Deep Joint Source-Channel Coding with Iterative Source Error Correction

Changwoo Lee
University of Michigan

Xiao Hu
University of Michigan

Hun-Seok Kim
University of Michigan

Abstract

In this paper, we propose an iterative source error correction (ISEC) decoding scheme for deep-learning-based joint source-channel coding (Deep JSCC). Given a noisy codeword received through the channel, we use a Deep JSCC encoder and decoder pair to update the codeword iteratively to find a (modified) maximum a-posteriori (MAP) solution. For efficient MAP decoding, we utilize a neural network-based denoiser to approximate the gradient of the log-prior density of the codeword space. Albeit the non-convexity of the optimization problem, our proposed scheme improves various distortion and perceptual quality metrics from the conventional one-shot (non-iterative) Deep JSCC decoding baseline. Furthermore, the proposed scheme produces more reliable source reconstruction results compared to the baseline when the channel noise characteristics do not match the ones used during training.

1 INTRODUCTION

Joint source-channel coding (JSCC) is a problem to find an encoder-decoder pair to transmit a compressible source reliably through a noisy channel. Deep-learning-based methods have brought advancement for (Deep) JSCC of images (Bourtsoulatze et al., 2019; Yang et al., 2021; Yang and Kim, 2022; Grover and Ermon, 2019; Choi et al., 2019) by *learning* a pair of encoder and decoder network models from data. Deep JSCC encoder and decoder pairs are parameterized by deep neural networks in the form of autoencoders (Kramer, 1991; Kingma and Welling, 2013). The latent variables at the output of the encoder is the transmitted codeword. The receiver uses a paired decoder to reconstruct the original source by decoding the noisy received codeword (i.e., the latent variables corrupted by the channel noise). The neural network encoder directly maps the source input to a codeword whose dimension is usually smaller than the input (i.e., signal compression), and the decoder network is trained to approximate the inverse mapping from the noisy codeword to the source.

In this paper, we consider an *asymmetric* communication case where the transmitter (e.g., an Internet-of-Things device) is *power-constrained* while the receiver (e.g., a powerful base station) enjoys *abundant resources*. The conventional Deep JSCC framework decodes a noisy codeword using the decoder network only once. This 'one-shot' decoding is efficient in terms of power and time complexity, but it does not provide optimal performance in practice. For example, (one-shot) Deep JSCC decoders often fail to produce better quality than conventional (not neural network-based) source and channel coding schemes when the noise power and compression ratio are low (Bourtsoulatze et al., 2019). Moreover, this suboptimality inevitably worsens when the channel noise characteristics shift from those of the noise model used for training.

To address these limitations of conventional Deep JSCC, we introduce an *iterative source error correction* (ISEC) scheme that iteratively decodes the received noisy codeword to gradually improve the reconstructed source quality. ISEC iteratively updates the codeword via gradient ascent by combining the *likelihood* of the current estimate based on the noise statistics and the *prior probability* of the codeword. We optimize/update the estimate from the noisy codeword to find the (local) minimum of a target loss using a Deep JSCC decoder. Accordingly, our proposed method approximately estimates a modified *maximum a-posteriori* (MAP) solution within the set of points around the observed codeword.

One can consider the Deep JSCC decoding as a non-linear inverse problem since the receiver only observes a non-linearly compressed, noisy version of the source. Thus obtaining an optimal solution requires a proper *prior* structure on the source domain for minimizing the negative likelihood of the source estimate based on the observed noisy codeword. To be specific, we use two types of the prior structure. We first adopt a Deep JSCC decoder network as an image prior, and modify the received noisy codeword by observing the reconstructed (and re-encoded) source at the

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

receiver to correct errors. Secondly, we impose an *implicit prior* to the codeword space because feeding an arbitrary input to a Deep JSCC decoder may produce meaningless image pixels due to the absence of a prior regularization during the training of Deep JSCC models (Grover and Ermon, 2019). Instead of explicitly modeling a prior density function which is difficult to obtain, we formulate ISEC to use the gradient of the log density approximated by a neural network *denoiser* inspired by (Kadkhodaie and Simoncelli, 2020).

We demonstrate that the proposed ISEC improves various distortion and perceptual quality metrics outperforming one-shot Deep JSCC decoding of low and high-resolution images in CIFAR-10 and Kodak datasets. The gain becomes more significant when the characteristics of the channel noise for the evaluation are different from those of the training environment. The contributions of this paper are as follows:

- We propose a new iterative source error correction (ISEC) decoding algorithm for modified MAP JSCC decoding of low- and high-resolution images.
- We formulate the ISEC MAP scheme to utilize the gradient of the prior distribution over the *codeword space* modeled by a bias-free denoiser network.
- We experimentally show that ISEC decoding outperforms the one-shot counterpart in various metrics, especially when the noise statistics mismatch those used during the training of the Deep JSCC encoder and decoder. We also analyze and quantify the impact of different terms in ISEC under various situations.[1]

2 BACKGROUND AND RELATED WORKS

2.1 Separate and Joint Source-Channel Coding

For a source with redundant information, a conventional separate source-channel coding (SSCC) approach transmits the source over a noisy channel using a separate source-dependent coding scheme (e.g., JPEG (Wallace, 1991) or better portable graphics (BPG) (Bellad, 2014)) followed by a source-independent channel coding scheme (e.g., low-density parity-check (LDPC) code (Gallager, 1962)). On the other hand, joint source-channel coding (JSCC) uses a single joint code for source encoding and channel coding.

2.2 Deep Joint Source-Channel Coding

Deep joint source-channel coding of images (Deep JSCC of images or shortly Deep JSCC) (Bourtsoulatze et al., 2019; Grover and Ermon, 2019) is a deep-learning-based approach of JSCC to transmit an image $\boldsymbol{x} \in \mathcal{X}$ ($\subset \mathbb{R}^m$) over a noisy channel and reconstruct the source at the receiver to minimize the error. Unlike SSCC schemes, there is no explicit boundary between source and channel (de)coding in Deep JSCC encoders (decoders).

Deep-JSCC training finds the parameters ϕ and θ of an encoder $E_\phi : \mathbb{R}^m \to \mathbb{R}^k$ and a decoder $D_\theta : \mathbb{R}^k \to \mathbb{R}^m$ modeled by deep neural networks. The received codeword $\boldsymbol{y}$ and the reconstructed source at the decoder output $\hat{\boldsymbol{x}}$ can be expressed by

$$\hat{\boldsymbol{x}} = D_\theta(\boldsymbol{y}), \quad \boldsymbol{y} = E_\phi(\boldsymbol{x}) + \boldsymbol{n},$$

where $\boldsymbol{n}$ is the additive channel noise that follows the distribution $P_\sigma(N)$ parameterized by σ. We assume $k < m$ (i.e., the encoder compresses the source to a smaller dimension), and without loss of generality, we further assume that the squared norm of codewords is bounded such that for all $\boldsymbol{x} \in \mathcal{X}$, $\|E_\phi(\boldsymbol{x})\|_2^2 \leq k$. The signal-to-noise ratio (SNR) in dB is defined by

$$SNR = 10 \log_{10} \frac{\|E_\phi(\boldsymbol{x})\|_2^2}{\|\boldsymbol{n}\|_2^2}.$$

We use the notation SNR_{train} and SNR_{test} to denote the SNR during the training and the test, respectively. *Channel Per Pixel* (CPP) indicates the inverted compression ratio

$$CPP = 0.5k/m,$$

where the factor of 0.5 is because of the complex-valued transmission of the codeword that allows transmitting *two* real-valued elements per channel-use as explained in the Deep JSCC literature (Bourtsoulatze et al., 2019; Yang et al., 2021; Yang and Kim, 2022; Kurka and Gündüz, 2020).

The pair (E_ϕ, D_θ) can be optimized/trained jointly with independent and identically distributed (i.i.d.) data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{d_{\text{train}}} \in \mathcal{X}$ and i.i.d. channel noise $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_{d_{\text{train}}} \sim P_{\sigma_{\text{train}}}(N)$:

$$(\phi, \theta) := \arg\min_{\phi,\theta} \frac{1}{d_{\text{train}}} \sum_{i=1}^{d_{\text{train}}} \|\boldsymbol{x}_i - D_\theta(E_\phi(\boldsymbol{x}_i) + \boldsymbol{n}_i)\|_2^2. \tag{1}$$

Due to the generalization ability of neural networks, Deep JSCC decoders can often provide a more reliable reconstruction of the image from the noisy codeword than separate source-channel coding (SSCC) schemes when the decoder faces unexpected/mismatched noise characteristics. It is also shown that under the additive white Gaussian noise with known noise variance, Deep JSCC outperforms SSCC schemes when SNR is low and the source compression ratio is high, whereas the gain diminishes in the opposite scenario (SNR is high and the compression ratio

[1] ISEC code is available at `https://github.com/changwoolee/isec-deep-jscc`

is low). Deep JSCC framework has been extended to binary symmetric and erasure channels (Choi et al., 2019), an SNR-adaptive framework (Ding et al., 2021), rate-adaptive framework (Yang and Kim, 2022), OFDM-guided framework (Yang et al., 2021), and feedback channels (Kurka and Gündüz, 2020). Also, Grover and Ermon (2019) discussed the Deep JSCC framework in the compressed sensing literature, focusing on reducing the number of measurements corrupted by mild noise. All prior Deep JSCC frameworks (except the schemes designed with feedback channels (Kurka and Gündüz, 2020)) employ a *one-shot decoding method* without iteration. Those one-shot Deep JSCC schemes can benefit from our proposed ISEC framework introduced in Section 3 to improve the quality of the reconstructed source by iterations at the receiver without modifying the transmitter or requesting retransmission. Since we assume that abundant power and computational resources (e.g., GPUs) are available at the receiver, improving performance is our primary objective, whereas the increased decoding complexity is a secondary concern in our framework.

2.3 Compressed Sensing using Generative Models

For compressed-sensing-based source compression and reconstruction (Donoho, 2006; Candes and Tao, 2006), deep generative models (DGMs) have emerged as a new type of prior for natural images. Compressed sensing using generative models (CSGM) employs generative adversarial networks (GANs) (Goodfellow et al., 2014) or variational autoencoders (VAEs) (Kingma and Welling, 2013), as first proposed by Bora et al. (2017). The intuition behind the approach is to recover the source $\boldsymbol{x}$ from a *linear* measurement $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}, \boldsymbol{A} \in \mathbb{R}^{k\times m}$, by searching over the range of a (possibly non-linear) generative model $G : \mathbb{R}^r \to \mathbb{R}^m$ which maps a simple distribution $P_{\mathcal{Z}}$ (e.g., isotropic Gaussian distribution) to the distribution of the sources $P_{\mathcal{X}}$. CSGM reconstructs $\hat{\boldsymbol{x}}_{\text{CSGM}}$ as follows:

$$\hat{\boldsymbol{z}}_{\text{CSGM}} := \underset{\boldsymbol{z}\in\mathcal{Z}}{\arg\min} \|\boldsymbol{y} - \boldsymbol{A}G(\boldsymbol{z})\|^2,$$
$$\hat{\boldsymbol{x}}_{\text{CSGM}} := G(\hat{\boldsymbol{z}}_{\text{CSGM}}),$$

where $\boldsymbol{z}$ is initialized randomly or to zero, then updated by a standard gradient descent method such as Adam (Kingma and Ba, 2014). Although the non-convexity of the optimization problem, CSGMs outperform conventional sparsity-based compressed sensing solvers on relatively simple source distributions such as human face images (Liu et al., 2015). Theoretical analysis and improvement have been reported by Joshi et al. (2021) whereas the techniques of inverting the generative models have been investigated in (Creswell and Bharath, 2018; Lei et al., 2019; Asim et al., 2020; Jalal et al., 2021).

Training DGMs on high-resolution natural images is a challenging task. Therefore, the direct application of CSGM to the Deep JSCC problem for natural images has been limited. Also, because the optimization process starts from random or zero-valued vectors as the GAN/DGM input, it requires thousands of iterations to converge to one of local optima(Bora et al., 2017). Although CSGM shares some similarities with our proposed ISEC method, there are a few main differences. 1) ISEC recovers the source from *non-linear* measurements $\boldsymbol{y} = E_\phi(\boldsymbol{x}) + \boldsymbol{n}$ to operate in lower SNR scenarios. 2) ISEC uses a Deep JSCC encoder and decoder as the DGM, trained with a highly complex latent distribution. 3) ISEC initializes the variable to $\boldsymbol{y}$, which is already close to the local minima, rather than a random or zero vector as in CSGM. 4) For a modified MAP solution, ISEC utilizes the approximate gradient of the *log prior density* of the codeword obtained by a denoiser neural network.

2.4 Gradient Estimation of Log Prior using Bias-free Denoiser

Our proposed decoding algorithm is built on the seminal work by Miyasawa (1961). Consider an observed point $\boldsymbol{r}$ of the original signal $\boldsymbol{s}$ corrupted by Gaussian noise with variance σ^2. Then, the gradient of the log density function $\nabla \log p(\boldsymbol{r})$ can be expressed with the minimum mean squared error (MMSE) estimator $\hat{\boldsymbol{s}}(\boldsymbol{r}) = \mathbb{E}[\boldsymbol{s}|\boldsymbol{r}] = \int \boldsymbol{s}p(\boldsymbol{s}|\boldsymbol{r})d\boldsymbol{s}$, satisfying

$$\nabla \log p(\boldsymbol{r}) = \frac{\hat{\boldsymbol{s}}(\boldsymbol{r}) - \boldsymbol{r}}{\sigma^2}. \tag{2}$$

Kadkhodaie and Simoncelli (2020) and Kawar et al. (2021) used a deep-neural-network-based MMSE estimator, where a DNN is trained to predict the noise instance $\boldsymbol{n}$ from the corrupted observation $\boldsymbol{r} = \boldsymbol{s} + \boldsymbol{n}$. A common choice for the DNN is a *bias-free* convolutional neural network due to its robustness to the unseen noise variance (Mohan et al., 2020). Our ISEC algorithm adopts a similar approach to formulate and solve a modified MAP optimization problem using a denoiser network. However, unlike prior works, we obtain the gradient of the log prior density in the *codeword* space.

