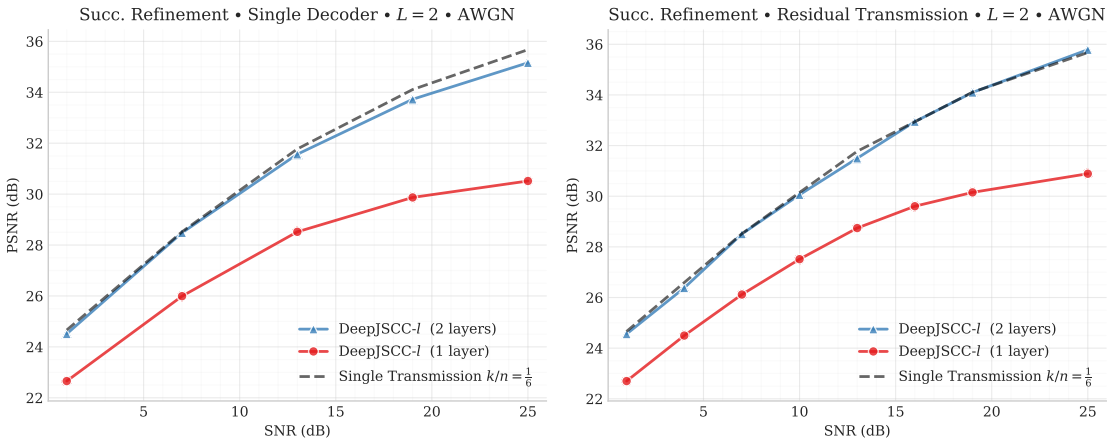


Fig. 10. Performance of alternative successive refinement DeepJSCC-l architectures on CIFAR-10 test images, transmitted over an AWGN channel with $k_1/n = k_2/n = 1/12$. (a) Single decoder scheme, (b) Residual transmission, $m = 10$. Both architectures have performance equivalent to the first scheme proposed (multiple decoders), but presenting different space and time complexities.

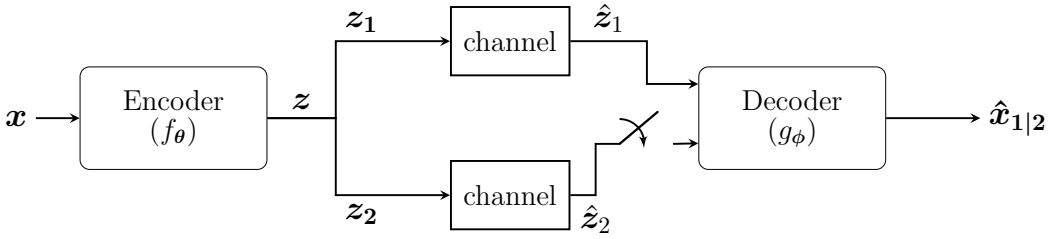


Fig. 11. Single decoder scheme with two layers. A single decoder is trained with different input sizes, being able to reconstruct the image with as many layers as it is provided with.

E. Alternative Architectures

The model architecture for DeepJSCC-l introduced in Fig. 3 does not represent the only viable solution for the successive refinement problem. Here, we discuss alternative DeepJSCC-l architectures with different trade-offs. We note that the trade-off is between the space and time complexity, and not necessarily the performance, as all the methods we present below achieve comparable performance to the one presented above.

1) Single Decoder: A downside of the model used previously (Fig. 3) is the fact that a separate decoder needs to be trained for each layer. Here we try an alternative model that uses a single encoder and a single decoder architecture for all the layers, as illustrated in

Fig. 11. In order to retrieve information from partial subsets, the decoder has to be trained for different input sizes. We achieve that by exposing a single decoder to different code lengths, and averaging its performance over all possible subsets of layers. In practical terms, that means creating a CNN model with fixed channel bandwidth $k = \sum_{i=1}^{L} k_i$, but randomly masking consecutive regions of size k_i from the end of the received message $\hat{\boldsymbol{z}}$ with zeros. In this way, the network should learn to specialize different regions of the code, using the initial parts to encode the main image content and the extra (occasionally erased) parts for additional layers. This can also be considered as "structured dropout", where the dropout during training allows training a decoder that can adapt to the available bandwidth.

Note that during training, the length of the transmitted code (i.e., the number of layers) is defined randomly at every batch. This is essential so that the encoder and decoder can preserve the performance of all the layers. An alternative approach that train subsets of layers sequentially until convergence with sizes 1 to L showed to be detrimental to the performance of the first layers. This happened because the training of higher order layers modified the parameters of previous layers.

