Neural Joint Source-Channel Coding via Bernoulli Latent Straight-Through Estimator

Jiwan Seo, Sanghyuk Kim, and Joonhyuk Kang

***Abstract*—Under infinite block length, from Shannon's separation theorem, it is well-known that by independent design of source coding and channel coding, the optimal throughput – the channel capacity – can be reached with careful design of the corresponding functionalities. However, when restricted to finite block length, the separation theorem does not necessarily hold, and hence joint source-channel coding (JSCC) theorem has been raised as an alternative strategy for achieving the capacity of the channel. However, as JSCC formulates highly non-convex problem which cannot be directly solved analytically, recently, deep learning has been proposed as a key enabler for JSCC. While most work on deep-learning-based JSCC focused on transmission of continuous signals through wireless channels, we focus in this paper the case of information-theoretic channel, namely binary symmetric channel, which can give useful insights on information-theoretic perspective on JSCC. Unlike recent work on deep-learning-based, or neural, JSCC for discrete channels that considered score function estimator to train the JSCC, in this paper, we improve the performance of neural JSCC by estimating the gradient more precisely as compared to the previous approach. Key idea is to consider *soft* codeword during training to enable *path-wise* gradient estimator which is proven to have lower variance than the score-function estimator. Experimental results on MNIST and CIFAR-10 datasets show that the proposed neural JSCC outperforms the previous work on JSCC for discrete channels, validating the effectiveness of the proposed gradient computation technique.**

***Index Terms*—Autoencoder, joint source-channel coding (JSCC), straight-through (ST) estimator, variational autoencoder (VAE), variational inference for Monte Carlo objectives (VIMCO).**

I. Introduction

BASED on Shannon's source-channel separation theorem [1], modern communication systems are composed of multiple modules in which source coding and channel coding are treated independently as different modules. The practical communication systems for data transmission across noisy channel consist of *source coding* part that compresses target data to eliminate the inherent redundancy of the data, and *channel coding* part that re-introduces redundancy for error detection and correction purpose. Recently, these coding blocks have been block-wisely developed to further improve their performance. For instance, maximum likelihood (ML) decoding process is regarded as an NP-hard problem, which means that there is a limitation to computational power [2]. Although prior studies on decoding process such as linear programming (LP) of LDPC code are studied, these iterative methods still have limitations by assuming an optimal system block [3].

In particular, for image data transmission, source coding with JPEG [4] or WebP along with channel coding via low density parity check (LDPC) [5], Turbo [6], or Polar [7] coding have demonstrated the performance that reaches the Shannon capacity. However, since Shannon's source-channel separation theorem only holds for long enough codes, to achieve Shannon's capacity also for short codes, joint source-channel coding (JSCC) model which combines source coding and channel coding process has been proposed [8].

In the meantime, deep learning (DL) has been utilized as a powerful tool for various fields including computer vision (CV) [9], natural language processing [10], and wireless communication systems [11], [12], [13], [14], [15] including JSCC that considered wireless image transmission scheme over the additive white Gaussian noise (AWGN) channel [16] and Rayleigh fading channel [17] [18] based on the autoencoder structure of machine learning.

While the works in [16], [17], [18] considered continuous channels, reference [19] proposed JSCC via neural network for discrete channels, coined neural error correction and source trimming (NECST). Specifically, the authors considered binary symmetric channel (BSC) and binary erasure channel (BEC), along with a variational autoencoder (VAE) model to account for stochastic encoding. By maximizing mutual information between image source data and noisy latent codeword, the authors showed that their model outperforms conventional LDPC encoder/decoder.

Learning the discrete-distributed domain has been considered one of the most challenging tasks in machine learning framework. While [19] presented a novel framework for discrete representation learning, it suffers from noisy gradient estimation that generally has large variance due to the nature of the score function estimator [20].

In this paper, we aim at further improving JSCC over discrete channels by proposing low-variance gradient estimator for NECST by replacing score function estimator to pathwise gradient estimator [20]. Key idea is to adopt a straight-through (ST) estimator [21] that allows gradient flow over non-differentiable functions, e.g., discrete sampling of the stochastic message, akin to the way that ST estimator has been

Manuscript received October 5, 2022; approved for publication, October 5, 2022. This paper is specially handled by EIC and Division Editor with the help of three anonymous reviewers in a fast manner.

This work has been supported by the Future Combat System Network Technology Research Center program of Defense Acquisition Program Administration and Agency for Defense Development (UD190033ED).

J. Seo, S. Kim and J. Kang are with the School of Electrical Engineering, Korea advanced institute of science and technology, Daejeon 34141, Korea emails: {jeewan0516, starmoon13, jkang}@kaist.ac.kr)

J. Kang is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2022.000043

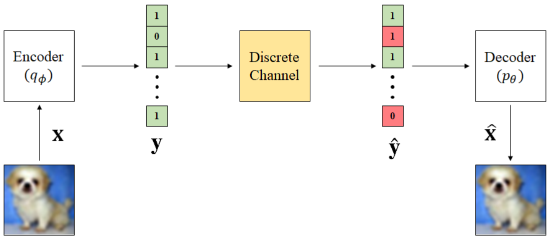

Figure 1. Block diagram of neural JSCC scheme. An image source data **x** is encoded into fixed-length codeword **y**. This codeword is transmitted through a discrete channel so that the decoder receives a noisy codeword **ŷ**. The decoder reconstructs the image source data denoted as **x̂** from the noisy codeword **ŷ**.