3 PROPOSED METHOD

We introduce an iterative decoding algorithm, namely *iterative source error correction* (ISEC), to refine the initial / one-shot output of the Deep JSCC decoder $D_\theta(\boldsymbol{y})$. Given a Deep JSCC (Bourtsoulatze et al., 2019; Grover and Ermon, 2019) encoder E_ϕ and decoder D_θ pair, ISEC finds a modified MAP solution in the codeword space via an iterative process.

3.1 Assumptions

We suppose both the transmitter and receiver share the same (E_ϕ, D_θ) pair, and the receiver observes the k-

dimensional noisy codeword $\boldsymbol{y} = E_\phi(\boldsymbol{x}) + \boldsymbol{n}$ of the m-dimensional source (image) $\boldsymbol{x}$ transmitted through the noisy channel. Also, the trained decoder network D_θ is assumed to be deterministic and injective. We assume the channel noise follows isotropic Gaussian distribution $\boldsymbol{n} \sim p_\sigma(N) = \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$. We further assume that the receiver can estimate the true noise variance σ^2 after observing the noisy codeword so that the noise variance is available for our decoding algorithm (Algorithm 1). The transmitter may or may not have the true noise variance information prior to transmission. Moreover, we consider an *asymmetric* communication case where the transmitter is power- and resource-constrained while the receiver has abundant computational resources with an unlimited power constraint for iterative decoding.

3.2 Maximum A-Posteriori Inference for Deep JSCC

Given $\boldsymbol{y}$, our goal is to find a *maximum a-posteriori* (MAP) solution $\hat{\boldsymbol{z}}_{\text{MAP}}$ in the codeword (latent) space $\mathcal{Z} \subseteq \mathbb{R}^k$:

$$\hat{\boldsymbol{z}}_{\text{MAP}} := \arg\max_{\boldsymbol{z} \in \mathcal{Z}} \log p(\boldsymbol{z}|\boldsymbol{y}), \tag{3}$$

where a reconstructed source $\hat{\boldsymbol{x}}_{\text{MAP}}$ given $\hat{\boldsymbol{z}}_{\text{MAP}}$ is obtained by $\hat{\boldsymbol{x}}_{\text{MAP}} := D_\theta(\hat{\boldsymbol{z}}_{\text{MAP}})$. That is, we find the reconstruction in the range of the decoder network, which is our first *prior* structure in the *source* domain. Iterating the time index $t = 0, \ldots, T-1$, we update $\boldsymbol{z}_t$ by gradient ascent defined as

$$\boldsymbol{z}_{t+1} := \boldsymbol{z}_t + \eta_t \cdot \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z}_t|\boldsymbol{y}),$$

where $\eta_t > 0$ is a step size and $\boldsymbol{z}_0 := \boldsymbol{y}$. Unfortunately, the direct estimation of the gradient of the log posterior density is not available in general. Thus we consider the following form instead:

$$\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}|\boldsymbol{y}) = \nabla_{\boldsymbol{z}} \log p(\boldsymbol{y}|\boldsymbol{z}) + \nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}). \tag{4}$$

Let $\mathcal{R}(D_\theta) \subset \mathcal{X}$ be the range of D_θ. Since D_θ is deterministic and injective, one could write $p(\hat{\boldsymbol{x}}|\boldsymbol{z}) = \delta(\hat{\boldsymbol{x}} - D_\theta(\boldsymbol{z}))$ where $\delta(\cdot)$ is a Dirac delta function. Consider a marginalized conditional distribution $p(\boldsymbol{y}|\boldsymbol{z})$ over $\hat{\boldsymbol{x}} \in \mathcal{R}(D_\theta)$:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{z}) &= \int_{\mathcal{R}(D_\theta)} p(\boldsymbol{y}, \hat{\boldsymbol{x}}|\boldsymbol{z}) \mathrm{d}\hat{\boldsymbol{x}} \\ &= \int_{\mathcal{R}(D_\theta)} p(\boldsymbol{y}|\hat{\boldsymbol{x}}, \boldsymbol{z}) p(\hat{\boldsymbol{x}}|\boldsymbol{z}) \mathrm{d}\hat{\boldsymbol{x}} \\ &= \int_{\mathcal{R}(D_\theta)} p(\boldsymbol{y}|\hat{\boldsymbol{x}}, \boldsymbol{z}) \delta(\hat{\boldsymbol{x}} - D_\theta(\boldsymbol{z})) \mathrm{d}\hat{\boldsymbol{x}} \\ &= p(\boldsymbol{y}|D_\theta(\boldsymbol{z})) \\ &= p_\sigma(\boldsymbol{y} - E_\phi(D_\theta(\boldsymbol{z}))). \end{aligned} \tag{5}$$

The first term on the right-hand side of (4) is a likelihood of noise distribution. Using (5), it can be written in the following form:

$$\nabla_{\boldsymbol{z}} \log p(\boldsymbol{y}|\boldsymbol{z}) = -\nabla_{\boldsymbol{z}} \frac{1}{2\sigma^2} \|\boldsymbol{y} - E_\phi(D_\theta(\boldsymbol{z}))\|_2^2. \tag{6}$$

As we seek a MAP solution in the domain of the decoder D_θ, optimizing $\boldsymbol{z}$ using the likelihood term is equivalent to finding an optimal $\boldsymbol{x}$ in the range of D_θ. This approach shares a similarity with the CSGM framework (Bora et al., 2017).

Since $\boldsymbol{y}$ is corrupted by noise, the direction in (6) does not always lead to reducing the source error $\|\boldsymbol{x} - D_\theta(\boldsymbol{z})\|_2^2$. Moreover, no distributional or structural prior is given for the *codeword* (latent) space between the Deep JSCC encoder and the decoder. Unlike the latent space of DGMs in the CSGM scheme which can be described by a compact set such as a closed ball (Bora et al., 2017), the latent space for Deep JSCC networks has a complex structure. Therefore, instead of finding a solution starting from a random or zero-valued vector, we start from $\boldsymbol{z}_0 := \boldsymbol{y}$, using the decoder D_θ trained to reconstruct $\boldsymbol{x}$ which is one of the local optima. Then, through the iterative inference (test) stage, we further refine the codeword. In the proposed scheme, a proper choice of the regularizer is critical to keep $\boldsymbol{z}_t$ in the local region around the noiseless codeword $E_\phi(\boldsymbol{x})$, especially when the codeword $\boldsymbol{y}$ is corrupted by noise with lower SNR than trained.

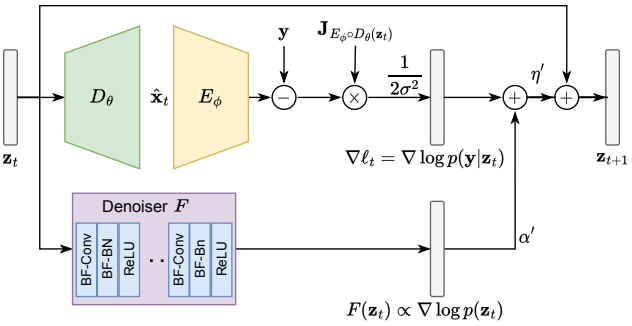


Figure 1: Illustration of ISEC at step t in (8). $\mathbf{J}_{E_\phi \circ D_\theta(\boldsymbol{z}_t)}$ refers to the Jacobian matrix of $E_\phi \circ D_\theta(\cdot) = E_\phi(D_\theta(\cdot))$ at $\boldsymbol{z}_t$.

3.3 Denoiser-based Regularizer for Modified MAP Inference

While a proper regularizer plays a critical role, the absence of a prior structure on the codeword space makes it difficult to find. This motivates us to consider an *implicit* prior over the codeword space. That is, instead of modeling the marginal distribution of the codeword space $p(\boldsymbol{z})$, we focus on estimating the *gradient* of the log density function $\nabla \log p(\boldsymbol{z})$. Given $\boldsymbol{z}_t$ at the t-th gradient ascent step, we know from (2) that the MMSE estimation $\hat{\boldsymbol{z}}(\boldsymbol{z}_t) = \mathbb{E}[\boldsymbol{z}|\boldsymbol{z}_t]$ can be used to approximate the gradient of the log prior

density of $\boldsymbol{z}_t$ by

$$\nabla \log p(\boldsymbol{z}_t) = \frac{\hat{\boldsymbol{z}}(\boldsymbol{z}_t) - \boldsymbol{z}_t}{\sigma_t^2} \approx \frac{F(\boldsymbol{z}_t)}{\sigma_t^2}, \tag{7}$$

where $F : \mathcal{Z} \to \mathcal{Z}$ is a *denoiser network* operates in the *codeword space*, and σ_t^2 is the variance of the current estimation $\hat{\boldsymbol{z}}(\boldsymbol{z}_t) - \boldsymbol{z}_t$. We train the denoiser network to predict the noise instance $\boldsymbol{n}$ from the corrupted codeword $\boldsymbol{y} = E_\phi(\boldsymbol{x}) + \boldsymbol{n}$. See A.2.2 for details on training.

To stabilize the optimization process, we propose a *modified* version of the MAP estimator based on the following expression with a parameter set $\{\alpha_t\}_{t=0}^{T-1}$:

$$\nabla_{\boldsymbol{z}_t} \log \tilde{p}_{\alpha_t}(\boldsymbol{z}_t|\boldsymbol{y}) := \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{y}|\boldsymbol{z}_t) + \alpha_t \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z}_t)$$

so that

$$\boldsymbol{z}_{t+1} := \boldsymbol{z}_t + \eta_t \cdot (\nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{y}|\boldsymbol{z}_t) + \alpha_t \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z}_t)).$$

By letting $\eta_t = \eta'$ and $\alpha_t = \alpha' \cdot \sigma_t^2$, we arrive at the expression for the t-th gradient ascent step:

$$\boldsymbol{z}_{t+1} := \boldsymbol{z}_t + \eta' \cdot \left(-\nabla \frac{1}{2\sigma^2} \|\boldsymbol{y} - E_\phi(D_\theta(\boldsymbol{z}_t))\|_2^2 + \alpha' F(\boldsymbol{z}_t) \right) \tag{8}$$

as illustrated in Figure 1.

Choice of hyperparameters Our preliminary experiments show that making hyperparameters adaptive to the SNR *mismatch* between the testing and the trained environment significantly improves the quality of reconstructions. Based on the trained noise variance σ_{train}^2, we adaptively scale the step size η' and the regularization parameter α' for the current noise variance σ^2. Given $\alpha > 0, \eta > 0$, and $\delta \geq 0$ which are the same for each model (ϕ, θ) trained for the same codeword dimension and SNR, we adjust the step size η and the contribution of the prior α' by

$$\alpha' := \alpha \cdot h(\sigma, \sigma_{\text{train}}, 2), \quad \eta' := \eta / h(\sigma, \sigma_{\text{train}}, \delta), \tag{9}$$

where

$$h(\sigma, \sigma_{\text{train}}, \delta) := \max\left(0.1, \left(\frac{\sigma^2}{\sigma_{\text{train}}^2}\right)^{\delta'}\right), \tag{10}$$

$$\delta' := \begin{cases} \delta & \text{if } \sigma^2 < \sigma_{\text{train}}^2 \\ 1 & \text{otherwise} \end{cases}. \tag{11}$$

For an extreme case where $\sigma^2 \to \infty$ and $\delta > 0$, $h(\sigma, \sigma_{\text{train}}, \delta) \to \infty$ thus $\alpha' \to \infty$ and $\eta' \to 0$. That is, the contribution of the likelihood to the gradient ascent becomes less significant and the step size decreases when the current noise variance (σ^2) is higher than the trained noise variance (σ_{train}^2). Algorithm 1 summarizes the overall procedure of the proposed ISEC MAP algorithm.