The results presented in Fig. 10a for $L = 2$ layers show that the performance of DeepJSCC-l with a single decoder is close to the single transmission bound. The achieved values are as good as in the multiple decoder architecture (Fig. 4). This model is particularly appealing as it represents a considerable reduction both in memory and processing as the model size remains the same regardless of the number of layers. However, while the multiple decoder scheme learns separate decoders for all the layers in parallel, the single decoder strategy has to be presented with different codelengths, increasing the training time.

2) Residual Transmission: Another alternative architecture we propose is based on residual transmission. Here, as illustrated in Fig. 12, each transmission is performed by an independent encoder/decoder pair acting sequentially. Instead of jointly optimizing all the parameters of all the layers simultaneously, we use a greedy approach in which an encoder/decoder pair is trained until convergence and their weights are fixed (frozen) so new pairs can be trained on top of it.

The first encoder/decoder pair (the base layer) behaves exactly as in the single transmission scheme, transmitting the original image $\boldsymbol{x}$, compressed at rate k_1/n, and retrieved as $\hat{\boldsymbol{x}}_1$. Then, in each subsequent layer j, the encoder uses as input the original image being transmitted, $\boldsymbol{x}$, and an estimate of the residual error between the original image and its

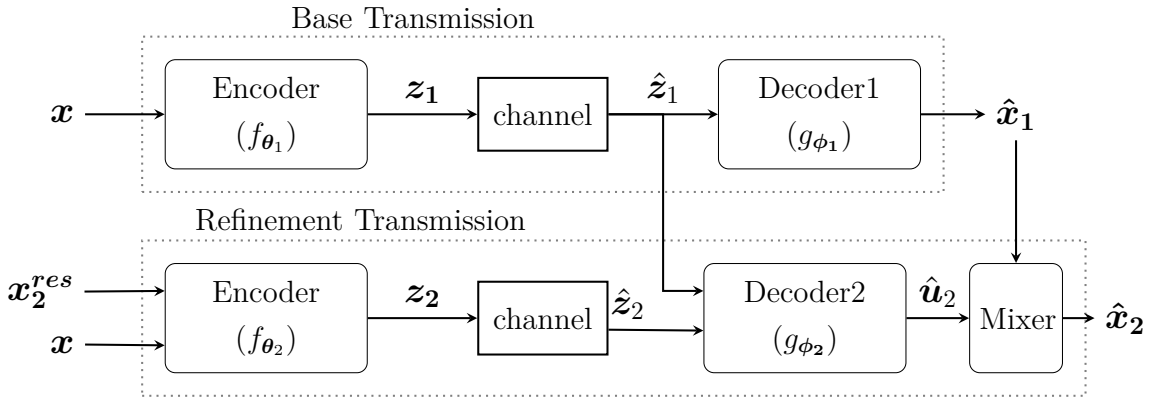


Fig. 12. Residual transmission scheme with two layers. At each layer, the residual of the previous transmissions is estimated and transmitted. Additional layers can be introduced without the need to retrain existing layers.

estimate of the receiver based on the previous $j-1$ layers, $\hat{\boldsymbol{x}}_{j-1}$,

$$\boldsymbol{x}_j^{res} \triangleq \boldsymbol{x} - \hat{\boldsymbol{x}}'_{j-1}.$$

Here, since the transmitter does not know the reconstructed image at the receiver, $\hat{\boldsymbol{x}}'_{j-1}$ is an estimate of $\hat{\boldsymbol{x}}_{j-1}$ based on the statistics of the dataset and the channel. We assume the transmitter has a local copy of the decoder parameters at previous layers. So, in order to generate $\hat{\boldsymbol{x}}'_{j-1}$, the transmitter simulates locally independent realizations of the channel and the decoder models, obtaining

$$\hat{\boldsymbol{x}}'_{j-1} = \frac{1}{m}\sum_{i=1}^{m} \tilde{\boldsymbol{x}}^i_{j-1},$$

where, with abuse of notation, $\tilde{\boldsymbol{x}}^i_{j-1}$ is the i-th realization of the simulation of the transmitter's image reconstruction, and m is the total number of independent channel realizations used to estimate the receiver's output. Note that this estimation at the transmitter side is necessary because we assume no feedback channel between the receiver and transmitter. In the presence of feedback, the receiver's reconstruction could be sent back to the transmitter [45].

In the residual transmission scheme, each layer $i > 1$ encodes and decodes an estimated residual image, containing the missing information not transmitted yet, that can be combined with the reconstruction at $i-1$, producing the refinement. The combination of the

previous reconstruction and refinement is done by the decoder network. At layer i, the decoder i receives as input the concatenation of all the channel outputs received so far to reconstruct a residual estimate $\hat{\boldsymbol{u}}_i$. Later, $\hat{\boldsymbol{u}}_i$ is combined with the reconstruction at the previous layer $\hat{\boldsymbol{x}}_{i-1}$ by a mixer network, formed by two sequential convolutional layers, to produce the final reconstruction $\hat{\boldsymbol{x}}_i$.

