Design of Communication Systems using Deep Learning: A Variational Inference Perspective

Vishnu Raj Sheetal Kalyani
Department of Electrical Engineering,
Indian Institute of Technology Madras,
Chennai, India, 600 036.
{ee14d213,skalyani}@ee.iitm.ac.in

***Abstract*—Recent research in the design of end to end communication system using deep learning has produced models which can outperform traditional communication schemes. Most of these architectures leveraged autoencoders to design the encoder at the transmitter and decoder at the receiver and train them jointly by modeling transmit symbols as latent codes from the encoder. However, in communication systems, the receiver has to work with noise corrupted versions of transmit symbols. Traditional autoencoders are not designed to work with latent codes corrupted with noise. In this work, we provide a framework to design end to end communication systems which accounts for the existence of noise corrupted transmit symbols. The proposed method uses deep neural architecture. An objective function for optimizing these models is derived based on the concepts of variational inference. Further, domain knowledge such as channel type can be systematically integrated into the objective. Through numerical simulation, the proposed method is shown to consistently produce models with better packing density and achieving it faster in multiple popular channel models as compared to the previous works leveraging deep learning models.**

***Index Terms*—Physical Layer, Deep Learning, Variational Inference, Autoencoders**

I. Introduction

The aim of any communication system is to perfectly reproduce the message at the receiver sent by a transmitter through a channel between the sender and receiver. Due to the noise characteristics of the channel, the transmitted signal can get corrupted, and the exact reconstruction of the message may not happen at the receiver. A robust communication system should be able to handle these corruptions due to the channel and reproduce the message with maximum faithfulness at the receiver.

Traditional communication systems follow a block by block design, optimized within the block for maximal performance. However, such a system may not result in a globally optimum solution across all blocks. The complexity of the signaling systems, along with the unknown effect from the channel, makes it difficult to design an optimal system across all the blocks. Lately, deep learning has seen extraordinary success in learning complex tasks involving natural signals such as images, speech, etc. In the area of communication systems also, applications of deep learning have resulted in improved results. In [1], a deep learning based approach by unfolding the projected gradient descent algorithm is proposed for MIMO detection. A deep learning based method for channel estimation in OFDM systems with one-bit quantization is developed in [2]. Interestingly, the one-bit quantized OFDM systems with deep learning based estimation is able to provide lower error than least-squares channel estimation with unquantized samples. Interested readers are redirected to [3] for a broad discussion on how deep learning can help to improve physical layer of communication systems.

In [4], the authors proposed the fascinating idea of an end to end design communication system based on the principles of autoencoders [7]. However, to train the system end to end, channel knowledge was required for computing the weight updates during backpropagation. To overcome the problem of unknown channel model, [5] proposed to train the network in two phases: in the first phase train both the transmitter and receiver networks in simulation with known channel model and second phase deploy the network in actual channel and fine-tune the receiver network alone. A practical approach to train systems from end to end without any assumptions about the channel is proposed in [8] based on simultaneous perturbation stochastic approximations. Another method is proposed in [6] based on output perturbations at the transmitter. Approaches to approximate the channel distribution with neural networks and use this as a surrogate channel for backpropagation are proposed in [9], [10].

The success of deep learning approach for transceiver design is not just limited to wireless communication systems. In the context of optical communication systems, [11] introduces an end to end deep learning based optical communication transceiver for generating robust transmit waveforms used for communication which is achieved by using a modified ReLU activation at the output of transmitter. In molecular communication systems, a deep learning based approach to optimize the receiver design in the presence of Inter-Symbol Interference (ISI) is presented in [12]. In the context of Underwater Acoustic communications, [13] proposes a novel channel estimation technique for Orthogonal Frequency Division Multiplexing (OFDM) systems which is capable of providing better performance than traditional Least Squares (LS) and Minimum Mean Square Error (MMSE) estimators.

Previous works on end to end communication system design using deep learning [4]–[6], [8], [14] relied on Autoencoders (AE) for designing the encoder and decoder. One of the original purposes of AE is to perform dimensionality reduction

TABLE I: Comparison of Proposed method to AE based models

Characteristic	AE-based methods [4]–[6]	Proposed Method
Basic Concept	Autoencoders	Variational Autoencoders
Accounts for noise in latent code	No	Yes
Constant SNR required at training	Yes	No
Method for Power control	Through normalization layer	Through KL-divergence term in loss function
Type of power constraint	Hard constraint	Soft constraint

[15] by using the latent codes produced by the encoder as compressed representation. The works in [4]–[6] used the concepts of AE to train an encoder for mapping a symbol to be transmitted to a constellation point and a decoder for decoding the learned mapping. However, when using AEs for end to end communication system design, two fundamental problems remain.