Algorithm 1 Iterative Source Error Correction

Input: $\boldsymbol{y}$, $\sigma > 0$, $\sigma_{\text{train}} > 0$ $\eta > 0$, $T > 0$, $\alpha > 0$, $\delta \geq 0$, E_ϕ, D_θ, F.
Output: $\hat{\boldsymbol{x}}$

$\alpha' \leftarrow \alpha \cdot h(\sigma, \sigma_{\text{train}}, 2)$
$\eta' \leftarrow \eta / h(\sigma, \sigma_{\text{train}}, \delta)$
$\boldsymbol{z}_0 \leftarrow \boldsymbol{y}$
for $t = 0, \ldots, T-1$ **do**
  $\ell_t \leftarrow -\frac{1}{2\sigma^2} \|\boldsymbol{y} - E_\phi(D_\theta(\boldsymbol{z}_t))\|_2^2$
  $\boldsymbol{d}_t \leftarrow F(\boldsymbol{z}_t)$
  $\boldsymbol{z}_{t+1} \leftarrow \boldsymbol{z}_t + \eta' \cdot (\nabla \ell_t + \alpha' \boldsymbol{d}_t)$
end for
$\hat{\boldsymbol{x}} \leftarrow D_\theta(\boldsymbol{z}_T)$
return $\hat{\boldsymbol{x}}, \boldsymbol{z}_T$

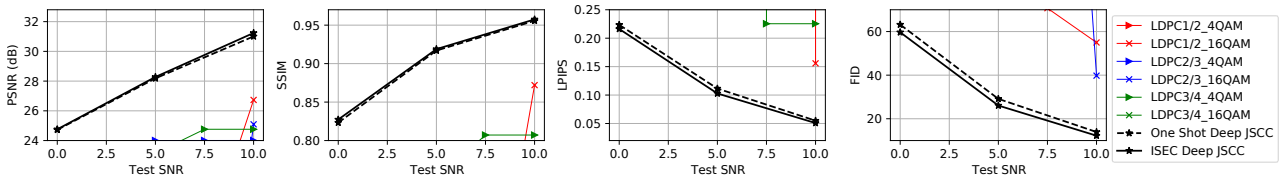
4 EXPERIMENTS

Dataset and Evaluation Metrics We conduct experiments to compare ISEC with one-shot decoding on a high resolution Kodak[2] dataset, which is composed of 24 768×512 pixel color images, as well as CIFAR-10 (Krizhevsky et al., 2009) low resolution image dataset, which contains $60,000$ 32×32 pixel color images. To quantify the distortion between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$, we choose peak signal-to-noise ratio (PSNR), $PSNR = 10 \cdot \log_{10} \frac{255^2}{MSE}$, $MSE = \frac{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2}{m}$, as well as structural similarity (SSIM, for CIFAR-10) (Wang et al., 2004), and multi-scale version of SSIM (MS-SSIM, for Kodak) (Wang et al., 2003). For the perceptual quality we use LPIPS (Zhang et al., 2018) distance and Fréchet inception distance (FID) (Heusel et al., 2017), which is based on the distance between features of images processed by a convolutional neural network (CNN). LPIPS compares the instance-wise difference of the deep features from VGGNet (Simonyan and Zisserman, 2014), and FID captures the distributional difference of deep features from InceptionV3 (Szegedy et al., 2016) network between the set of original source images and the set of reconstructed images.

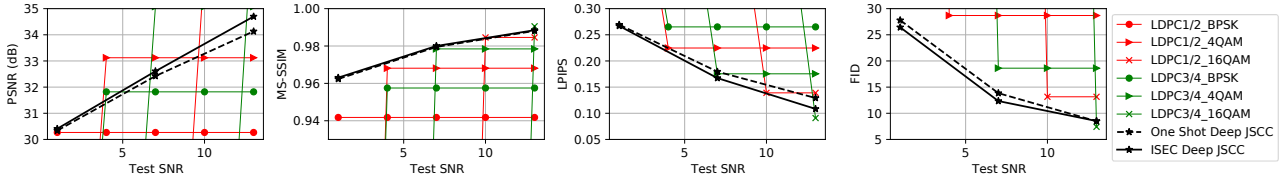
Deep JSCC Encoder, Decoder, and Denoiser Structures We parameterize the Deep JSCC encoders and decoders using fully-convolutional neural networks (CNNs) by following the approach in (Bourtsoulatze et al., 2019) which allows us to encode and decode an image regardless of the image size. Also, to ensure the injectivity of the decoder network, we use a parameterized rectified linear unit (PReLU) (He et al., 2015) as the activation function. Finally, the output of the encoder is normalized to have a maximum norm of $\sqrt{k}$ to satisfy the boundedness of the codeword. Specific configurations of the networks are presented in Appendix A.1.

We use a bias-free version of convolutional neural network (CNN) denoiser (Mohan et al., 2020) adopting the struc-

[2] http://r0k.us/graphics/kodak/



(a) CIFAR-10



(b) Kodak dataset

Figure 2: Performance of BPG-LDPC-QAM, One-shot and ISEC Deep JSCC on CPP=1/6 when $SNR_{\text{train}} = SNR_{\text{test}}$.

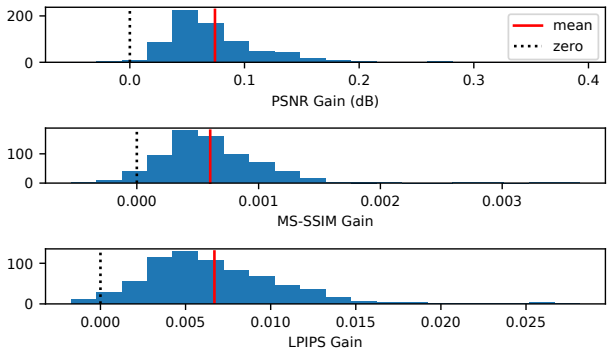


Figure 3: Histograms of ISEC gains for randomly cropped 256×256 Kodak images using 7dB model, CPP=1/16.

ture in (Kadkhodaie and Simoncelli, 2020) as specified in Appendix A.2.

Training and Testing We used Open Images V6 (Krasin et al., 2017) and the CIFAR-10 training dataset to train Deep JSCC encoders and decoders for evaluating Kodak and CIFAR-10 test datasets, respectively. We evaluate the low compression ratio models with CPP=1/6 for both CIFAR-10 and Kodak datasets, and high compression ratio models with CPP=1/12 and 1/16 for CIFAR-10 and Kodak datasets, respectively. We use the term an *n-dB model* to denote the Deep JSCC encoder and decoder pair trained with n-dB additive white Gaussian noise channel SNR. See A.1 and A.2.2 for details.

Parameters: We run ISEC for $T = 50$ iterations for all our experiments. For the Kodak dataset, we report the quality and distortion metrics by averaging metric values from 10 independent noise realizations for each image to reduce the variance caused by the random channel noise. Values of α, δ, and η used in the experiments are presented in Appendix A.4.

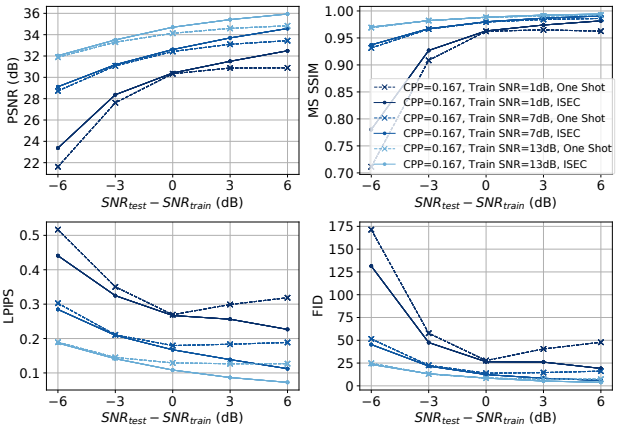
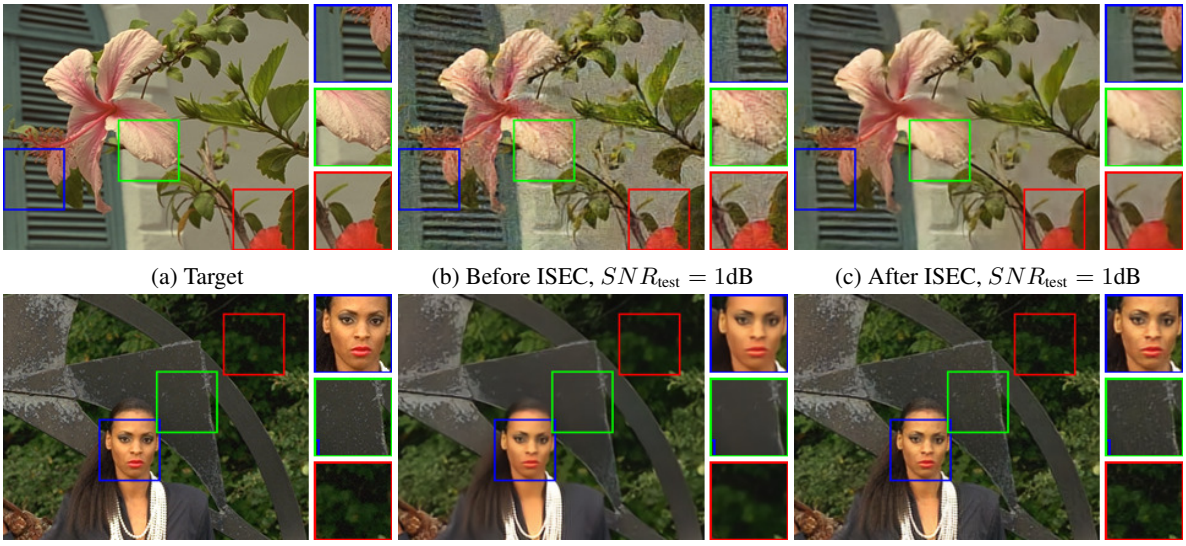


Figure 4: Reconstruction performance of ISEC on Kodak dataset when $SNR_{\text{test}} \neq SNR_{\text{train}}$, CPP=1/6. The x-axis indicates the SNR difference between the tested and trained environment.

4.1 Performance Under Trained Environment

We first examine our algorithm when the distribution of the channel noise is not different from that used during the training of the Deep JSCC model. The PSNR, SSIM, LPIPS, and FID metrics before and after ISEC for CIFAR-10 and Kodak images with CPP=1/6 are presented in Figure 2. Note that higher PSNR and SSIM indicate better quality while lower LPIPS and FID are desired. For the classical non-learning-based baseline, we include Better Portable Graphics (BPG) (Bellad, 2014) and low-density parity-check (LDPC) code (Gallager, 1962) with BPSK, 4-, or 16-QAM modulation. No encoding and decoding failure was observed when CPP=1/6 for the evaluation in Figure 2. BPG encoding or decoding failures can happen in CPP=1/12 or 1/16 settings. We fill the reconstruction with the mean of the input image when BPG encoding fails.



(a) Target (b) Before ISEC, $SNR_{\text{test}} = 1$dB (c) After ISEC, $SNR_{\text{test}} = 1$dB

(d) Target (e) Before ISEC, $SNR_{\text{test}} = 13$dB (f) After ISEC, $SNR_{\text{test}} = 13$dB

Figure 5: Before and after ISEC on Kodak dataset using SNR_{train}=7dB model on CPP=1/16. All images are cropped.

Similarly, we use a zero-valued image for reconstruction if BPG decoding fails.

For CIFAR-10 images, ISEC shows moderate gain in all 0, 5, and 10dB SNR cases compared to one-shot JSCC. We suppose the gain is moderate because the dimension of the images in the dataset is small so the JSCC encoder and decoder pair learns near-optimal coding and decoding for the trained SNR. On the contrary, the gains are more significant as shown in Figure 2b when ISEC is applied to the high-resolution Kodak image dataset.

As reported in (Bourtsoulatze et al., 2019), the conventional (separate) source-channel coding scheme (BPG-LDPC-QAM) outperforms Deep JSCC (and ISEC) schemes on PSNR (for high-resolution images such as Kodak). However, the perceptual metrics (MS-SSIM, LPIPS, and FID) are better with Deep JSCC and ISEC especially when SNR is low or moderate. The same trend was observed when CPP=1/16 as presented in Appendix.

Figure 3 shows the distribution of the gains from ISEC on the Kodak images evaluated with the 7dB model at CPP=1/16. Here we randomly sample 30 patches of 256×256 pixels from each Kodak image. We do not include FID for the histogram because it is measured on the entire set of images. Histograms show most patches exhibit a positive gain in all three metrics. The same tendency is observed from 1dB and 13dB models as well in Appendix.

4.2 Performance Under Mismatched SNRs

We evaluate ISEC when the test SNR is different from the trained SNR. It represents a practical scenario where the transmitter does not have the knowledge of the channel SNR which is estimated at the receiver after receiving/observing the noisy codeword. As shown in Figure 4, ISEC successfully handles mismatched test SNRs when they are higher or lower than the trained SNR when CPP=1/6. The results for CPP=1/16 are provided in Appendix. Note that, unlike Deep JSCC and ISEC, the conventional BPG-LDPC-QAM scheme experiences catastrophic decoding failures (image is not decodable) when the test/actual SNR is lower than targeted, whereas the receiver cannot improve the reconstruction quality when the test SNR is higher than targeted.

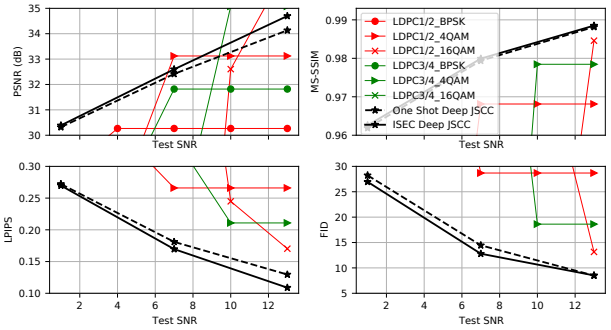


Figure 6: Average performance of Kodak images under additive white *Laplacian* noise with CPP=1/6 using the models trained with AWGN. ISEC generalizes well for the out-of-distribution noise.