Results of this scheme can be seen in Fig. 10b for the same scenario considered in Fig. 10a, using $m = 10$ for the received image estimations. The results show that the scheme is able to achieve results very close to the previous schemes. As expected, the first layer performance is exactly the same as single transmission with rate $k/n = 1/12$, given that the base layer is trained without the knowledge of subsequent layers. Particularly interesting, however, is the fact that the network is able to predict a valid residual representation, from the estimation of the channel using only $m = 10$ independent realizations.

The main advantage of this scheme is the fact that each encoder/decoder pair can be optimized separately, given the result of the previous layers. Although this is more computationally demanding, it allows design flexibility; as opposed to the first two architectures, this architecture allows adding new layers as required, without the need to retrain the whole encoder/decoder network from scratch. This could be used, for example, in a dynamic system that adds refinement layers as resources become available, or in a distributed communication setting, in which relay transmitters located at different regions complement the transmission by sending refinement images. We would like to note that this scheme can also be combined with other transmissoin techniques, i.e., a refinement layer can be transmitted for a base layer transmitted using a digital separation-based approach, in which case, the exact reconstruction at the receiver would be known.

3) Architecture Comparison: In the previous sections three alternative DeepJSCC-l architectures for successive refinement have been introduced: (a) multiple decoder networks, (b) a single decoder network, and (c) residual transmission. Numerical results show that all three architectures achieve nearly the same performance, suggesting that they can all produce successively refinable representations of natural images over an AWGN channel. However, while all the schemes are equivalent in terms of performance, other aspects can be considered when choosing the architecture to be used in practice.

In terms of memory complexity, the single decoder architecture has clear advantages, as it requires only one encoder and one decoder network, regardless of L. The residual

transmission scheme is the most expensive, as for every layer in L, a new pair of encoder and decoder has to be trained. The multiple decoder scheme needs training different decoders per layer, but has only one encoder regardless of L.

In terms of time (computational) complexity, as the encoder and decoder blocks used in all the schemes are the same, all but the residual transmission scheme have equivalent complexity during inference phase. The residual transmission scheme has additional overhead, as it has the extra steps of emulating each transmission m times and mixing different layers' outputs. In terms of time complexity during training, the multiple decoder architecture has advantage over the others as it can train all the layers simultaneously and in parallel, given that each layer has its own decoder. The single decoder scheme increases the time complexity of the training, as different layers should be trained sequentially, requiring more iterations of the algorithm until convergence. Lastly, the residual transmission scheme has the highest training time complexity, as apart from having to train each layer sequentially (as in the single decoder), it also has to train the mixer component to estimate the image reconstructions at the receiver. However, as stated previously, although more memory and time consuming during training, the residual network is the only scheme that allows the addition of new layers a *posteriori*, without the need of retraining the networks of previous layers.

V. Multiple Description JSCC

In multiple description communications we still transmit the image over L parallel channels, but we have a distinct virtual decoder corresponding to any subset $\mathscr{S} \subseteq [L]$ of these channels. For example, with $L = 2$ layers, we have three potential decoders, as illustrated in Fig. 13. While decoders 01_2 and 10_2, each decodes the underlying image from only one of the layers, decoder 11_2 decodes the same image using both layers. In general, all possible subsets can be indexed with binary numbers formed by L bits, so that the i-th least significant bit is 1 if $i \in \mathscr{S}_j$ or 0 otherwise. Thus, we can have a total of $2^L - 1$ decoders (excluding the empty subset), for all possible combinations of channel outputs.

Note that, in the $L = 2$ case, if we remove Decoder 10_2 we recover the successive refinement problem. The multiple description problem is a generalization of the successive refinement problem, and it is considerably more challenging as it has to be able to combine any subset of the channel outputs to reconstruct the image, and hence, there is no natural ordering of the