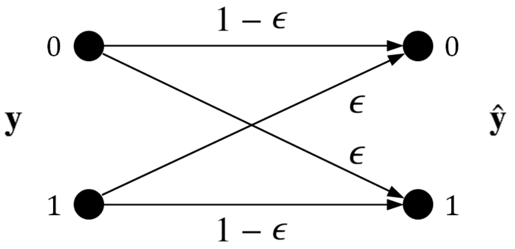

Figure 2. The binary symmetric channel (BSC) with flip probability ϵ.

utilized in Gumbel-softmax [22] and concrete distributions [23] for modeling VAE.

The rest of the paper is organized as follows. In Sec. II, we describe the considered system model and summarize the objective function of neural JSCC. Then, we introduce a naïve, vanilla score function estimator to train the neural JSCC, followed by an improved version that uses multi-samples to reduce the variance of the gradient estimator, which is essentially the NECST training framework. Then, in Sec. III, the proposed training scheme for neural JSCC with path-wise gradient computation with the aid of ST estimator is organized, and the corresponding experimental results are presented in Sec. IV. Lastly, Sec. V concludes the paper.

II. System model

In this section, we review NECST [19]. As mentioned, since discrete latent variables are not directly differentiable, the authors [19] used score function based stochastic gradient estimation techniques (also known as REINFORCE [24]) with variational inference for Monte Carlo objectives (VIMCO) estimator [25]. In this section, we briefly summarize 1) the overall neural NJSCC system model and coding scheme; and 2) the VIMCO-based mutual information optimization in NECST [19].

A. Coding Process over Discrete Channel

Coding process of JSCC is compromised with stochastic encoder (q_ϕ) and decoder (p_θ) that are parameterized by vectors ϕ and θ, respectively. As illustrated in Fig. 1, the encoder network $q_\phi(\mathbf{x})$ maps an input image data $\mathbf{x}$ to an m-bit codeword $\mathbf{y}$ as

$$q_\phi(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{m} \sigma\big([f_\phi(\mathbf{x})]_i\big)^{\mathbf{y}_i}\big(1-\sigma\big([f_\phi(\mathbf{x})]_i\big)\big)^{(1-\mathbf{y}_i)} \quad \text{for } \mathbf{y}_i \in \{0,1\}, \tag{1}$$

with parameterized function $f_\phi(\cdot)$ defined from multi-layer neural network with L^e layers as [26]

$$f_\phi(\mathbf{x}) = f_{\phi^{(L^e)}}(f_{\phi^{(L^e-1)}}(\cdots f_{\phi^{(1)}}(\mathbf{x}))), \tag{2}$$

where $f_{\phi^{(l)}}(x) = g(W^{(l)}x + b^{(l)})$ represents the non-linear activation function of the lth layer with parameter $\phi^{(l)} = \{W^{(l)}, b^{(l)}\}$ with weight matrix $W^{(l)}$ and bias vector $b^{(l)}$ given the vector of neural network parameters $\phi = \{\phi^{(l)}\}_{l=1,\cdots,L^e-1}$, and $[\cdot]_i$ stands for ith element of the corresponding vector. The non-linear function $g(\cdot)$ can be, e.g., a Rectified Linear Unit (ReLU) or a hyperbolic tangent function. While we design the last layer of parameterized function $f_\phi(\cdot)$ of (2) to set the size of the output $f_\phi(\mathbf{x}) \in \mathbb{R}^m$, so that it can represent the m-bit codeword, still, it is in real domain which cannot be directly expressed in m-bit codeword $\mathbf{y}$. Accordingly, we transform the length-m real vector $f_\phi(\mathbf{x})$ into probability vector of length m with sigmoid activation function

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{3}$$

Note that this function is applied element-wise (see (1) and maps negative infinity into probability value of zero, and positive infinity to probability of one. Now, treating $\sigma([f_\phi(\mathbf{x})]_i)$ as the probability of ith bit (among m bits) being 1, we have (1), the stochastic encoder that computes probability of generating random bit $\mathbf{y}$ given input $\mathbf{x}$. Note that all-one vector $\mathbf{y}$ would have probability of $q_\phi(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{m} \sigma\big([f_\phi(\mathbf{x})]_i\big)^{\mathbf{y}_i=1}$, while all-zero vector $\mathbf{y}$ has the probability $\prod_{i=1}^{m}(1-\sigma\big([f_\phi(\mathbf{x})]_i\big))^{(1-\mathbf{y}_i=0)}$.

From (1), the m-bit codeword $\mathbf{y} \sim q_\phi(\cdot|\mathbf{x})$ will be transmitted over discrete channel model, e.g., binary symmetric channel (BSC). We focus on BSC channel in this paper since BSC is commonly treated as a complex channel compared to binary erasure channel (BEC) [27]. In case of BSC, each element of the encoded codeword $\mathbf{y} \in \{0,1\}^m$ is independently flipped with probability ϵ (e.g., $0 \rightarrow 1$ with probability ϵ; $0 \rightarrow 0$ with probability $1-\epsilon$) to form a noisy codeword $\hat{\mathbf{y}}$ as shown in Fig. 2.