We show visual comparisons in Figure 5 using example Kodak images (more examples in Appendix) when the test SNR is 6dB lower than the trained SNR. It is observed that a one-shot Deep JSCC decoder adds many artifacts whereas ISEC can *correct* some of them to improve the

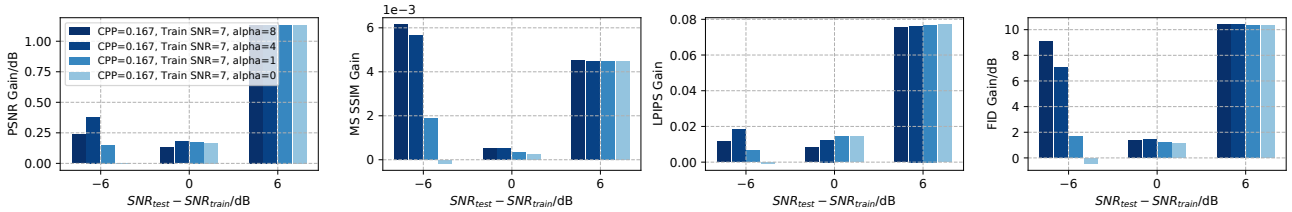


Figure 7: Quality metric gains observed from different $\alpha \in \{8, 4, 1, 0\}$.

quality. *Implicit prior* captured by the denoiser network plays a substantial role when the test SNR deteriorates from the training SNR, which is further analyzed in Section 4.4. Intuitively speaking, when $SNR_{\text{test}} < SNR_{\text{train}}$, the gradient of the log likelihood in (6) can be in the wrong direction due to the severe noise. However, this can be compensated by the denoiser network which estimates the gradient of the log prior predicting the (rough) direction toward the original codeword through the iterative update process.

It is worth noting that Figure 4 and 5 show that the reconstruction quality of the one-shot JSCC remains almost the same or even degrades producing blurry images when the test SNR is higher than training SNR. On the other hand, ISEC successfully recovers the details from the blurry initial (one-shot) reconstruction.

Although our framework uses a non-linear encoder ($\boldsymbol{y} = E_\phi(\boldsymbol{x}) + \boldsymbol{n}$) unlike linear measurements in CSGM ($\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$), the gain observed by ISEC in high SNRs is consistent with the claim in the CSGM literature (Bora et al., 2017, Thm. 1.2.) that the reconstruction error is proportional to the norm of the observation noise and the number of measurements. However, for general rate-quality tradeoffs, it is known that Deep JSCC (even without ISEC) outperforms CSGM which relies on a linear encoder $\boldsymbol{A}$ (Grover and Ermon, 2019). In summary, ISEC enhances the robustness of Deep JSCC against unexpected downward SNR whereas the reconstruction quality improves by ISEC when the actual SNR is higher than expected. The same benefits are unattainable in conventional separate source-channel coding schemes.

4.3 Performance in Additive White Laplace Noise

In this section, we evaluate the case where the noise distribution is shifted from the trained. We test the Kodak dataset in an *additive white Laplace noise* (AWLN) channel with CPP=1/6 that has the same variance as the Gaussian noise (AWGN) channel used during the training of Deep JSCC models (E_ϕ, D_θ) and the denoiser network F. The same decoding algorithm and parameter selection rules specified in Algorithm 1 are applied to evaluate the AWLN case. Figure 6 shows that ISEC achieves performance improvement from one-shot decoding in this noise distribution mismatch scenario. Despite the same SNR, the conventional

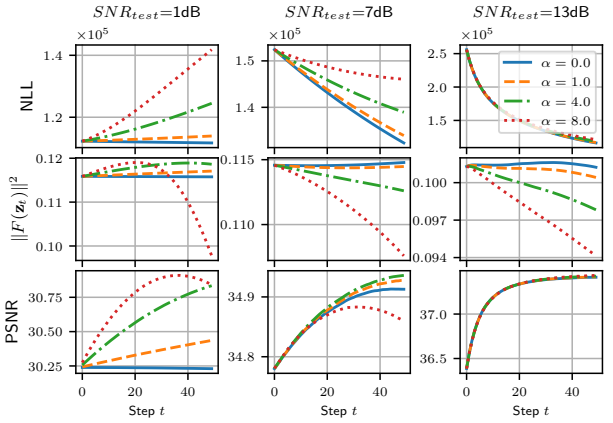


Figure 8: ISEC objectives over steps using the 7dB Deep JSCC model with CPP=1/6 on `kodim06`.

BPG-LDPC-QAM scheme undergoes performance degradation under the AWLN channels compared to the AWGN case, showing worse perceptual quality metrics (MS-SSIM, LPIPS, and FID) than the Deep JSCC one-shot decoding and ISEC. On the other hand, after applying ISEC, all evaluation criteria are improved from the one-shot decoding despite being performed under white Laplace distribution. Note that neither transmitters nor receivers are aware of the distributional shift of the channel noise at any stage. This shows that our proposed decoding algorithm generalizes well and enhances the robustness of the Deep JSCC scheme under the unseen channel noise distribution.

4.4 Role of Implicit Prior

We study the role of the implicit prior in ISEC with varying α in (8) and (9) which scales the output of the denoiser to approximate the gradient of the log prior density of the codeword. A larger α makes each step rely more on the prior information. We tested the 7dB model with CPP=1/6 in 1, 7, and 13dB SNRs with $\alpha \in \{0, 1, 4, 8\}$ for 3 different test SNRs. Figure 7 summarizes the results. When the test SNR is 13dB, the α does not affect the performance much. However, higher gains are achieved by using a proper parameter α when the test SNR is the same or lower than the training SNR. It indicates that the approximated gradient of log prior density plays a key role when the observed SNR

is lower than that of the training environment.

We also show in Figure 8 how the negative likelihood (NLL) $\frac{1}{2\sigma^2}\|\boldsymbol{y} - E_\phi(D_\theta(\boldsymbol{z}_t))\|_2^2$, the squared norm of the denoiser output $\|F(\boldsymbol{z}_t)\|_2^2$, and the PSNR of the reconstruction change along the ISEC step $t = 0, \ldots, 49$ with different α's. When the test SNR is 1dB, notice the PSNR does not improve by ISEC with $\alpha = 0$ and 1, whereas when $\alpha = 4$, all metrics improve from one-shot decoding through ISEC iterations. We also note that when $\alpha = 4$ or 8, the NLL increases from the starting point as PSNR enhances while the squared norm of the denoiser rapidly decreases (i.e., low $\|\nabla \log p(\boldsymbol{z}_t)\|^2$). This shows that the denoiser with the proper choice of α can deal with noise variance higher than the trained environment because of the bias-free structure. When $\alpha = 8$, however, the prior information dominates the likelihood too much so that the PSNR drops after a few tens of iterations. As the test SNR increases to 13dB, the impact of the denoiser becomes negligible and the likelihood dominates.

5 CONCLUSION

We introduced ISEC for Deep JSCC for the scenario employing a power-limited transmitter and a resource-abundant receiver. Our method extends the Deep JSCC decoding in an iterative way by correcting the source error in the codeword space. We introduce a modified MAP decoding algorithm with an approximated gradient of the log density using a bias-free denoiser network. Through iterations, the missed information from the one-shot decoding can be recovered by ISEC to improve various quality measures, especially when the observed channel noise characteristics are mismatched from the training environment.

Limitations and Future Work. Although we assume the receiver has abundant computing resources unlike the resource-constrained transmitter, the iterative decoding in ISEC increases the computational cost and power consumption compared to the one-shot decoding. Reducing the number of iterations to lower the complexity of ISEC is a possible extension of this work. Applying and evaluating ISEC to different source types such as audio and sensor data is left as future work.

Acknowledgments

This research was supported in part by NSF CAREER Award #1942806, and also through computational resources and services provided by Advanced Research Computing at the University of Michigan, Ann Arbor.

References

Muhammad Asim, Max Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pages 399–409. PMLR, 2020.

Fabrice Bellad. Bpg image format. 2014. URL `https://bellard.org/bpg/`.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546. PMLR, 2017.

Eirina Bourtsoulatze, David Burth Kurka, and Deniz Gündüz. Deep joint source-channel coding for wireless image transmission. *IEEE Transactions on Cognitive Communications and Networking*, 5(3):567–579, 2019.

Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52 (12):5406–5425, 2006.

Kristy Choi, Kedar Tatwawadi, Aditya Grover, Tsachy Weissman, and Stefano Ermon. Neural joint source-channel coding. In *International Conference on Machine Learning*, pages 1182–1192. PMLR, 2019.

Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. *IEEE transactions on neural networks and learning systems*, 30(7):1967–1974, 2018.

Mingze Ding, Jiahui Li, Mengyao Ma, and Xiaopeng Fan. Snr-adaptive deep joint source-channel coding for wireless image transmission. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1555–1559. IEEE, 2021.

David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Aditya Grover and Stefano Ermon. Uncertainty autoencoders: Learning compressed representations via variational information maximization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2514–2524. PMLR, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

Ajil Jalal, Sushrut Karmalkar, Alex Dimakis, and Eric Price. Instance-optimal compressed sensing via posterior sampling. In *International Conference on Machine Learning*, pages 4709–4720. PMLR, 2021.

Babhru Joshi, Xiaowei Li, Yaniv Plan, and Ozgur Yilmaz. Plugin: A simple algorithm for inverting generative models with recovery guarantees. *Advances in Neural Information Processing Systems*, 34:24719–24729, 2021.

Zahra Kadkhodaie and Eero P Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser. *arXiv preprint arXiv:2007.13640*, 2020.

Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. *Advances in Neural Information Processing Systems*, 34:21757–21769, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013.

Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

David Burth Kurka and Deniz Gündüz. Deepjscc-f: Deep joint source-channel coding of images with feedback. *IEEE Journal on Selected Areas in Information Theory*, 1(1):178–193, 2020.

Qi Lei, Ajil Jalal, Inderjit S Dhillon, and Alexandros G Dimakis. Inverting deep generative models, one layer at a time. *Advances in neural information processing systems*, 32, 2019.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Koichi Miyasawa. An empirical bayes estimator of the mean of a normal population. *Bull. Inst. Internat. Statistics*, 1:181–188, 1961.

Sreyas Mohan, Zahra Kadkhodaie, Eero P. Simoncelli, and Carlos Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlSmC4FPS.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Gregory K. Wallace. The jpeg still picture compression standard. *Commun. ACM*, 34(4):30–44, apr 1991. ISSN 0001-0782. doi: 10.1145/103085.103089. URL https://doi.org/10.1145/103085.103089.

Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. URL http://dblp.uni-trier.de/db/journals/tip/tip13.html#WangBSS04.

Mingyu Yang and Hun-Seok Kim. Deep joint source-channel coding for wireless image transmission with adaptive rate control. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5193–5197. IEEE, 2022.

Mingyu Yang, Chenghong Bian, and Hun-Seok Kim. Deep joint source channel coding for wireless image transmission with ofdm. In *IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

A Network Configurations, Training, and Test Details

This Appendix section presents the configurations and training methods of deep neural networks used in our paper. The parameters of ISEC used in Section 4 are also specified.

A.1 Deep JSCC Encoder and Decoder

A.1.1 Configuration

Deep JSCC encoders and decoders are convolutional neural networks (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; He et al., 2016). In both networks, batch normalization (Ioffe and Szegedy, 2015) is used between each convolution layer and a non-linear activation. Specifically, we use the following parameterized rectified linear unit (PReLU) (He et al., 2015) as the activation function

$$\mathsf{PReLU}(u) = \begin{cases} u & u \geq 0 \\ \rho \cdot u & u < 0 \end{cases}$$

where $\rho \geq 0$ is a trainable parameter shared within the layer. We use convolution layers with stride 2 for downsampling, and bilinear upsampling with a factor of 2 for upsampling the width and height of the tensor at the encoder and decoder, respectively. Hence the encoder network downsamples the input image with size $m = H \times W \times 3$ into $k = \frac{HWC}{4\times 4}$, and the decoder network maps the compressed (noisy) codeword back into the original pixel domain. Also, the power normalization layer is added at the end of the encoder to satisfy the power constraint $\|E_\phi(\boldsymbol{x})\|_2^2 \leq k$:

$$\text{PowerNorm}(\boldsymbol{v}, k) = \begin{cases} \boldsymbol{v} & \text{if } \|\boldsymbol{v}\|_2^2 \leq k \\ \sqrt{k}\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2} & \text{otherwise} \end{cases}.$$

Residual blocks (He et al., 2016) depicted in Figure 9a are added to the network for the overall configuration specified in Table 1 and 2 for CIFAR-10 and Kodak datasets, respectively.

A.1.2 Training Method

The weights and biases of Deep JSCC encoders and decoders are updated by minimizing the mean squared error (i.e., (1)) between the input $\boldsymbol{x}$ and the reconstruction from the noisy codeword $D_\theta(E_\phi(\boldsymbol{x}) + \boldsymbol{n})$, where $\boldsymbol{n}$ is a channel noise.

For low-resolution images, Deep JSCC encoders and decoders are trained with CIFAR-10 (Krizhevsky et al., 2009) training set, which contains $50,000$ 32×32 color images of 10 different classes. We use the batch size of 64 for 234,300 steps with Adam (Kingma and Ba, 2014) optimizer with parameter $\beta_1 = 0.0$ and $\beta_2 = 0.9$. The learning rate was set to 0.0002 and reduced to 0.00002 at the 117,150th step.

For high-resolution images, we use a subset of 273,230 images from Open Images V6 (Krasin et al., 2017) training dataset[3]. The images are randomly cropped to 128×128 pixel patches. We use the batch size of 128 and train the model for 96,030 steps with Adam (Kingma and Ba, 2014) optimizer with parameter $\beta_1 = 0.0$ and $\beta_2 = 0.9$, where the learning rate is set to 0.001 and reduced to 0.0001 at the 64,020th step.