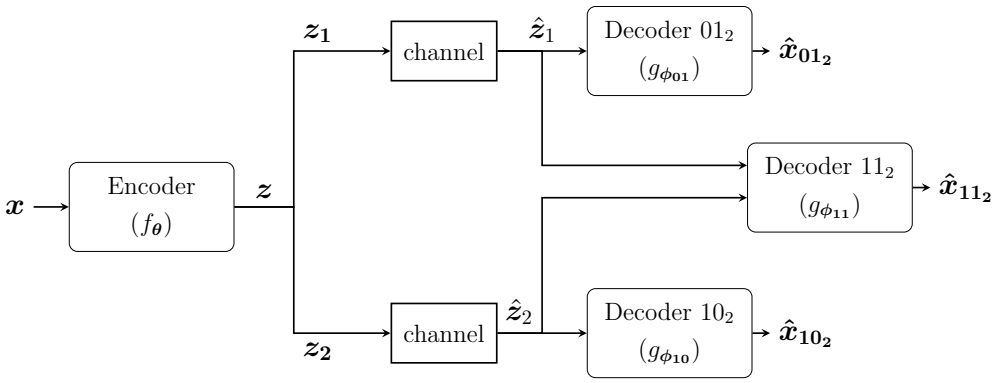


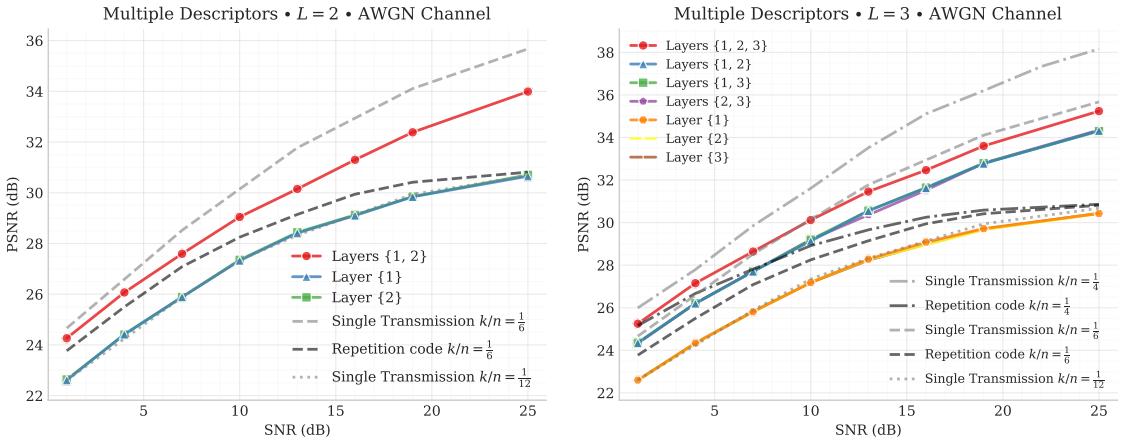
Fig. 13. DeepJSCC-l for multiple descriptions problem, where all possible subsets of channel outputs are received and decoded by decoders. Here, with $L = 2$, decoders 01_2 and 10_2 reconstruct the image using distinct sets of channel outputs, while decoder 11_2 uses all available information for its reconstruction. Note that decoder indexes and parameters, and reconstructions are indexed in binary base.

transmissions into layers. In general, multiple description coding should be used when each codeword can be received independently, whereas successive refinement is more appropriate when there is an ordering among the channels, i.e., the signal over the i-th channel is received successfully iff all the previous transmissions are received. For example, this might be the case if the channels are ordered in time, and the receiver stops after receiving a random number of channels. On the other hand, if we consider transmission over an OFDM system with L subcarriers, where different receivers are capable of receiving over different subsets of the subcarriers. The transmitter will need to employ a multiple description encoding scheme to guarantee that the image can be reconstructed by tuning into any subset of the subcarriers.

Similarly to the previous section we will present different possible architectures to realize multiple description coding; however, since the layers are independent and not sent sequentially, the residual transmission model does not apply here.

A. Multiple Decoder Architecture

The encoder-decoder DeepJSCC-l architecture with a single encoder network and multiple decoders proposed in Section IV can be expanded and adapted to the multiple description problem. A single encoder generates vector $\boldsymbol{z}$ with channel bandwidth k, and $2^L - 1$ decoders are trained jointly using as inputs all different channel output subsets. Thus, we modify Eq.



(a) (b)

Fig. 14. Performance of multiple description problem on CIFAR-10 test images. (a) $L = 2$, AWGN channel, $k_1/n = k_2/n = 1/12$ (b) $L = 3$, AWGN channel, $k_1/n = k_2/n = 1/12$.