Mathematically, the distribution of the noisy codeword $\hat{\mathbf{y}}$ with BSC channel q_{channel} can be formulated as

$$q_{\text{channel}}\big(\hat{\mathbf{y}}|\mathbf{x};\phi,\epsilon\big) = \prod_{i=1}^{m}(\sigma([f_\phi(\mathbf{x})]_i)) - 2\sigma([f_\phi(\mathbf{x})]_i))\epsilon + \epsilon)^{\hat{\mathbf{y}}_i}$$

$$\times \big(1 - \sigma([f_\phi(\mathbf{x})]_i)\big) \tag{4}$$

$$+ 2\sigma([f_\phi(\mathbf{x})]_i)) - \epsilon\big)^{(1-\hat{\mathbf{y}}_i)}, \tag{5}$$

with channel noise level ϵ. Given received noisy codeword $\hat{\mathbf{y}}$ with length m, the decoder network $p_\theta(\cdot|\hat{\mathbf{y}})$ then maps $\hat{\mathbf{y}}$ to reconstructed source data $\hat{\mathbf{x}}$. The decoder network is also defined by parameterized multi-layer neural network analogously to the encoder network as

$$p_\theta(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(f_{\phi^{(L^d)}}(f_{\phi^{(L^{d-1})}}(\cdots f_{\phi^{(1)}}(\mathbf{y})))\sigma_x^2 I\right), \tag{6}$$

with L^d layers assuming Gaussian distribution $\mathcal{N}(\cdot,\cdot)$. In practice, mean of the probabilistic decoder $p_\theta(\mathbf{x}|\mathbf{y})$ in (6) is used as the estimated input $\hat{\mathbf{x}}$.

B. NECST optimization with score function estimator

Estimating and optimizing the mutual information between unknown random variables (e.g., dataset, noisy codeword) is intractable which hinders precise optimization of neural JSCC over discrete channel. To this end, in [19], NECST approximately optimized the variational bound based on amortized inference [28], [29] by maximizing the mutual information between input data $\mathbf{x}$ and noisy codeword $\hat{\mathbf{y}}$ as

$$\max_{\theta,\phi} \; I(\mathbf{x},\hat{\mathbf{y}};\theta,\phi,\epsilon) = H(\mathbf{x}) - H_\theta(\mathbf{x}|\hat{\mathbf{y}};\phi,\epsilon)$$

$$\geq H(\mathbf{x}) + \mathbb{E}_{(\mathbf{x},\hat{\mathbf{y}})\sim q_{\text{channel}}(\mathbf{x},\hat{\mathbf{y}};\phi,\epsilon)}[\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}})\big)], \tag{7}$$

where $H(\cdot)$ denotes differential entropy. To logically denote the encoding distribution with the noisy channel in (7), we define $q_{\text{channel}}(\mathbf{x},\hat{\mathbf{y}};\phi,\epsilon)$ as

$$q_{\text{channel}}(\mathbf{x},\hat{\mathbf{y}};\phi,\epsilon) = q_{\text{channel}}\big(\hat{\mathbf{y}}|\mathbf{x};\phi,\epsilon\big) \times p(\mathbf{x}), \tag{8}$$

where $p(\mathbf{x})$ is the ground-truth of input data distribution. Since the expectation of (7) cannot be computed explicitly, (7) is approximated to form an objective function for NECST as

$$\max_{\theta,\phi} \mathcal{L}_{\text{NECST}}\big(\theta,\phi;x,\epsilon\big)$$

$$= \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}})\big)], \tag{9}$$

with cardinality of the training dataset $|\mathcal{D}|$. As optimization of (9) requires gradient estimation due to discrete sampling, for which [19] adopted score function estimator which suffers from large-variance issue, NECST considers multi-sample objective named VIMCO [25]. Precisely, VIMCO is a score function-based gradient estimator that modifies the discrete latent vector $\mathbf{y}$ into the average of the K-sampled variable $\frac{1}{K}\sum_{i=1}^{K}\mathbf{y}^i$ for learning.

1) Vanilla Score Function Approach: First, we introduce the *vanilla* score function estimator that solves (9) which suffers from large variance, yet being an an unbiased gradient estimator for (9). As computing the gradient with respect to ϕ for the NECST loss $\mathcal{L}_{\text{NECST}}(\theta,\phi;x,\epsilon)$ is the tricky part, we first compute the gradient with respect to the decoder parameter vector θ which is easier to estimate.

$$\nabla_\theta \mathcal{L}_{\text{NECST}}(\theta,\phi;x,\epsilon)$$

$$= \nabla_\theta \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}})\big)]$$

$$= \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\nabla_\theta\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}})\big)]. \tag{10}$$