We set CPP=1/6 or 1/12 for low-resolution images, and CPP=1/6 or 1/16 for high-resolution images. For each CPP evaluation, we train three encoder-decoder pairs for three channel SNR cases of 0, 5, and 10 dB for low-resolution images. Similarly, three pairs are trained for 1, 7, and 13 dB SNR for high-resolution images per each CPP. We use the term *n-dB model* to denote the Deep JSCC encoder and decoder pair trained with n-dB channel noise. All networks are trained on a 2.9 GHz Intel Xeon Gold 6226R processor and one NVIDIA A40 GPU.

A.2 Bias-Free Denoiser

A.2.1 Configuration

For the codeword (latent) denoiser, we use a bias-free denoiser (Mohan et al., 2020). Specifically, we use 20 Convolution-Batch Normalization-ReLU blocks *without any additive bias terms* based on the structure in (Kadkhodaie and Simoncelli,

[3]We use the first two tarball file (`train_0.tar.gz` and `train_1.tar.gz`) of the training set from the following website: `https://github.com/cvdfoundation/open-images-dataset`

2020). As illustrated in Figure 9b, the bias-free convolution layers are composed of $3 \times 3 \times 64$ kernels except for the last layer.

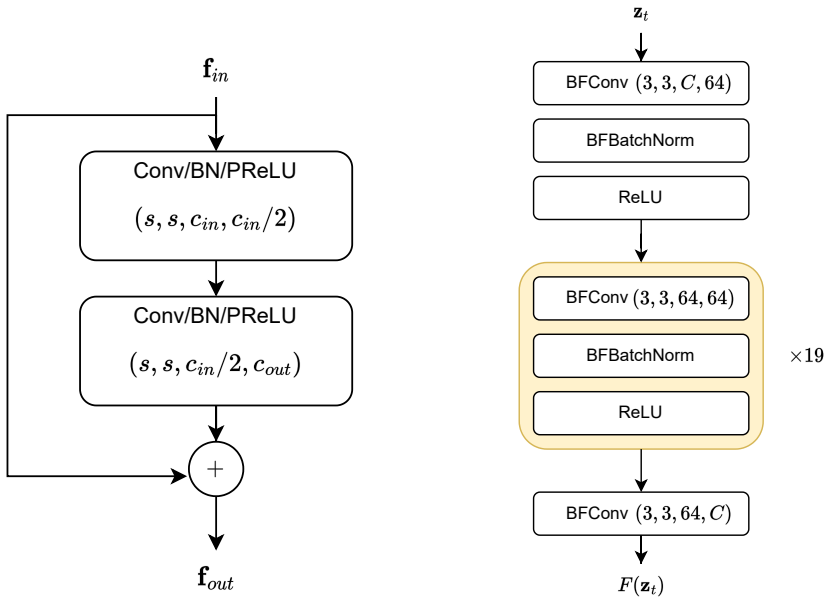
A.2.2 Training Method

Bias-free denoisers used in this paper operate in the *codeword* domain. Thus we trained each denoiser (there are multiple versions with different latent/codeword sizes) to predict the *channel noise instance* from a corrupted codeword. That is, given $\boldsymbol{x}_i$ in the training set we first map $\boldsymbol{x}_i$ to the codeword $E_\phi(\boldsymbol{x}_i)$, where $E_\phi : \mathbb{R}^m \to \mathbb{R}^k$ is a Deep JSCC encoder network. Then we sample a k-dimensional channel noise following the predefined noise distribution $\boldsymbol{n}_i \sim P_\sigma(N)$ which does not change during the training. Then, we update the weights of the denoiser network $F' : \mathbb{R}^k \to \mathbb{R}^k$ to minimize the mean squared error between the noise instance $\boldsymbol{n}_i$ and the output of the denoiser network $F'(E_\phi(\boldsymbol{x}_i) + \boldsymbol{n}_i)$ given the corrupted codeword $E_\phi(\boldsymbol{x}_i) + \boldsymbol{n}_i$:

$$F := \arg\min_{F'} \frac{1}{d_{\text{train}}} \sum_{i=1}^{d_{\text{train}}} \|\boldsymbol{n}_i - F'(E_\phi(\boldsymbol{x}_i) + \boldsymbol{n}_i)\|_2^2.$$

The bias-free denoisers for CIFAR-10 are trained using codewords of the CIFAR-10 training data. The models are trained for 234,300 steps with batch size of 64 using Adam optimizers with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 0.0002 and decayed to 0.00002 at the 117,150th step.

The bias-free denoisers for Kodak experiments are trained with codewords of randomly cropped 128×128 pixel patches in the subset of the Open Images V6 training dataset used for Deep JSCC model training. We trained each bias-free denoiser for 53,350 steps with batch size of 128 using Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We kept the learning rate policy the same as the CIFAR-10 denoiser training.



(a) (b)

Figure 9: (a): Residual Block used in Deep JSCC encoders and decoders. The elements of the tuple (s, s, c, d) denote the width and height of the kernel, and the numbers of input and output channels of the convolution layer, respectively. If $c_{in} \neq c_{out}$, a 1×1 convolution layer is applied to the shortcut. (b): Bias-free Denoiser.

A.3 Data pre- and post-processing

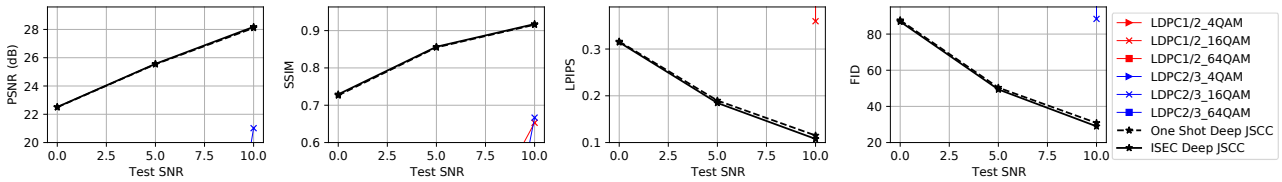
All image pixels are normalized from the integer values $[0, 255]$ to floating point values $[-1, 1]$ before being fed into the network. During testing, we use the output of the decoder in the range of $[-1, 1]$ directly to evaluate LPIPS and FID.

Type	Kernel Size (s)	Stride↓, Upscale↑	Output Shape ($C_h \times H \times W$)
Encoder			
Conv/BN/PReLU	7	1	(32, 32, 32)
Conv/BN/PReLU	5	2↓	(64, 16, 16)
Residual	3	1	(64, 16, 16)
Conv/BN/PReLU	5	2↓	(128, 8, 8)
Residual	3	1	(128, 8, 8)
Residual	3	1	(128, 8, 8)
Residual	3	1	(128, 8, 8)
Residual	3	1	(k/64, 8, 8)
Power Normalization	-	-	(k/64, 8, 8)
Decoder			
Conv/BN/PReLU	3	1	(128, 8, 8)
Residual	3	1	(128, 8, 8)
Residual	3	1	(128, 8, 8)
Upsample	-	2↑	(128, 16, 16)
Conv/BN/PReLU	3	1	(64, 16, 16)
Residual	3	1	(64, 16, 16)
Upsample	-	2↑	(64, 32, 32)
Conv/BN/PReLU	3	1	(32, 32, 32)
Residual	3	1	(32, 32, 32)
Conv/BN/Tanh	5	1	(3, 32, 32)

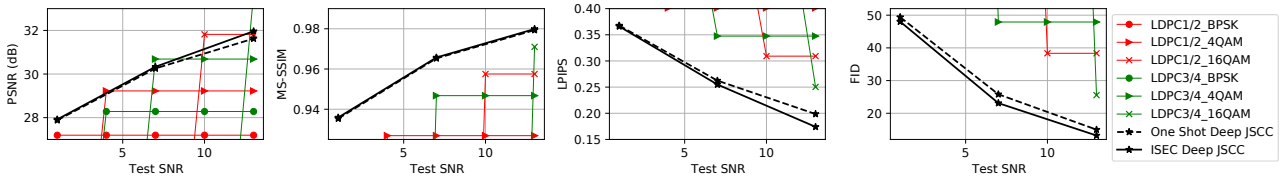
Table 1: Deep JSCC Encoder and Decoder structure for CIFAR-10. Kernel size: the width and height of the kernel of the convolution layer, C_h: number of hidden channels of the network, k: dimension of the codeword space.

Type	Kernel Size (s)	Stride↓, Upscale↑	Output Shape ($C_h \times H \times W$)
Encoder			
Conv/BN/PReLU	7	2↓	$(128, H/2, W/2)$
Conv/BN/PReLU	5	2↓	$(128, H/4, W/4)$
Residual	5	1	$(128, H/4, W/4)$
Residual	5	1	$(128, H/4, W/4)$
Residual	5	1	$(128, H/4, W/4)$
Conv/BN	5	1	$(\frac{16k}{HW}, H/4, W/4)$
Power Normalization	-	-	$(\frac{16k}{HW}, H/4, W/4)$
Decoder			
Conv/BN/PReLU	5	1	$(128, H/4, W/4)$
Residual	5	1	$(128, H/4, W/4)$
Residual	5	1	$(128, H/4, W/4)$
Upsample	-	2↑	$(128, H/2, W/2)$
Conv/BN/PReLU	5	1	$(128, H/2, W/2)$
Residual	5	1	$(128, H/2, W/2)$
Upsample	-	2↑	$(128, H, W)$
Conv/BN/Tanh	7	1	$(3, H, W)$

Table 2: Deep JSCC Encoder and Decoder structure for Kodak dataset. C_h: number of hidden channels of the network, k: dimension of the codeword space. For *Upsample*, we used bilinear upsampling.



(a) CIFAR-10, CPP=1/12



(b) Kodak dataset, CPP=1/16

Figure 10: Performance of BPG-LDPC-QAM, One-shot and ISEC Deep JSCC when $SNR_{\text{train}} = SNR_{\text{test}}$.

The decoder outputs are re-mapped to the original integer range of $[0, 255]$ for the evaluation of PSNR and (MS-)SSIM. Random horizontal flipped images were created to augment the dataset during training.

A.4 Parameters for ISEC

In this section, we specify the parameters of Algorithm 1 used in our experiments. Note that α and δ are used for calculating α' in Algorithm 1, which controls the effect of the gradient of implicit prior (i.e., (7)), whereas η is a step size of ISEC. Throughout CIFAR-10 evaluations, α and δ are fixed to 1.0. Regardless of CPP, we use $\eta = 0.004$, 0.002, and 0.001 for 0, 5, and 10dB models, respectively. For the Kodak dataset, we set $\alpha = 2$ for 1dB models and $\alpha = 4$ for 7 and 13dB models. When CPP=1/6, the parameter pair (δ, η) is set to $(0.5, 0.001)$, $(1.0, 0.001)$, and $(2.0, 0.005)$ for the 1, 7, and 13dB models, respectively. When CPP=1/16, we use $(0.0, 0.001)$, $(0.0, 0.0005)$, and $(2.0, 0.005)$ for the 1, 7, and 13dB models, respectively. In particular, we use $(\delta, \eta) = (2.0, 0.001)$ when test SNRs are 16dB and 19dB since the step size grows abnormally large otherwise.

A.5 LDPC codes setting

For all of our LDPC experiments on the separate source-channel coding scheme, we used the IEEE 802.11 WiFi standard with a block length of 648 and a submatrix size of 27. The LDPC codes are decoded using the belief propagation algorithm with 10 iterations.

B Additional Quantitative Results

B.1 Performance of Low Channel-per-pixel models Under Trained Environment.

In Figure 10a and 10b, we present the performance of one-shot vs. ISEC Deep JSCC for the CIFAR-10 dataset at CPP=1/12 and the Kodak dataset at CPP=1/16, respectively. Compared to the performance when CPP=1/6 in Figure 2, ISEC and one-shot Deep JSCC decoding methods outperform BPG-LDPC-QAM with larger gaps in perceptual quality metrics. The one-shot decoding and ISEC are tested under the same channel characteristics as the training. ISEC increases all performance metrics in lower CPP cases although the observed gains from ISEC are smaller than those from higher CPP cases.