(5), producing the following loss function:

$$\mathcal{L} = \frac{1}{(2^L - 1)N} \sum_{j=1}^{2^L-1} \sum_{i=1}^{N} d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i_j). \tag{6}$$

Fig. 14a and 14b show results for $L = 2$ and $L = 3$, respectively. We consider individual layers with constant size (i.e., $k_i = k/L$, $\forall i \in 1, \ldots, L$), so the decoders work with bandwidth as multiples of k/L. In all the experiments, we consider $k_i/n = 1/12$ as the bandwidth ratio. We see in these figures that the quality of the reconstruction of all the decoders with a single layer (i.e., k/L bandwidth) is equivalent, and is almost as good as what a single layer encoder with the same dimension would produce. When more than one layer is available, the decoder can reconstruct the input image with much better quality compared to the single-layer decoders; the combined performance, however, is inferior to a scheme which would only target the joint decoder. This is in contrast to the successive refinement problem, in which case the successive refinability could be achieved with almost no loss in the final performance. This performance loss is expected, and can be explained by the fact that, as each single-layer receiver tries to reconstruct the whole input $\boldsymbol{x}$ on its own, the information context common to both increases, and as a result, the amount of information available for the multi-layer decoders decreases. Such a rate loss is also observed in theoretical results for multiple description coding. For example, while Gaussian sources are successively refinable;

that is, they can be compressed into multiple layers, each reconstruction operating on the optimal rate-distortion curve, this is not possible in the case of multiple description coding [9]. We also consider another baseline where a simple repetition code is used to transmit the same codeword over both channels, and the receiver decodes using the average of the different channel output signals it receives. Effectively, multiple codewords in this scheme are used to reduce the effect of channel noise, rather than providing complementary source descriptions. Therefore, this scheme provides gains from multiple layers only in the low SNR region, and we can see that our model outperforms this baseline at all SNR values, indicating that different layers learn complementary representations of the image, transmitting more information than a single layer repeated.

As in Section IV-D, the encoder and decoder architectures of the multiple descriptor model can be extended and trained in more complex datasets. Fig. 15 shows samples of images from the Kodak dataset from the extended model ([19]) trained on Imagenet. While the improvement from both layers is difficult to observe by naked eye at this resolution, certain details can be noticed, e.g., letter "A" is more clear in Fig. 15b compared to the other two reconstructions.

B. Single Encoder-Decoder Network

The single decoder model can be adapted to this scenario by simply training a decoder that inserts zeroes on blocks that are not received, according to their positions in the latent vector. The training and evaluation procedures remain the same as in the successive refinement case. The same trade-offs apply here, that is, while multiple decoders save in training time, the single decoder saves in memory. Note, however, that the number of subsets of possible channel output layers increase exponentially with L, making the single decoder's learning task much more complex. Fig. 16 shows results for $L = 2$ and $L = 3$. Although the results are not as good as those of the multiple decoders, single layer transmission has comparable performance to a single transmission with equivalent bandwidth, and transmission with multiple layers are in general better than the repetition scheme with the same bandwidth.

VI. Summary and Conclusions

We have explored the use of deep learning based methods for progressive image transmission over wireless channels. Building on recent results showing that artificial neural networks

(a) original image

(b) $\hat{\boldsymbol{x}}_{\mathbf{11_2}}$
PSNR: 26.85 / MS-SSIM: 0.9671

(c) $\hat{\boldsymbol{x}}_{\mathbf{01_2}}$
PSNR: 24.98 / MS-SSIM: 0.9487

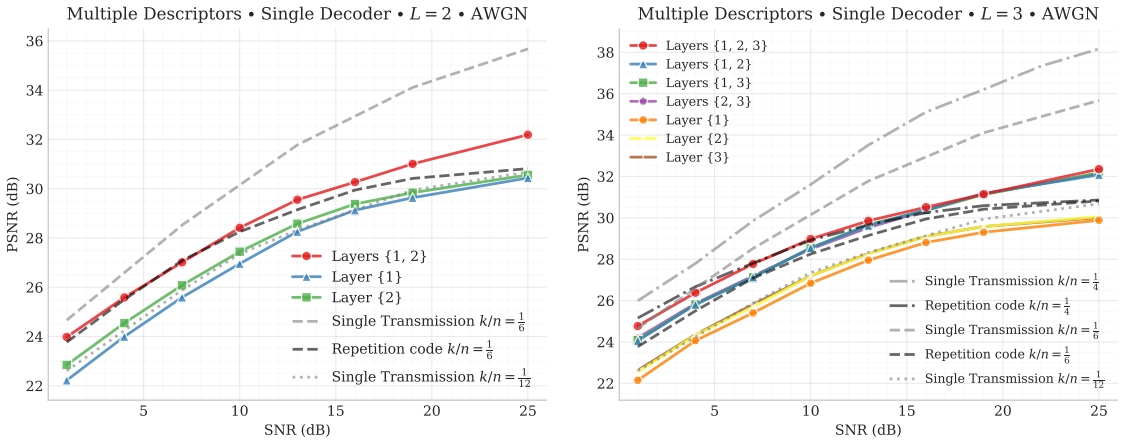
(d) $\hat{\boldsymbol{x}}_{\mathbf{10_2}}$
PSNR: 24.96 / MS-SSIM: 0.9472

Fig. 15. Examples of reconstructions for different subsets of multiple description problem for $L = 2$.

can be very effective in learning end-to-end JSCC algorithms, we explored whether the network can be extended to also learn successive refinement strategies, which would provide additional flexibility.