Above (24) can be easily estimated with Monte Carlo sampling as

$$\nabla_\theta \mathcal{L}_{\text{NECST}}(\theta,\phi;x,\epsilon) \simeq \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\frac{1}{N}\sum_{n=1}^{N}[\nabla_\theta\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^{\mathbf{n}})\big)];$$

$$\mathbf{y}^n \sim q_\phi(\mathbf{y}|\mathbf{x}). \tag{11}$$

In detail, we sample $\mathbf{y}^n$, hence the $\hat{\mathbf{y}}^n$, at forward pass from the conditional stochastic encoder distribution $q_\phi(\mathbf{y}|\mathbf{x})$ given the input $\mathbf{x}$, then the function $p_\theta(\mathbf{x}|\mathbf{y})$ given $\hat{\mathbf{y}}^n$ can be evaluated deterministically. Now we estimate the gradient $\nabla_\phi\mathcal{L}_{\text{NECST}}(\theta,\phi;x,\epsilon)$ with respect to the parameters ϕ, which is more tricky part when training the NECST model. The difficulty comes from dependency of parameter ϕ to the *distribution* of the generated samples $\mathbf{y}$ through $q_\phi(\cdot|\mathbf{x})$, while the loss $\mathcal{L}_{\text{NECST}}$ is computed using the samples $\mathbf{y}$. Mathematically, the corresponding gradient can be computed as

$$\nabla_\phi \mathcal{L}_{\text{NECST}}(\theta,\phi) \tag{12}$$

$$= \nabla_\phi \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}})\big)]$$

$$= \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\nabla_\phi\log q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)$$

$$\times\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}})\big)]. \tag{13}$$

Similar to (10), we can also use Monte Carlo sampling to estimate the gradient with respect to the encoder parameter as

$$\nabla_\phi \mathcal{L}_{\text{NECST}}(\theta,\phi;x,\epsilon)$$

$$\simeq \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\frac{1}{N}\sum_{n=1}^{N}[\nabla_\phi\log q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)\times\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^{\mathbf{n}})\big)];$$

$$\mathbf{y}^n \sim q_\phi(\mathbf{y}|\mathbf{x}). \tag{14}$$

2) Multi-Sample Score Function Approach: However, the variance of score function estimator scales with the sample vector dimension, it generally suffers from high gradient variance issues [22] and [23]. Accordingly, [19] apply a Monte Carlo variance reduction technique, VIMCO, leverages this large variance via multi-samples in (14). Key idea is to sample K latent variables $\mathbf{y}$ instead of single draw in (14) to reduce the variance of the gradient estimate in (14). The NECST optimizing objective with VIMCO can be written as

$$\mathcal{L}^K_{\text{VICMO}}\big(\theta,\phi;x,\epsilon\big)$$

$$= \max_{\theta,\phi}\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}^{1:\mathrm{K}}\sim q_{\text{channel}}(\mathbf{y}|\mathbf{x};\epsilon,\phi)}[\log\frac{1}{K}\sum_{k=1}^{K}\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^k)\big)]. \tag{15}$$

From this modified objective function, the gradient with respect to the decoder parameter vector θ can be written as

$$\begin{aligned}
&\nabla_\theta \mathcal{L}^K_{\text{NECST}}(\theta,\phi;x,\epsilon)\\
&= \nabla_\theta \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}^{\mathbf{1:K}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\log\frac{1}{K}\sum_{k=1}^{K}\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^k)\big)]\\
&= \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}^{\mathbf{1:K}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\log\frac{1}{K}\sum_{k=1}^{K}\nabla_\theta\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^k)\big)],
\end{aligned} \tag{16}$$

and hence estimate with Monte Carlo sampling as

$$\begin{aligned}
&\nabla_\theta \mathcal{L}^K_{\text{NECST}}(\theta,\phi;x,\epsilon)\\
&\simeq \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\frac{1}{N}\sum_{n=1}^{N}[\log\frac{1}{K}\sum_{k=1}^{K}\nabla_\theta\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^{n,k})\big)];\\
&\qquad\qquad \mathbf{y}^{n,k}\sim q_\phi(\mathbf{y}|\mathbf{x}).
\end{aligned} \tag{17}$$

Similarly, the gradient with respect to the encoder parameter vector ϕ can be computed with lower variance thanks to multi sampling as

$$\begin{aligned}
&\nabla_\phi \mathcal{L}^K_{\text{NECST}}(\theta,\phi)\\
&= \nabla_\phi \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}^{\mathbf{1:K}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\log\frac{1}{K}\sum_{k=1}^{K}\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^k)\big)]\\
&= \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}^{\mathbf{1:K}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\nabla_\phi \log q_{\text{channel}}(\hat{\mathbf{y}}^{1:K}|\mathbf{x};\epsilon,\phi)\\
&\quad\times\log\frac{1}{K}\sum_{k=1}^{K}\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^k)\big)]\\
&= \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}^{\mathbf{1:K}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\nabla_\phi \log\prod_{k=1}^{K} q_{\text{channel}}(\hat{\mathbf{y}}^{k}|\mathbf{x};\epsilon,\phi)\\
&\quad\times\log\frac{1}{K}\sum_{k=1}^{K}\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^k)\big)]\\
&= \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}^{\mathbf{1:K}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\nabla_\phi \sum_{k=1}^{K}\log q_{\text{channel}}(\hat{\mathbf{y}}^{k}|\mathbf{x};\epsilon,\phi)\\
&\quad\times\log\frac{1}{K}\sum_{k=1}^{K}\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^k)\big)].
\end{aligned} \tag{18}$$