B.2 Mismatched Noise Characteristics

The performance curves when the training and testing SNR mismatch ranges from -5dB to 10dB are shown in Figure 11. We also present visualization of the first 36 images of the CIFAR-10 test dataset in Figure 14. Since the codeword dimension of CIFAR-10 images is much smaller than that of Kodak images, the error recovery via ISEC in lower SNR environments is not significant (note there might be room to improve since we did not optimize the ISEC parameters α, δ,

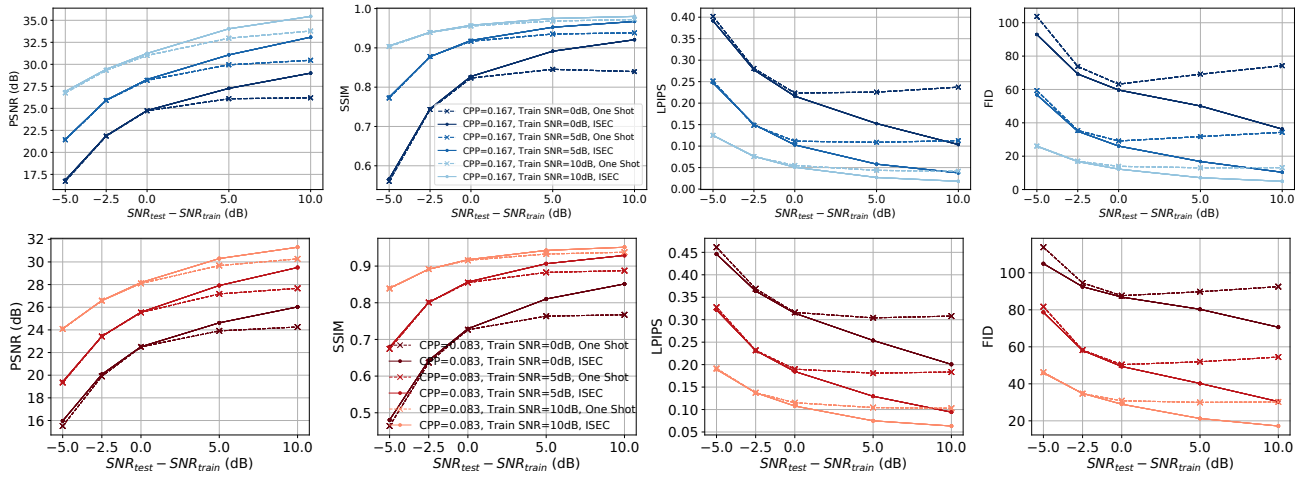


Figure 11: Reconstruction Performance of ISEC on CIFAR-10 dataset when $SNR_{\text{test}} \neq SNR_{\text{train}}$. Top row: CPP=1/6, bottom row: CPP=1/12.

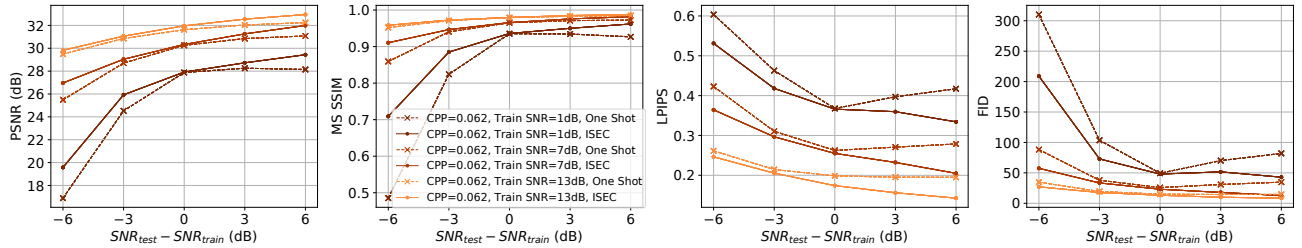


Figure 12: Reconstruction Performance of ISEC on Kodak dataset when $SNR_{\text{test}} \neq SNR_{\text{train}}$, CPP=1/16. The x-axis indicates the SNR difference between tested and trained environment.

and η). The suboptimal performance of one-shot decoding when the test SNR is higher than the training SNR is observed in Figure 11 and 14h. ISEC provides significant gains in such scenarios.

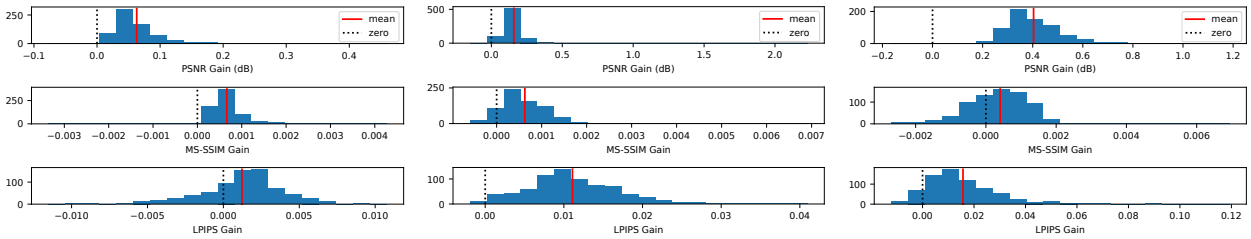
Figure 12 shows the performance gain on Kodak dataset when $SNR_{\text{test}} \neq SNR_{\text{train}}$, CPP=1/16. The same trend is observed as when CPP=1/6 (see Figure 4).

B.3 Distributions of Gains from ISEC for Kodak images

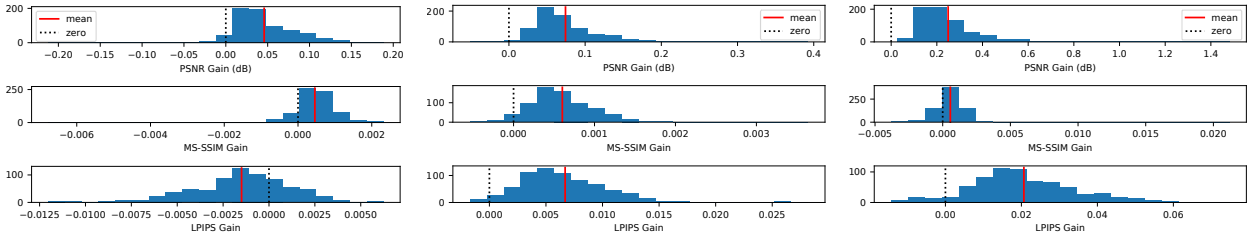
In this section, we show distributions of the gain from ISEC for three different test SNRs (1, 7, and 13 dB) at CPP=1/6 and 1/12 using the 7dB model. It supplements the analysis shown in Figure 3. Figure 13 shows histograms of PSNR, MS-SSIM, and LPIPS gains from ISEC for randomly cropped 256×256 pixel Kodak image patches. We noticed ISEC does not provide LPIPS gains when CPP=1/16 and SNR=1dB (-6dB lower than the training SNR). We presume the ISEC gain diminishes for this challenging scenario (compression rate is high at CPP=1/16 and SNR is much lower than expected) because the quality of Deep JSCC one-shot decoding degrades considerably so that the starting point of ISEC is far away from the local optima. Nevertheless, the average gains of all other metrics are above zero without a significant number of outliers exhibiting negative gains.

C Additional Qualitative/Visual Comparisons

Figure 14 shows additional CIFAR-10 images decoded by one-shot decoding and ISEC based on the 5dB Deep JSCC model with CPP=1/6. Additional Kodak dataset images for one-shot decoding and ISEC results are shown in the following figures: Figure 15, 16, and 17 are at CPP=1/6 for 1, 7, and 13dB models, respectively. Figure 18, 19, and 20 are outputs from 1, 7, and 13dB models, respectively, at CPP=1/16.

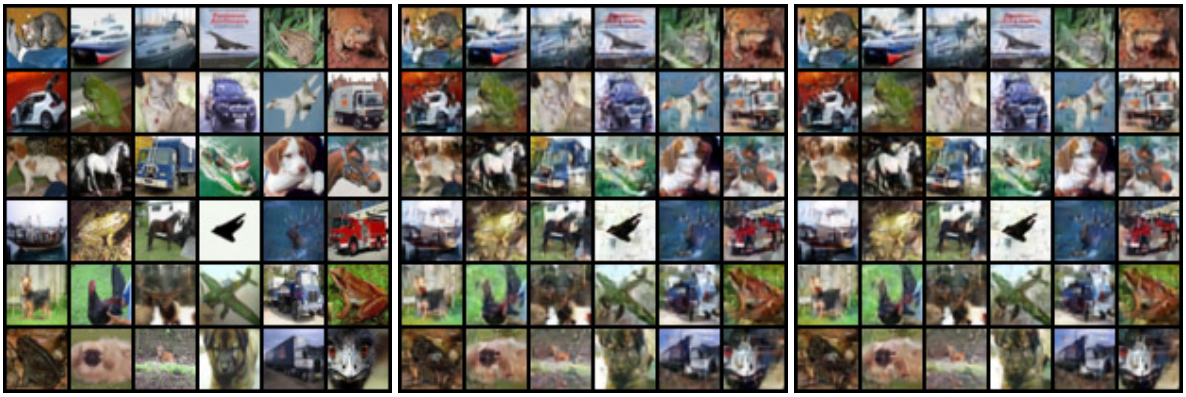


(a) CPP=1/6, SNR_{test}=1dB (b) CPP=1/6, SNR_{test}=7dB (c) CPP=1/6, SNR_{test}=13dB

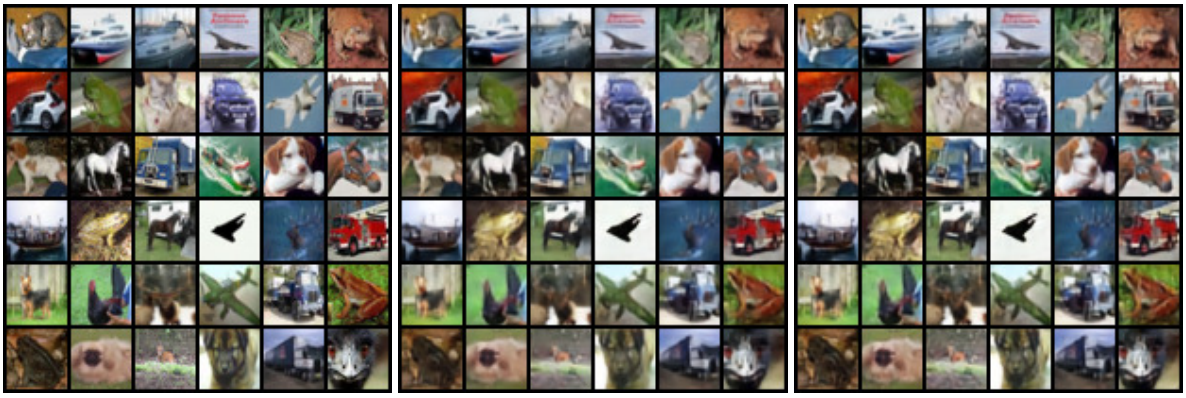


(d) CPP=1/16, SNR_{test}=1dB (e) CPP=1/16, SNR_{test}=7dB (f) CPP=1/16, SNR_{test}=13dB

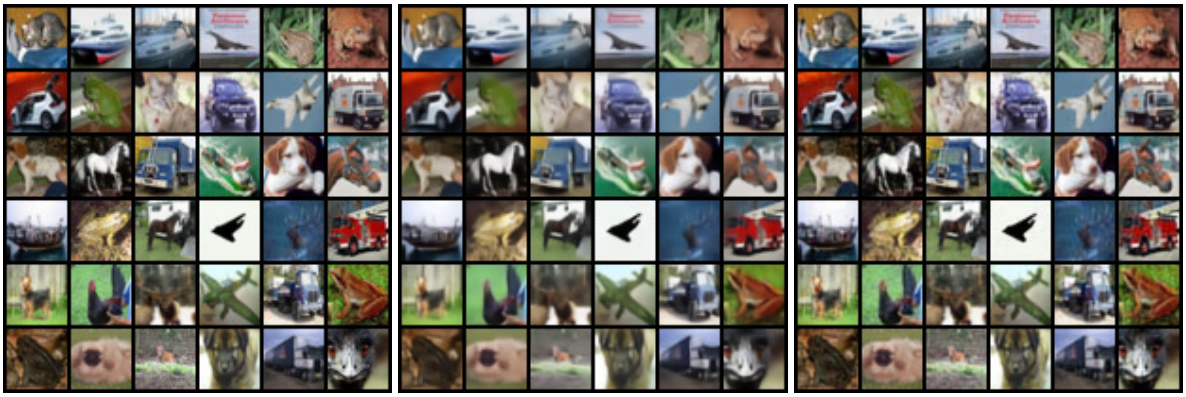
Figure 13: Histograms of ISEC gains for randomly cropped 256×256 Kodak images using $SNR_{\text{train}} = 7$dB model.