We introduced DeepJSCC-l, a group of deep-learning based JSCC algorithms able to encode and decode images over multiple channels, allowing flexible and adaptive-bandwidth transmissions. To the best of our knowledge, this is the first time that a hierarchical JSCC scheme has been developed and tested for practical information sources and channels.

We presented a series of architectures and experimental results, highlighting practical applications for DeepJSCC-l. The results show the versatility of the model to not only



(a) (b)

Fig. 16. Performance of the single decoder architecture, with the same configurations as in Fig. 14.

learn the layered representation (for both successive refinement and multiple description problems), but also superior performance when compared to state-of-the-art methods over a wide range of SNRs and channel bandwidths. Adaptability to environmental changes is also demonstrated, with the model showing graceful degradation when there is mismatch between the design and the deployment channel qualities, and the possibility to learn to operate in diverse channels, such as fading channels.

Appendix A
Multi-Objective Trade-offs

Both the successive refinement and multiple description problems are formulated as a multi-objective problem. Our models made the assumption that all objectives have equal weights, as shown in Eqns. (5) and (6), in which all the losses are averaged with equal weights. Alternative approaches can also be considered with different weights.

A. Successive Refinement Trade-offs

In the successive refinement problem, one can consider that each layer's reconstruction has different weights, so reconstructions with less or more bandwidth can be prioritized. Thus, we can rewrite Eqn. (5) as:

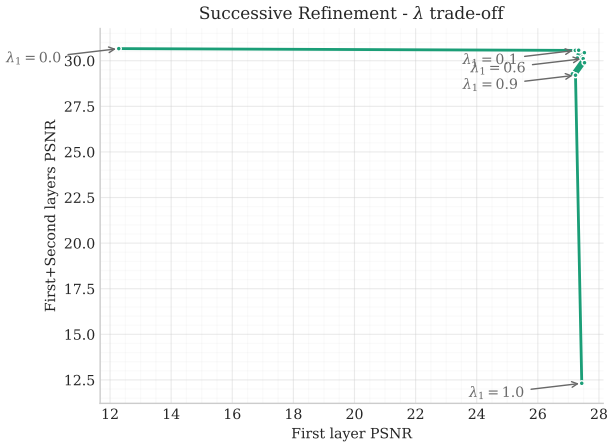


Fig. 17. Trade-off between the PSNR achieved by the base layer and that is achieved by combining both layers in the successive refinement problem.

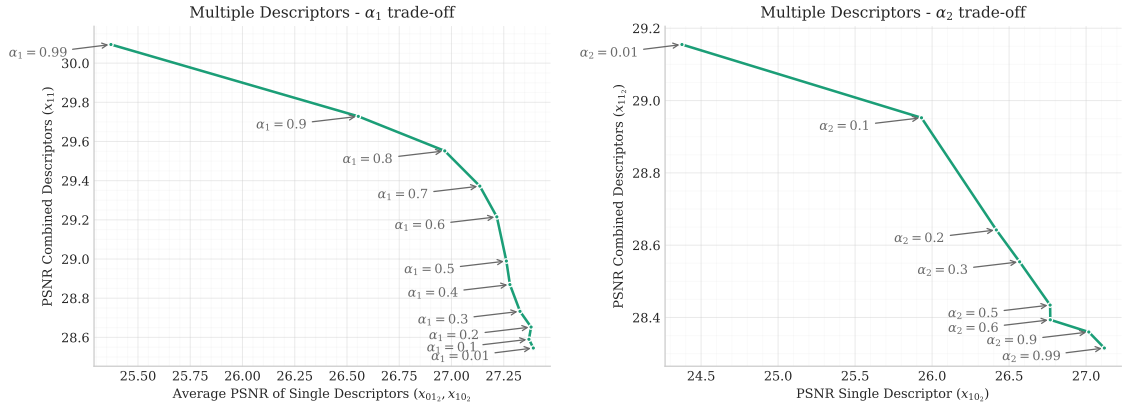
$$\mathcal{L} = \frac{1}{L}\sum_{j=1}^{L} \lambda_j d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}_j^i), \tag{7}$$

where $\sum_{j=1}^{L} \lambda_j = 1$.