Finally, this gradient can be estimated with Monte Carlo sampling as

$$\begin{aligned}
&\nabla_\phi \mathcal{L}^K_{\text{NECST}}(\theta,\phi;x,\epsilon)\\
&\simeq \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\frac{1}{N}\sum_{n=1}^{N}[\nabla_\phi\frac{1}{K}\sum_{k=1}^{K}\log q_{\text{channel}}(\hat{\mathbf{y}}^{n,k}|\mathbf{x};\epsilon,\phi)\\
&\quad\times\log\frac{1}{K}\sum_{k=1}^{K}\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}^{n,k})\big)];\\
&\qquad\qquad \mathbf{y}^{n,k}\sim q_\phi(\mathbf{y}|\mathbf{x}).
\end{aligned} \tag{19}$$

III. Low-Variance Gradient Estimator for Neural JSCC

Even though NECST utilized VIMCO to reduce the variance of the estimated gradient as compared to vanila score function estimator, it still suffers from large variance due to limitations of score function estimator while it also requires increase computational complexity due to multi sampling. Furthermore, it requires different objective function for decoder (15) and encoder (17) that requires careful design of the loss functions. Recalling that the reason for such multi samples and different objective function stemming from score function estimation comes from the non-differentiable nature of sample $\mathbf{y}$ which is a realization of sample draw from parameterized distribution $q_\phi(\mathbf{y}|\mathbf{x})$, we simplify the problem by making the codeword $\mathbf{y}$ a differentiable function, and hence reduce the variance of the gradient estimation. Key idea is to use path-wise gradient estimation instead of score-function estimation by replacing the sampled codeword $\mathbf{y}$ with *soft codeword* $\mathbf{y}'$ that can be directly differentiated with respect to the encoder parameter vector ϕ.

1) Path-wise Gradient Approach: As mentioned, to optimize the encoder (q_ϕ) and decoder (p_θ) at once, we introduce an end-to-end optimized neural JSCC model without using VIMCO that requires multi-sampling Bernoulli-distributed binary codeword during training step. Accordingly, we directly minimize the loss function $\mathcal{L}_{\text{NECST}}\big(\theta,\phi;x,\epsilon\big)$. However, the problem lies in the discrete nature of the latent codeword $\mathbf{y}$.

The Straight-Through (ST) estimator [21] is a novel method to simply avoid its limitation. This technique is widely used in recent discrete VAE researches [30] [31] to allow gradient flow over non-differentiable backward pass. To this end, we propose to utilize lower variance gradient estimator via path-wise gradient estimation to solve (14) via ST estimator. Precisely, the ST estimator replaces non-differentiable m-bit latent codeword $\mathbf{y}$ into differentiable codeword $\mathbf{y}'$ as

$$\mathbf{y}' = \texttt{stop\_gradient}\Big(\mathbf{y}-\sigma\big(f_\phi(\mathbf{x})\big)\Big)+\sigma\big(f_\phi(\mathbf{x})\big), \tag{20}$$

where the function $\texttt{stop\_gradient}(\cdot)$ removes dependency with respect to the learnable parameters, i.e., θ and ϕ here. Note that the value itself is same for $\mathbf{y}'$ and $\mathbf{y}$. Thus, the gradient of Bernoulli-sampled noisy codeword $\nabla_\phi\hat{\mathbf{y}}$ can be replaced by $\nabla_\phi\hat{\mathbf{y}}'$.

ST gradient estimator in (20) lead to ignore the Bernoulli distributed sampling in encoding part (1). Finally, we can redefine end-to-end optimizing objective (9) for our neural JSCC model with Bernoulli ST (BernST) method as

$$\begin{aligned}
&\min_{\theta,\phi}\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\mathbf{y}|\mathbf{x};\epsilon,\phi)}[-\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}})\big)]\\
&\simeq \underbrace{\min_{\theta,\phi}\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\mathbf{y}|\mathbf{x};\epsilon,\phi)}[-\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}')\big)]}_{:=\mathcal{L}_{\text{BernST}}\left(\theta,\phi;x,\epsilon\right)},
\end{aligned} \tag{21}$$

where $\hat{\mathbf{y}}'$ is the output of the channel using the differentiable codeword $\mathbf{y}'$ generated from (18).

Given the loss function, the gradient with respect to the decoder parameter θ can be computed as

$$\begin{aligned}
&\nabla_\theta\mathcal{L}_{\text{BernST}}(\theta,\phi;x,\epsilon)\\
&= \nabla_\theta\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)}[\log\big(p_\theta(\mathbf{x}|\hat{\mathbf{y}}')\big)]
\end{aligned}$$

$$= \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{\hat{\mathbf{y}}\sim q_{\text{channel}}(\hat{\mathbf{y}}|\mathbf{x};\epsilon,\phi)} [\nabla_\theta \log\left(p_\theta(\mathbf{x}|\hat{\mathbf{y}}')\right)], \tag{22}$$

which can be estimated using Monte Carlo sampling as

$$\nabla_\theta \mathcal{L}_{\text{BernST}}(\theta,\phi;x,\epsilon) \simeq \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \frac{1}{N} \sum_{n=1}^{N} [\nabla_\theta \log\left(p_\theta(\mathbf{x}|\hat{\mathbf{y}}'^{\mathbf{n}})\right)];$$

$$\text{for } \mathbf{y}^n \sim q_\phi(\mathbf{y}|\mathbf{x}) \text{ and}$$

$$\mathbf{y}'^n = \texttt{stop\_gradient}\Big(\mathbf{y}^n - \sigma\big(f_\phi(\mathbf{x})\big)\Big) + \sigma\big(f_\phi(\mathbf{x})\big). \tag{23}$$