1) By using a normalization layer at the end of transmitter (encoder), the AE based designs effectively hard constrain the parameter space. The normalization layer was introduced to achieve power constraint at the encoder since otherwise, one can trivially increase the transmit power to achieve better reconstruction at the receiver. However, such a hard constraint in one of the layers of a deep network will impact the loss surface and parameter space one can explore [16]. This could lead to trading off better designs for hard power constraints.
2) In the context of communication systems, the decoder has to operate at a noisy version of the latent code produced by the encoder (transmitter and channel combined). However, autoencoders are not designed to act on noisy latent codes and to the best of our knowledge, there exists no theoretical work on the behavior of AEs in the presence of noisy latent codes. A variant of AEs known as *Noisy Autoencoders* can be used to work with noisy inputs [17], but not with a noise corrupted latent variable.

This motivated us to investigate models that can handle noisy latent codes, has a theoretical backing for the same, and which also imposes power constraint but as a soft constraint - hence enabling more exploration and subsequently leading to better constellation designs when compared to using AEs.

We propose a method based on the principles of Variational Autoencoders (VAEs) [18] which allows to account for the noisy latent codes and provides a systematic way for introducing soft constraints on transmit power. VAEs were originally proposed as a distribution approximating method for generative modeling. VAEs approximate a complete distribution of the latent codes, typically using a multivariate Gaussian distribution by characterizing the mean and variance of the distribution. The proposed method uses the encoder to predict only the location of the transmit symbol, and channel is the entity that adds corruption to the transmitted symbol. Hence the mean of the conditional distribution at receiver is decided by the encoder while the variance is dictated by the channel. Compared to the autoencoder based design in existing literature, this approach provides a new interpretation for the noise corruption happening to the transmitted symbols.

The proposed approach and the new interpretation following it help in deriving objective functions which can include prior information about the channel models or domain-specific information and can be used to train transmitter and receiver jointly. We show that the models trained with loss functions derived based on this interpretation accelerate the training speed. Further, the proposed method is able to recover the objective functions used by previous works under appropriate assumptions. Also, by appropriately choosing the input representation for symbols, we show that deep learning based systems can recover Gray coding through the training process. The main differences in the proposed method when compared to existing AE based models are given in Table I. In summary, this work introduces a deep learning based method for end to end design of communication systems which can systematically handle noise corruption of transmit symbols. The results show that the proposed method can produce consistently better models fast when compared to previous works.

A. Notations

Bold face lower-case letters (eg. $\mathbf{x}$) denote column vector. Bold face upper-case letters (eq. $\mathbf{X}$) denote matrix. Script face letters (eg. $\mathcal{S}$) denotes a set, $|\mathcal{S}|$ denotes the cardinality of the set $\mathcal{S}$. $f(\mathbf{x};\boldsymbol{\theta})$ represents a function which takes in a vector $\mathbf{x}$ and has parameters $\boldsymbol{\theta}$. $\mathcal{D}_{KL}(p(X)||q(Y))$ denotes KL divergence between random variables X and Y with distributions $p(\cdot)$ and $q(\cdot)$ respectively. $p_\theta(\cdot)$ represents a distribution with parameters θ. $\mathbb{E}_p$ is the expectation operator with respect to distribution p. $\mathbf{0}_m$ represents an all zero vector of length m, $\boldsymbol{I}_m$ represents identity matrix of dimension $m \times m$. $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate Gaussian with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The trace of a matrix $\mathbf{A}$ is denoted by $tr(\mathbf{A})$.

II. End to End Modeling of Communication Systems

A communication system can be seen as a model that recreates a copy of the message which is sent by the transmitter at the receiver end. Let $\mathbf{x} \in \mathcal{X}$ be the information to be sent from the transmitter. Modern communication systems convert the data $\mathbf{x}$ to a representation $\mathbf{z} \in \mathcal{Z}$ which is suitable for transmission over a noisy channel. A corrupted version of $\mathbf{z}$, denoted by $\hat{\mathbf{z}}$ is received at the destination. The receiver tries to recover the best possible reconstruction of $\mathbf{x}$ from the observed $\hat{\mathbf{z}}$.