We consider $L = 2$, and set $\lambda_1 = 1 - \lambda_2$. Fig. 17 presents the simulation results. When extreme cases are considered ($\lambda_1 \cong 0$, or $\lambda_1 \cong 1$), only one of the layers dominate, as expected, with the performance of the other diminishing (12.5 dB). However, for all the other intermediate values of λ_1, the choice has small impact on the overall performance of the model. This is in line with the claim that DeepJSCC-l is essentially successively refinable, so the addition of weights will not interfere in the overall performance. Therefore, we use the same weights (i.e., $\lambda_j = 1/L, \forall j \in 1, \ldots, L$) in all the experiments presented in the paper.

B. Multiple Description Trade-offs

As with the successive refinement problem, a multiple description transmission scheme needs to balance multiple objectives, each corresponding to the reconstruction quality of a different subset of layers. We can simplify the trade-off between different subsets by targeting the same quality if the image is decoded from the same number of layers. We will simplify further, and assume that we only consider decoders that receive single layers (indexed by $j = 2^l, \forall l \in 1, \ldots, L-1$) and the decoder that receives all the layers ($j = 2^L - 1$). We will



(a) (b)

Fig. 18. Performance impact of varying the weights of different components in the multiple description problem. (a) Combined transmission vs. single components; (b) multiple description vs successive refinement.

then have two different quality targets, one achieved by decoding a single layer, and the other by jointly decoding all the layers. To understand the trade-off between the two, we modify the loss function in Eqn. (6) adding a weight α_1 as follows:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(\alpha_1 d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i_{2^L-1}) + (1-\alpha_1)\frac{1}{L}\sum_{l=0}^{L-1} d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i_{2^l})\right). \tag{8}$$

Note that, when $\alpha_1 = 1$, we only care about the joint decoder and recover the non-layered DeepJSCC scheme, and when $\alpha_1 = 0$, we only care about the single-layer decoder, which correspond to L different transmissions with $1/Lth$ of the bandwidth ratio.

Fig. 18a shows the results comparing the performance of the joint multi-layer transmission (y axis) and the average performance of single descriptor (x axis) for different values of α_1 and $L = 2$. The figure clearly illustrates the trade-off between the performance of the side and joint decoders: for small values of α_1 the side decoders' average performance improves, approaching that of a single transmission line, as shown in Fig. 14a. On the other hand, as α_1 increases, the performance of the joint decoder improves, at the expense of the side decoders. When α_1 approaches 1, we approach the performance of a single decoder using all the available channel bandwidth.

Another possible trade-off is the choice between giving all the subsets the same weights, or prioritizing a sequence of subsets that produce successive refinement. Thus, the loss function, for the case of $L = 2$ becomes:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left[(1-\alpha_2)\left(d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i_{11_2}) + d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i_{01_2})\right) + (\alpha_2) d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i_{10_2})\right]. \tag{9}$$

Fig. 18b presents the results for different values of α_2, comparing the performance of the second descriptor, $d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i_{10_2})$, and the combined successive refinement transmission, $d(\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i_{11_2})$. The results show the impact in the performance of the successive refinement when the second descriptor is used to independently represent a full image (instead of just refining the first descriptor). The higher the α_2, the more emphasis is given to the decoding performance of the second descriptor alone, which decreases the performance of both descriptors combined. Finding the right balance might depend on the application and the likelihood of different subsets being experienced in the specific scenario under consideration.

References

[1] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74–93, 2001.

[2] R. M. Gray and A. D. Wyner, "Source coding for a simple network," *Bell System Technical Journal*, vol. 53, no. 9, pp. 1681–1721, 1974.

[3] J. K. Wolf, A. D. Wyner, and J. Ziv, "Source coding for multiple descriptions," *The Bell System Technical Journal*, vol. 59, no. 8, pp. 1417–1426, 1980.

[4] A. E. Gamal and T. Cover, "Achievable rates for multiple descriptions," *IEEE Transactions on Information Theory*, vol. 28, no. 6, pp. 851–857, 1982.

[5] V. N. Koshelev, "Hierarchical coding of discrete sources," *Problemy peredachi informatsii*, vol. 16, no. 3, pp. 31–49, 1980.

[6] W. H. R. Equitz and T. M. Cover, "Successive refinement of information," *IEEE Transactions on Information Theory*, vol. 37, no. 2, pp. 269–275, Mar. 1991.

[7] B. Rimoldi, "Successive refinement of information: characterization of the achievable rates," *IEEE Transactions on Information Theory*, vol. 40, no. 1, pp. 253–259, 1994.

[8] J. Nayak, E. Tuncel, D. Gunduz, and E. Erkip, "Successive refinement of vector sources under individual distortion criteria," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1769–1781, 2010.

[9] L. Ozarow, "On a source-coding problem with two channels and three receivers," *Bell System Technical Journal*, vol. 59, no. 10, pp. 1909–1921, 1980.

[10] V. A. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 821–834, 1993.