Similarly, the gradient for the encoder parameter vector ϕ can be obtained as

$$\begin{aligned}
&\nabla_\phi \mathcal{L}_{\text{BernST}}(\theta,\phi;x,\epsilon) \\
&= \nabla_\phi \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_\delta [\log\left(p_\theta(\mathbf{x}|\hat{\mathbf{y}}'(\phi,\delta))\right)] \\
&= \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_\delta [\nabla_\theta \log\left(p_\theta(\mathbf{x}|\hat{\mathbf{y}}'(\phi,\delta))\right)],
\end{aligned} \tag{24}$$

where the expectation $\mathbb{E}_\delta$ is taken with respect to inherent randomness that does not depend on the learnable parameter ϕ and θ, e.g., channel noise in $q_{\text{channel}}(\cdot)$, and we particularly emphasize the direct dependency of $\hat{\mathbf{y}}'$ with respect to parameter ϕ and the inherent random variable δ by $\hat{\mathbf{y}}'(\phi,\delta)$. Furthermore, the gradient can be estimated using Monte Carlo sampling as

$$\nabla_\phi \mathcal{L}_{\text{BernST}}(\theta,\phi;x,\epsilon) \simeq \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \frac{1}{N} \sum_{n=1}^{N} [\nabla_\phi \log\left(p_\theta(\mathbf{x}|\hat{\mathbf{y}}'^{\mathbf{n}})\right)];$$

$$\text{for } \mathbf{y}^n \sim q_\phi(\mathbf{y}|\mathbf{x}) \text{ and}$$

$$\mathbf{y}'^n = \texttt{stop\_gradient}\Big(\mathbf{y}^n - \sigma\big(f_\phi(\mathbf{x})\big)\Big) + \sigma\big(f_\phi(\mathbf{x})\big). \tag{25}$$

It is worth noting that, the objective of NECST (15) and BernST (21) are essentially same except for that the proposed objective (21) has lower variance with less computational complexity which leads to an efficient and effective learning of entire neural JSCC system.

IV. Experimental Results

In this section, to validate our method, we demonstrate the compression and error correction performance of BernST and compare them to NECST. Our comparison is on the MNIST [32] and CIFAR-10 [33] datasets using 50000/10000/10000 and 45000/5000/10000 split into the training, validation and test sets, respectively. For fair comparison, the architecture and hyperparameters of the encoder and decoder are set as in [19]. All of our experiments are implemented using Pytorch, which allows for automatic differentiation through the gradient updates.

To begin with, Table I and Table II compare the performance of compression and error correction with fixed number of bits for MNIST of 28×28 gray scale images of handwritten digits and CIFAR-10 of 32×32 RGB images dataset, an RGB image dataset of 32x32 size, respectively. The results reported the L2 distance between the original and reconstructed image in test set per single image that showed the best performance on the validation set during training. It can be easily checked that the proposed method achieves better reconstructed image compared to original NECST thanks to the proposed variance reduction technique.

Table I
L2 DISTANCE BETWEEN ORIGINAL AND RECONSTRUCTED IMAGE ON MNIST DATASET.

MNIST dataset		
Noise ϵ	100-bit NECST	100-bit BernST
0	9.826	**8.298**
0.1	15.05	**12.58**
0.2	22.62	**20.31**
0.3	32.11	**31.33**
0.4	**45.51**	47.76

Table II
L2 DISTANCE BETWEEN ORIGINAL AND RECONSTRUCTED IMAGE ON CIFAR-10 DATASET.

CIFAR-10 dataset		
Noise ϵ	500-bit NECST	500-bit BernST
0	**19.25**	19.3
0.1	29.73	**28.59**
0.2	44.13	**42.25**
0.3	64.13	**62.42**
0.4	101.2	**100.6**

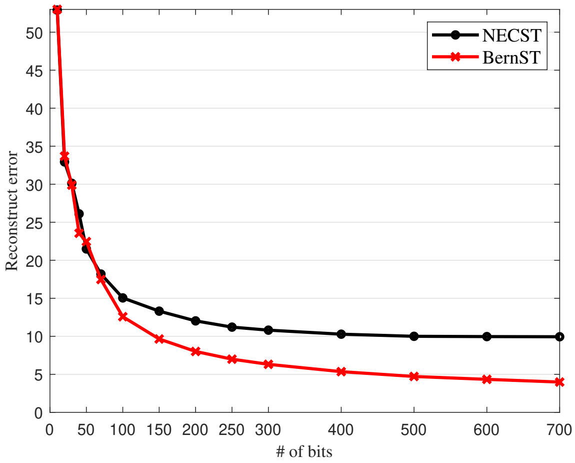

Figure 3. Reconstruction error (lower is better) according to number of bits used for NECST and BernST on MNIST dataset. The channel noise level is fixed to $\epsilon = 0.1$.

To investigate the effect of number of bits for transmission, Fig. 3 shows reconstruction performance over BSC channel as a function of the number of bits with fixed noise level $\epsilon = 0.1$. In Fig. 3, we can see that our proposed method outperforms NECST over all regions while the gap becomes enlarged in the sufficient bits regime due to the ability of neural joint source coding that handles compression and noise correction simultaneously.