(a) Target (b) One-shot, $SNR_{\text{test}} = 0$dB (c) ISEC, $SNR_{\text{test}} = 0$dB

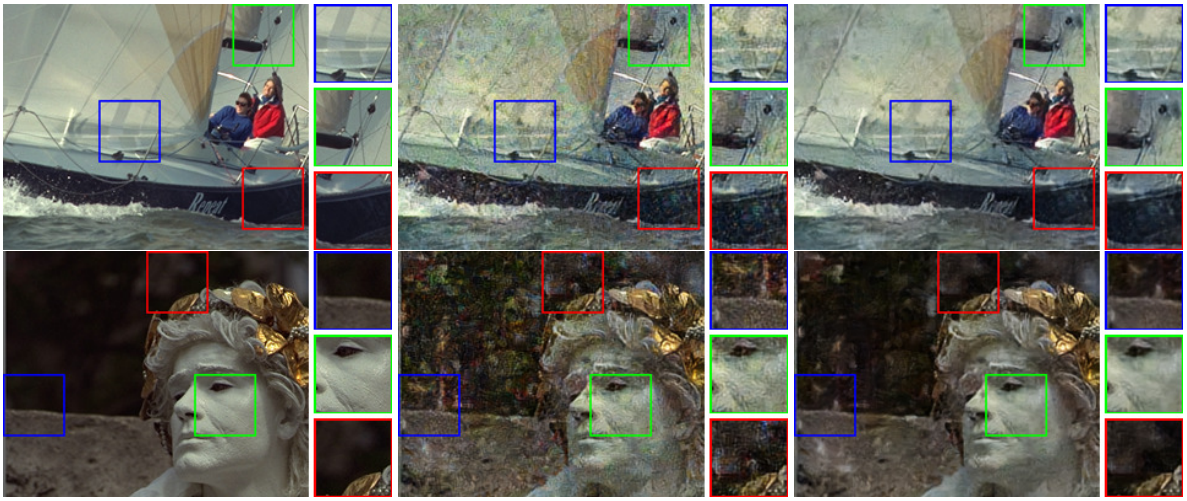


(d) Target (e) One-shot, $SNR_{\text{test}} = 5$dB (f) ISEC, $SNR_{\text{test}} = 5$dB

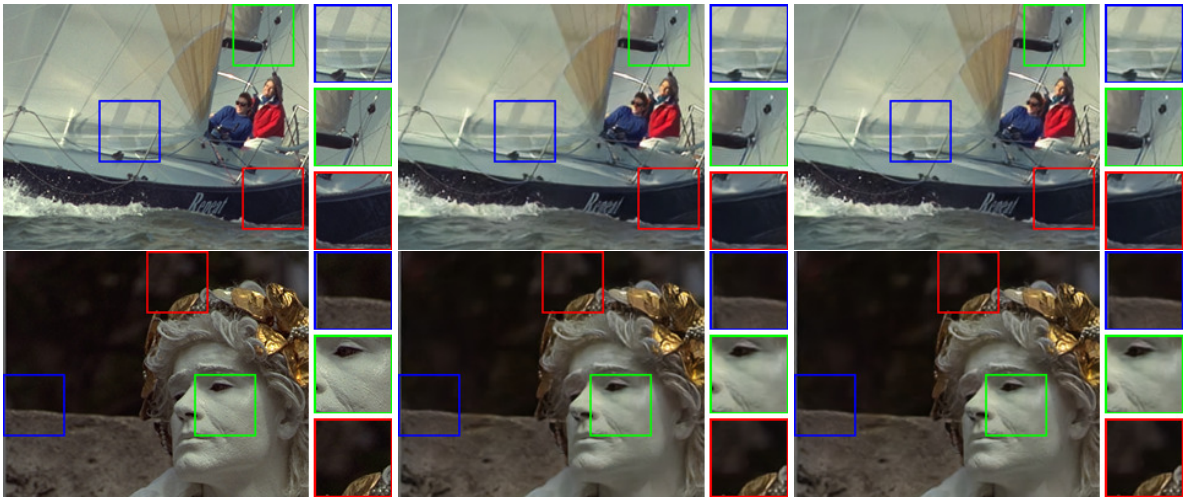


(g) Target (h) One-shot, $SNR_{\text{test}} = 15$dB (i) ISEC, $SNR_{\text{test}} = 15$dB

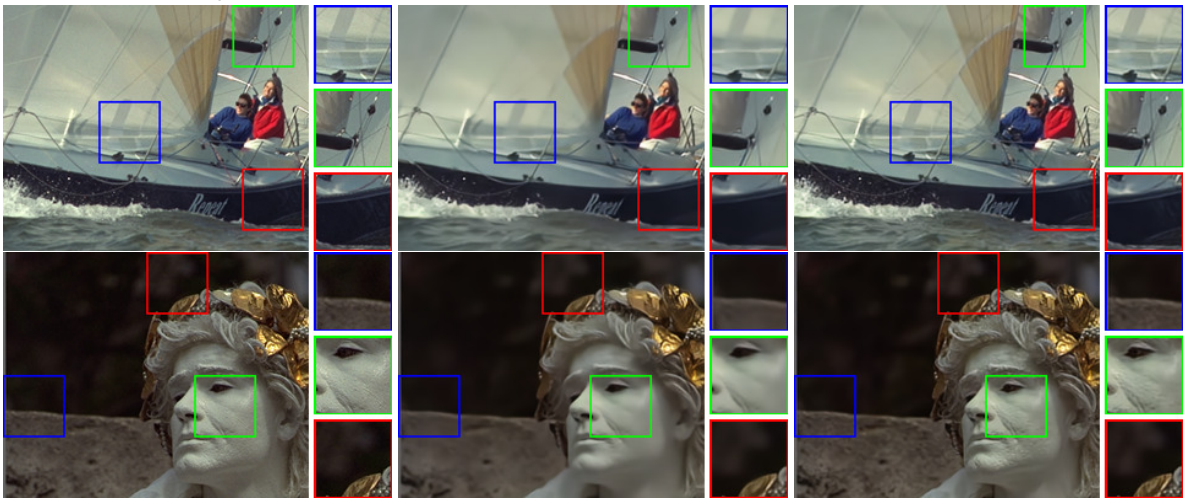
Figure 14: One-shot and ISEC decoding of CIFAR-10 test images using 5dB model on CPP=1/6.



(a) Target (b) Before ISEC, $SNR_{\text{test}} = -5$dB (c) After ISEC, $SNR_{\text{test}} = -5$dB



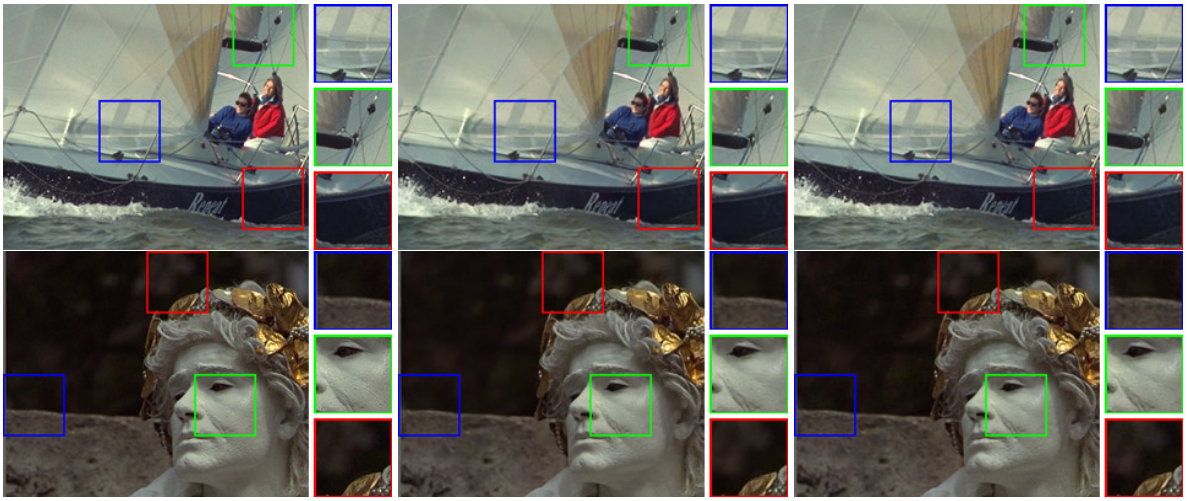
(d) Target (e) Before ISEC, $SNR_{\text{test}} = 1$dB (f) After ISEC, $SNR_{\text{test}} = 1$dB



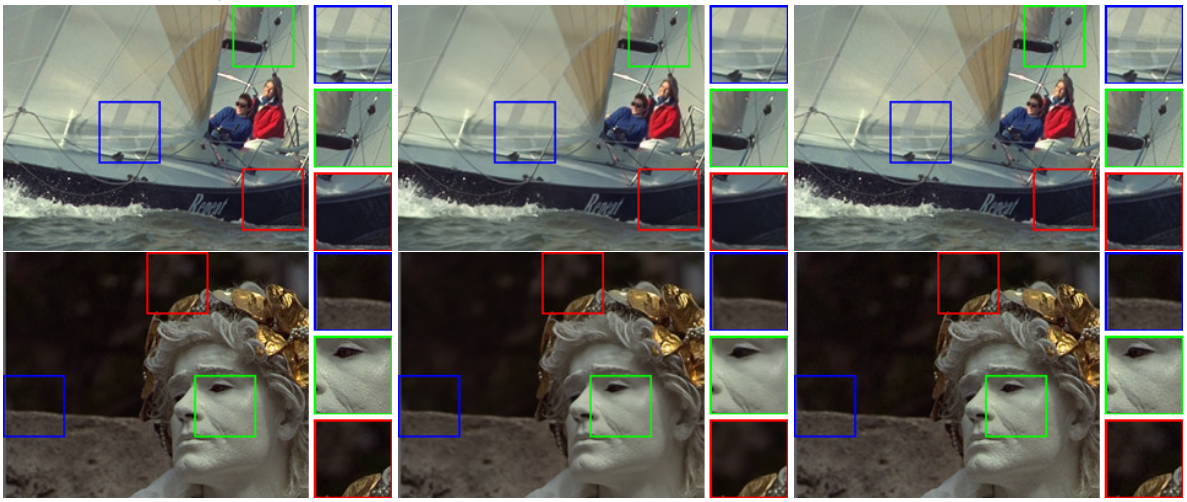
(g) Target (h) Before ISEC, $SNR_{\text{test}} = 7$dB (i) After ISEC, $SNR_{\text{test}} = 7$dB

Figure 15: Before and After ISEC on Kodak images using SNR_{train}=1dB model on CPP=1/6.

(a) Target (b) Before ISEC, $SNR_{\text{test}} = 1$dB (c) After ISEC, $SNR_{\text{test}} = 1$dB

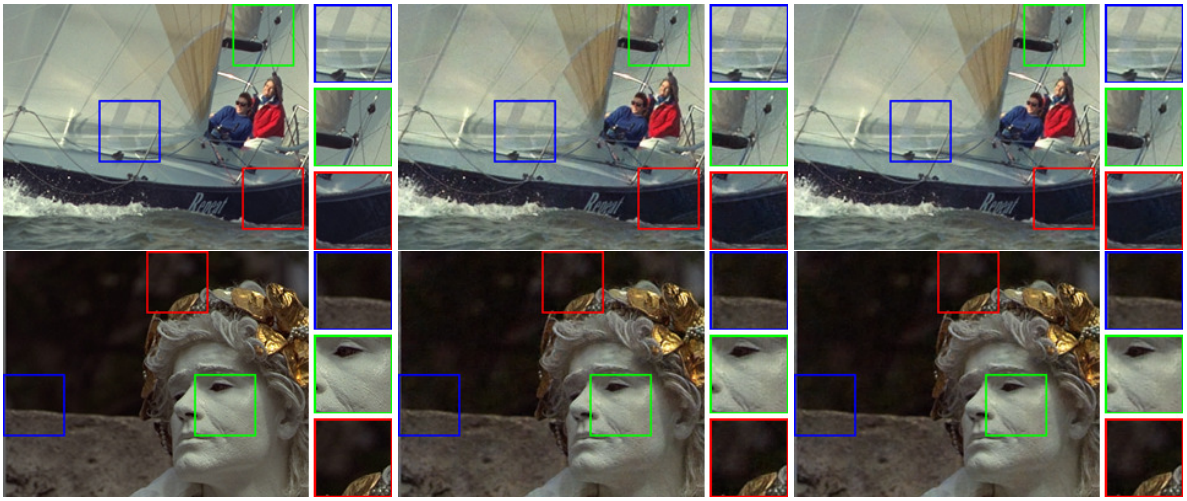


(d) Target (e) Before ISEC, $SNR_{\text{test}} = 7$dB (f) After ISEC, $SNR_{\text{test}} = 7$dB

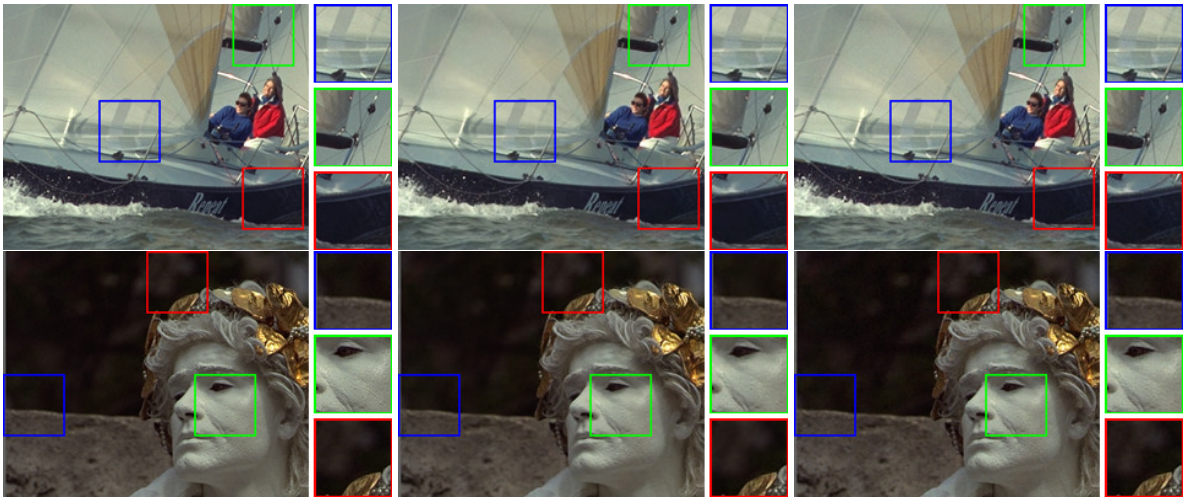


(g) Target (h) Before ISEC, $SNR_{\text{test}} = 13$dB (i) After ISEC, $SNR_{\text{test}} = 13$dB

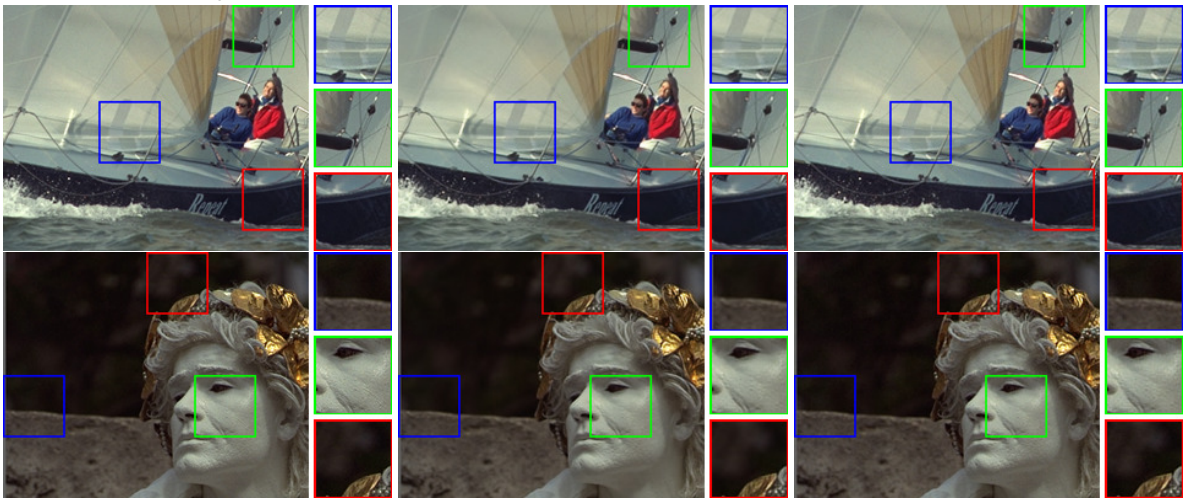
Figure 16: Before and After ISEC on Kodak images using SNR_{train}=7dB model on CPP=1/6.