[11] V. A. Vaishampayan, N. J. A. Sloane, and S. D. Servetto, "Multiple-description vector quantization with lattice codebooks: design and analysis," *IEEE Transactions on Information Theory*, vol. 47, no. 5, pp. 1718–1734, 2001.

[12] Y. Frank-Dayan and R. Zamir, "Dithered lattice-based quantizers for multiple descriptions," *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 192–204, 2002.

[13] H. Jafarkhani and V. Tarokh, "Multiple description trellis-coded quantization," *IEEE Transactions on Communications*, vol. 47, no. 6, pp. 799–803, 1999.

[14] Y. Wang, M. T. Orchard, V. Vaishampayan, and A. R. Reibman, "Multiple description coding using pairwise correlating transforms," *IEEE Transactions on Image Processing*, vol. 10, no. 3, pp. 351–366, 2001.

[15] Y. Steinberg and N. Merhav, "On hierarchical joint source-channel coding with degraded side information," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 886–903, 2006.

[16] M. Gastpar, "To code or not to code," p. 170, 2002. [Online]. Available: http://infoscience.epfl.ch/record/33163

[17] E. Bourtsoulatze, D. Burth Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 3, pp. 567–579, Sep. 2019.

[18] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 4774–4778.

[19] D. Burth Kurka and D. Gündüz, "Joint source-channel coding of images with (not very) deep learning," in *Proc of International Zurich Seminar on Information and Communication (IZS 2020)*. ETH Zurich, 2020, pp. 90–94.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[21] N. Farsad, M. Rao, and A. Goldsmith, "Deep learning for joint source-channel coding of text," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018.

[22] D. Gunduz, "Joint source channel coding of information sources using neural networks," Aug. 2018, Patent GB2576702A;WO2020035684A1.

[23] K. Choi, K. Tatwawadi, T. Weissman, and S. Ermon, "NECST: neural joint source-channel coding," *CoRR*, vol. abs/1811.07557, 2018. [Online]. Available: http://arxiv.org/abs/1811.07557

[24] Y. M. Saidutta, A. Abdi, and F. Fekri, "M to 1 joint source-channel coding of gaussian sources via dichotomy of the input space based on deep learning," in *2019 Data Compression Conference (DCC)*, 2019, pp. 488–497.

[25] ——, "Joint source-channel coding for gaussian sources over awgn channels using variational autoencoders," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 1327–1331.

[26] Z. Xuan and K. Narayanan, "Analog joint source-channel coding for Gaussian sources over AWGN channels with deep learning," in *2020 International Conference on Signal Processing and Communications (SPCOM)*, 2020, pp. 1–5.

[27] L. Liu, A. Solomon, S. Salamatian, and M. Médard, "Neural network coding," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[28] M. B. Mashhadi, Q. Yang, and D. Gündüz, "CNN-based analog CSI feedback in FDD MIMO-OFDM systems," in *IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8579–8583.

[29] J. Shao and J. Zhang, "Bottlenet++: An end-to-end approach for feature compression in device-edge co-inference systems," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.

[30] M. Jankowski, D. Gunduz, and K. Mikolajczyk, "Wireless image retrieval at the edge," *CoRR*, vol. cs.IT/2007.10915, 2020. [Online]. Available: https://arxiv.org/abs/2007.10915

[31] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2016.

[32] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *Proc. Int. Conf. on Machine Learning (ICML)*, vol. 70, Aug. 2017, pp. 2922–2930.

[33] J. Ballé, D. Minnen, S. Singh, S. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2018.

[34] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 10 771–10 780.

[35] D. Minnen and S. Singh, "Channel-wise autoregressive entropy models for learned image compression," in *Int. Conf. on Image Compression (ICIP), 2020*, 2020.

[36] L. Zhao, H. Bai, A. Wang, and Y. Zhao, "Deep multiple description coding by learning scalar quantization," in *2019 Data Compression Conference (DCC)*, 2019, pp. 615–615.

[37] ——, "Multiple description convolutional neural networks for image compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 8, pp. 2494–2508, 2019.

[38] X. Lu, H. Wang, W. Dong, F. Wu, Z. Zheng, and G. Shi, "Learning a deep vector quantization network for image compression," *IEEE Access*, vol. 7, pp. 118 815–118 825, 2019.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv:1502.01852v1 [cs.CV]*, 2015.

[40] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.

[41] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[42] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[43] C. Shannon, "The zero error capacity of a noisy channel," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 8–19, Sep. 1956.

[44] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," *arXiv preprint arXiv:1511.06281*, 2015.

[45] D. B. Kurka and D. Gündüz, "DeepJSCC-f: Deep joint source-channel coding of images with feedback," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 178–193, May 2020.