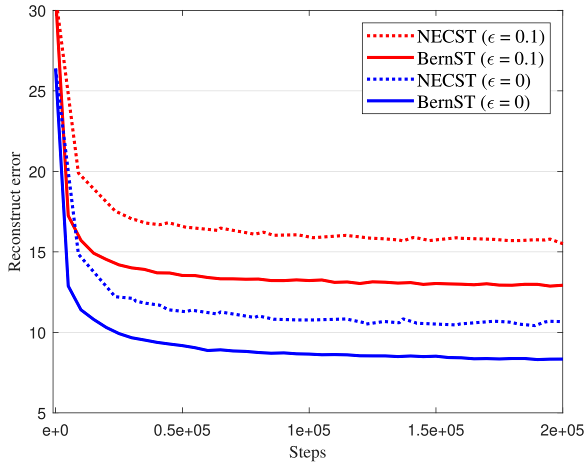

Figure 4. Reconstruction error (lower is better) comparison of BernST and NECST on MNIST dataset. Noiseless channel ($\epsilon = 0$) and noisy ($\epsilon = 0.1$) channel environments are considered. The loss is computed from validation set during training. The number of samples for NECST is fixed to $K = 5$.

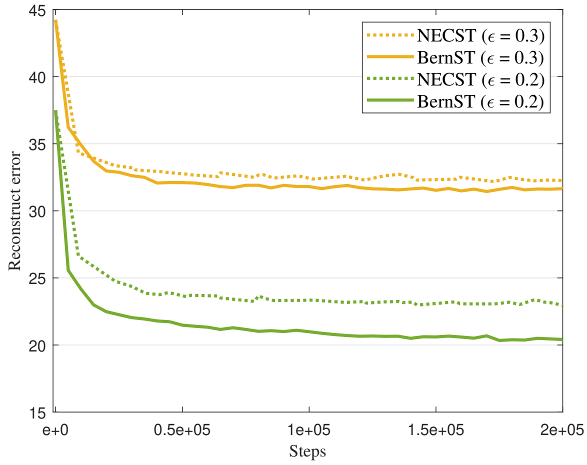

Figure 5. Reconstruction error (lower is better) comparison of BernST and NECST on MNIST dataset. Noisy ($\epsilon = 0.2$ and 0.3) channel environments are considered. The loss is computed from validation set during training. The number of samples for NECST is fixed to $K = 5$.

Figs. 4 and 5 shows the reconstruction error during the training process for BernST and NECST according to the channel noise level. There is not much difference in the early stages of learning, but this is because the encoder is not well trained in the early stages and there is not enough information in the encoded codewords. Fig. 4 directly shows that reconstruction performance of BernST is much higher than NECST during entire training process. As shown in Fig. 5, the advantages of BernST methods are not clearly evident as the channel noise level is increased ($\epsilon > 0.3$), it is meaningful that it consistently product good performance at overall channel noise levels.

Lastly, as mentioned, while the proposed BernST encodes source data with single-sampled Bernoulli codewords, NECST with VIMCO requires multi samples. To examine the effect of multi samples, we plot the reconstruction error with respect to training iterations for different multiple sample numbers. In Fig. 6, increase in multi sample indeed achieves better reconstruction performance thanks to variance reduction also as reported in [19], our method outperforms NECST with single sample thanks to the low-variance nature of pathwise gradient estimator.

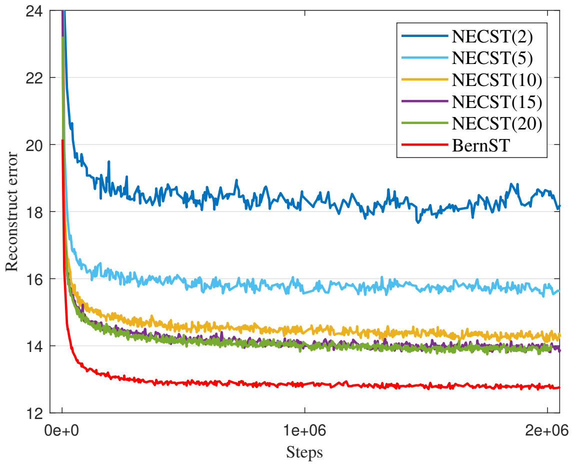

Figure 6. Reconstruction error (lower is better) comparison of BernST and multi-sample objective for NECST on MNIST dataset. The loss is computed from validation set during training. Different number of multi-samples are considered, i.e., $K = (2, 5, 10, 15, 20)$ (15). The channel noise level is fixed to $\epsilon = 0.1$.

V. Conclusion

In this paper, neural JSCC for discrete channels has been proposed with reduced variance for the gradient estimator that leads to improved performance as compared to the state-of-the-art neural JSCC, NECST. By utilizing path-wise gradient computation with straight-through estimator, we were able to show that the proposed BernST reconstructs the original image better than NECST for almost all noise levels and number of bits. Also, the proposed approach shows faster convergence with lower computational complexity thanks to the accurate estimation of the training gradient. Experimental results on both MNIST and CIFAR-10 dataset demonstrate that the proposed neural JSCC shows better compression and error correction performance as compared to the previous neural JSCC system for various channel conditions. As a final remark, one of the limitations of the existing (including ours) neural JSCC system is that the model must be trained separately on various code lengths and dataset. Future work may consider meta-learning [26], [34] of neural JSCC system for a variable-length code to adapt to the new environment quickly.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[2] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Trans. Inf. Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[3] M. Punekar and M. F. Flanagan, "Low complexity linear programming decoding of nonbinary linear codes," in *Proc. IEEE Allerton*, 2010.