(a) Target (b) Before ISEC, $SNR_{\text{test}} = 7$dB (c) After ISEC, $SNR_{\text{test}} = 7$dB



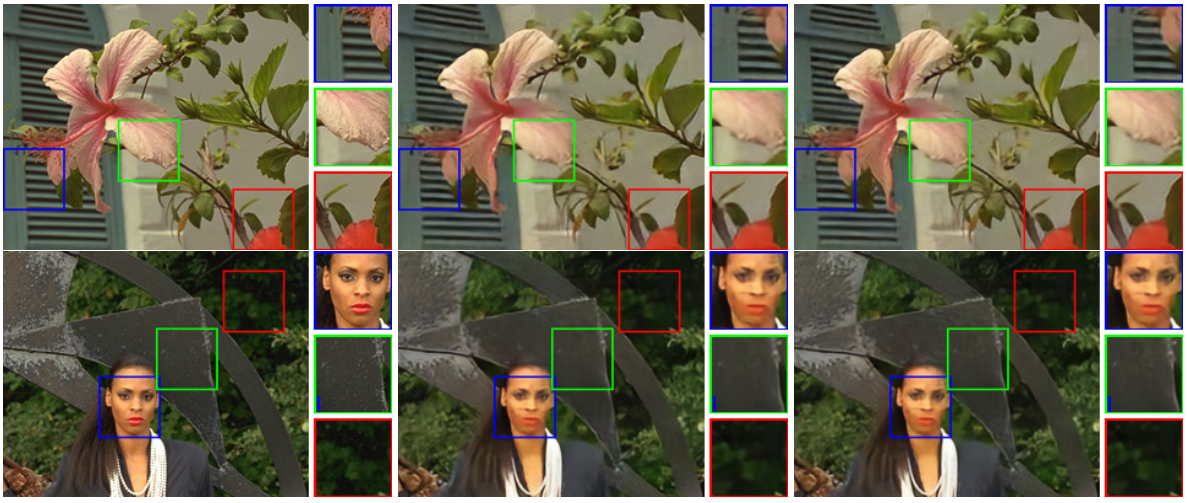
(d) Target (e) Before ISEC, $SNR_{\text{test}} = 13$dB (f) After ISEC, $SNR_{\text{test}} = 13$dB



(g) Target (h) Before ISEC, $SNR_{\text{test}} = 19$dB (i) After ISEC, $SNR_{\text{test}} = 1$dB

Figure 17: Before and After ISEC on Kodak images using SNR_{train}=13dB model on CPP=1/6.

(a) Target (b) Before ISEC, $SNR_{\text{test}} = -5$dB (c) After ISEC, $SNR_{\text{test}} = -5$dB

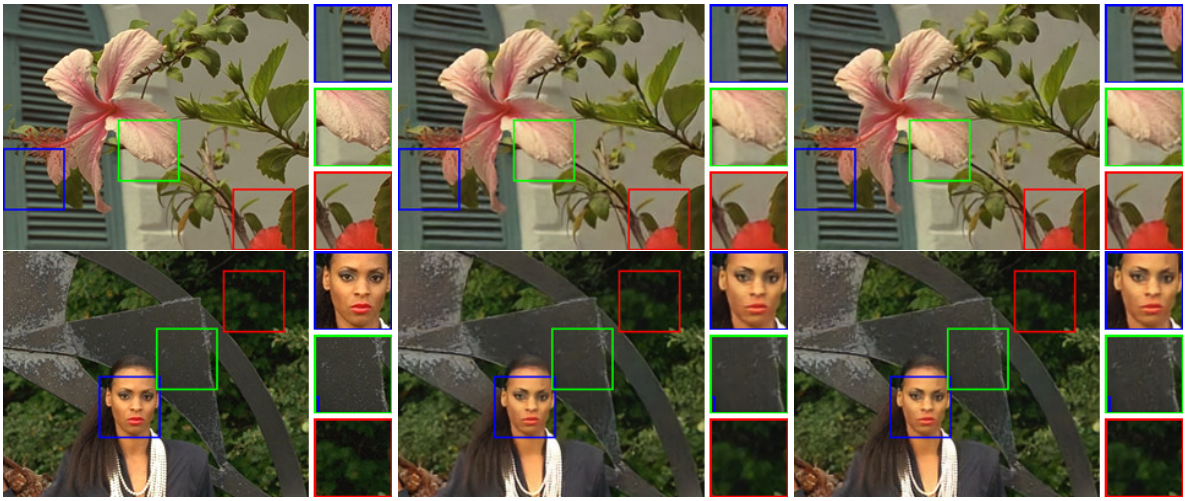


(d) Target (e) Before ISEC, $SNR_{\text{test}} = 1$dB (f) After ISEC, $SNR_{\text{test}} = 1$dB

(g) Target (h) Before ISEC, $SNR_{\text{test}} = 7$dB (i) After ISEC, $SNR_{\text{test}} = 7$dB

Figure 18: Before and After ISEC on Kodak images using SNR_{train}=1dB model on CPP=1/16.

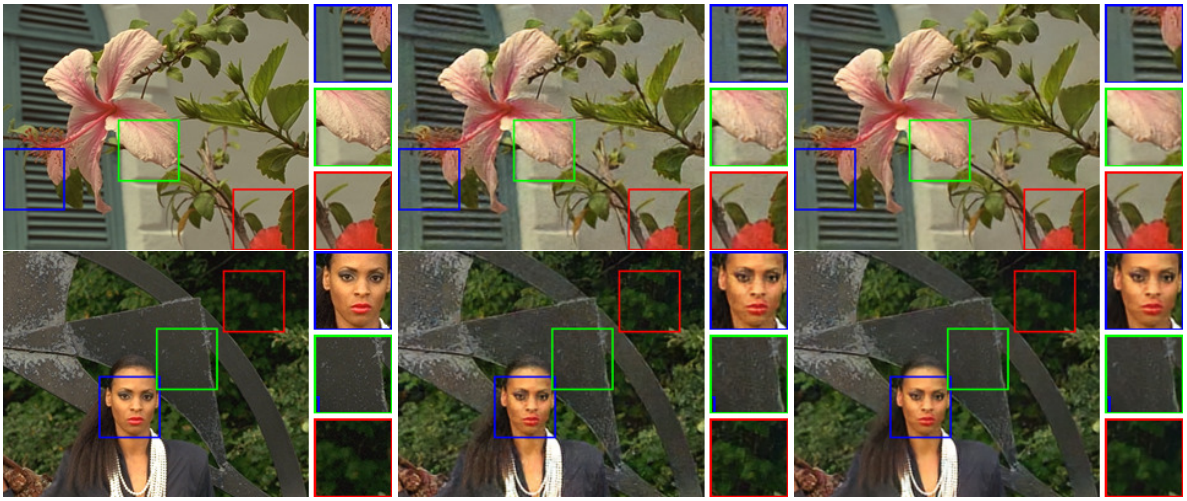
(a) Target (b) Before ISEC, $SNR_{\text{test}} = 1$dB (c) After ISEC, $SNR_{\text{test}} = 1$dB



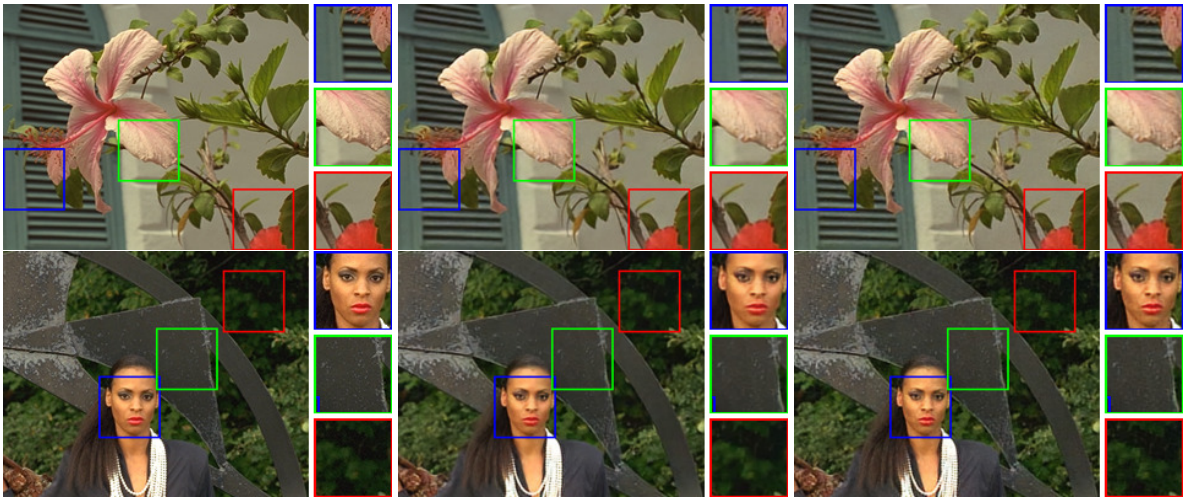
(d) Target (e) Before ISEC, $SNR_{\text{test}} = 7$dB (f) After ISEC, $SNR_{\text{test}} = 7$dB

(g) Target (h) Before ISEC, $SNR_{\text{test}} = 13$dB (i) After ISEC, $SNR_{\text{test}} = 13$dB

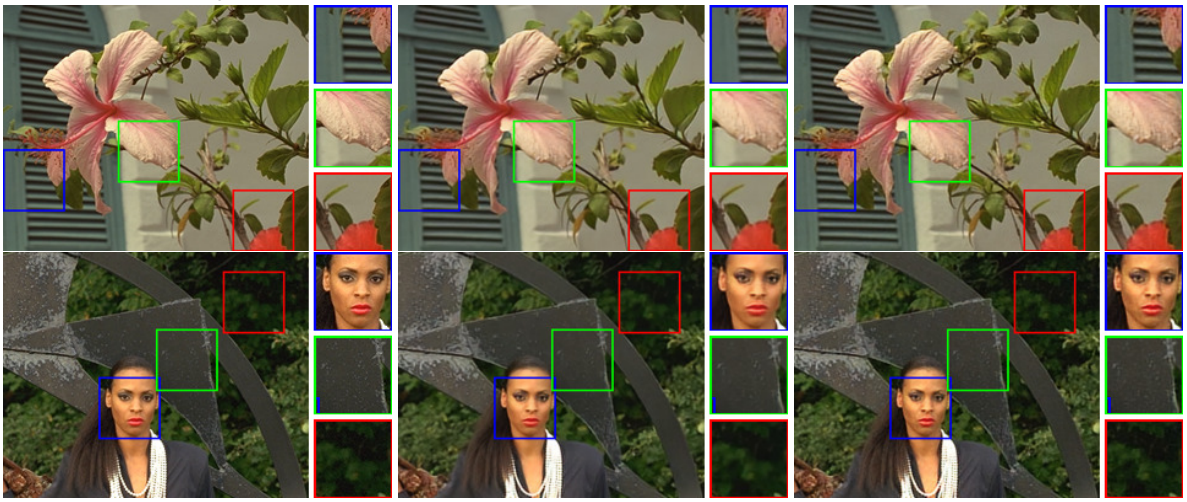
Figure 19: Before and After ISEC on Kodak images using SNR_{train}=7dB model on CPP=1/16.



(a) Target (b) Before ISEC, $SNR_{\text{test}} = 7$dB (c) After ISEC, $SNR_{\text{test}} = 7$dB



(d) Target (e) Before ISEC, $SNR_{\text{test}} = 13$dB (f) After ISEC, $SNR_{\text{test}} = 13$dB



(g) Target (h) Before ISEC, $SNR_{\text{test}} = 19$dB (i) After ISEC, $SNR_{\text{test}} = 1$dB

Figure 20: Before and After ISEC on Kodak images using SNR_{train}=13dB model on CPP=1/16.