[4] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[5] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Lett.*, vol. 32, no. 18, p. 1645, 1996.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. IEEE ICC*, 1993.

[7] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes," in *Proc. IEEE ISIT*, 2008.

[8] Y. Zhong, F. Alajaji, and L. L. Campbell, "On the joint source-channel coding error exponent for discrete memoryless systems," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1450–1468, 2006.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Inf. Process. Syst.*, vol. 25, 2012.

[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *2014, arXiv preprint arXiv:1406.1078*.

[11] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648–664, 2018.

[12] T. O'shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.

[13] S. Park, O. Simeone, and J. Kang, "Meta-learning to communicate: Fast end-to-end training for fading channels," in *Proc. IEEE ICASSP*, 2020.

[14] S. Park, O. Simeone, and J. Kang, "End-to-end fast training of communication links without a channel model via online meta-learning," in *Proc. IEEE SPAWC*, 2020.

[15] H. Kim *et al.*, "Communication algorithms via deep learning," *2018, arXiv preprint arXiv:1805.09317*.

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

[17] D. B. Kurka and D. Gündüz, "Deepjscc-f: Deep joint source-channel coding of images with feedback," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 178–193, 2020.

[18] M. Yang, C. Bian, and H.-S. Kim, "Deep joint source channel coding for wireless image transmission with ofdm," in *Proc. IEEE ICC*, 2021.

[19] K. Choi, K. Tatwawadi, A. Grover, T. Weissman, and S. Ermon, "Neural joint source-channel coding," in *Proc. ICML*, 2019.

[20] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih, "Monte carlo gradient estimation in machine learning." *J. Mach. Learn. Res.*, vol. 21, no. 132, pp. 1–62, 2020.

[21] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *2013, arXiv preprint arXiv:1308.3432*.

[22] E. Jang, S. S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," vol. 2017, *arxiv/1611.01144*.

[23] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," vol. 2017, *arxiv/1611.00712*.

[24] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[25] A. Mnih and D. Rezende, "Variational inference for monte carlo objectives," in *Proc. ICML*, 2016.

[26] O. Simeone, S. Park, and J. Kang, "From learning to meta-learning: Reduced training overhead and complexity for communication systems," in *Proc. IEEE 6G SUMMIT*, 2020.

[27] T. Richardson and R. Urbanke, *Modern coding theory*. Cambridge university press, 2008.

[28] D. Barber and F. Agakov, "Kernelized infomax clustering," *Advances in Neural Inf. Process. Syst.*, vol. 18, 2005.

[29] D. J. MacKay *et al.*, *Information Theory, Inference and Learning Algorithms*. Cambridge university press, 2003.

[30] A. Van Den Oord *et al.*, "Neural discrete representation learning," *Advances in Neural Inf. Process. Syst.*, vol. 30, 2017.

[31] A. Razavi, A. Van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *Advances in Neural Inf. Process. Syst.*, vol. 32, 2019.

[32] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[33] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," *URL http://www. cs. toronto. edu/kriz/cifar. html*, vol. 5, no. 4, p. 1, 2010.

[34] S. Park, H. Jang, O. Simeone, and J. Kang, "Learning to demodulate from few pilots via offline and online meta-learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 226–239, 2020.

Jiwan Seo received the B.S. and M.S. degrees in the Department of Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2019 and 2021. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, KAIST, Daejeon, South Korea. His research interests include the field of communication theory, machine learning, and statistical signal processing, with emphasis on machine learning usage for communication systems.

Sanghyuk Kim received the B.S. degree in the Department of Mathematical Science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2017, and M.S. degree in the Department of Electrical Engineering , KAIST, Daejeon, South Korea in 2019. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, KAIST, Daejeon, South Korea. His research interests include intelligent reflecting surface (IRS), wireless power transfer, and localization.

Joonhyuk Kang received the B.S.E. and M.S.E. degrees from Seoul National University, Seoul, South Korea, in 1991 and 1993, respectively, and the Ph.D. degree in Electrical and Computer Engineering from The University of Texas at Austin, Austin, in 2002. From 1993 to 1998, he was a Research Staff Member at Samsung Electronics, Suwon, South Korea, where he was involved in the development of DSP-based real-time control systems. In 2000, he was with Cwill Telecommunications, Austin, TX, USA, where he participated in the project for multicarrier CDMA systems with antenna array. He was a Visiting Scholar with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA, from 2008 to 2009. He is currently serving as Head of the School of Electrical Engineering (EE), KAIST, Daejeon, South Korea. His research interests include signal processing and machine learning for wireless communication systems. He is a life-member of the Korea Information and Communication Society and the Tau Beta Pi (the Engineering Honor Society). He is a Recipient of IEEE VTS Jack Neubauer Memorial Award in 2021 for his paper titled "Mobile Edge Computing via a UAV-Mounted Cloudlet: Optimization of Bit Allocation and Path Planning."