The transmitter can be viewed as a function which takes in the information $\mathbf{x}$ and computes the intermediate representation $\mathbf{z}$ as $\mathbf{z} = f(\mathbf{x})$. The channel which corrupts $\mathbf{z}$ can be represented as $\hat{\mathbf{z}} = h(\mathbf{z})$. Here $h(\cdot)$ is a stochastic function which when applied on $\mathbf{z}$ gives output $\hat{\mathbf{z}}$. Finally the receiver can be characterized as another function which computes the best possible reconstruction of $\mathbf{x}$ from $\hat{\mathbf{z}}$ as $\hat{\mathbf{x}} = g(\hat{\mathbf{z}})$.

Following [4], we can model a communication system as an autoencoder. The transmitter function is represented using a neural network parameterized by $\boldsymbol{\theta}_T$ such that $\mathbf{z} = f(\mathbf{x}; \boldsymbol{\theta}_T)$ and the receiver function is represented using another neural network parameterized by $\boldsymbol{\theta}_R$ such that $\hat{\mathbf{x}} = g(\hat{\mathbf{z}}; \boldsymbol{\theta}_R)$. However, the channel function $h(\cdot; \boldsymbol{\theta}_C)$, is typically unknown in a communication system and is generally considered as a stochastic mapping from $\mathbf{z}$ to $\hat{\mathbf{z}}$. This channel function models both the hardware imperfections in the system as well as the channel impairments. Hence the communication system can be represented as

$$\mathbf{z} = f(\mathbf{x}; \boldsymbol{\theta}_T) \tag{1}$$

$$\hat{\mathbf{z}} = h(\mathbf{z}; \boldsymbol{\theta}_C) \tag{2}$$

$$\hat{\mathbf{x}} = g(\hat{\mathbf{z}}; \boldsymbol{\theta}_R) \tag{3}$$

A schematic representation of the mentioned design using neural network function approximators is provided in Fig. 1.

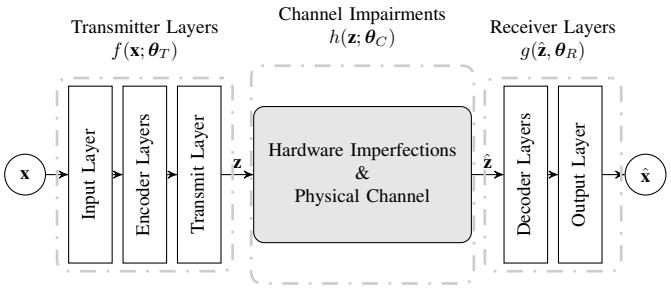


Fig. 1: Autoencoder based Communication System

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ represent the collection of input symbols and $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i\}_{i=1}^N$ represent the collection of decoded symbols. The goal of an end to end communication system design is to find the parameters $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_R$ such that

$$\boldsymbol{\theta}_T, \boldsymbol{\theta}_R = \underset{\boldsymbol{\theta}_T, \boldsymbol{\theta}_R}{\arg\max}\, \mathcal{G}(\mathbf{X}, \hat{\mathbf{X}}) \tag{4}$$

where $\mathcal{G}(\mathbf{X}, \hat{\mathbf{X}})$ is a gain function which calculates how well the system is able to reconstruct the message in dataset $\mathbf{X}$. Note that channel parameter $\boldsymbol{\theta}_C$ is not a learnable parameter and hence not the part of the optimization objective as it is dictated by channel. Previous works [4]–[6], [8] used one-hot encoding to represent the message symbols $\mathbf{x}$ and the gain is calculated based on categorical cross-entropy over all the training samples. That is, $\mathcal{G}(\mathbf{X}, \hat{\mathbf{X}}) = \sum_{\mathbf{x} \in \mathbf{X}} \log(p_{\mathbf{x}})$, where $p_{\mathbf{x}} = p_{\mathbf{x}}(\hat{\mathbf{x}})$ corresponds to the normalized (to 1) score given to the message $\mathbf{x}$ from the output softmax layer.

In the following section, we discuss how to capture the latent code corruption by channel into the model using the principles of variational inference and use the developments in the generative modeling capabilities of auto-encoder networks for simultaneously training the transmitter and the receiver.

III. Variational Inference perspective

Efficient reconstruction $\hat{\mathbf{x}}$ of message $\mathbf{x}$ from the received representation $\hat{\mathbf{z}}$ at receiver can be achieved if full knowledge of channel is available. However, the stochastic nature of channel function $h(\cdot)$ and the lack of knowledge of the channel parameters $\boldsymbol{\theta_C}$ makes this goal challenging. The joint density of the data that is transmitted $\mathbf{x}$ and the received signal $\hat{\mathbf{z}}$ can be represented as

$$p(\mathbf{x}, \hat{\mathbf{z}}) = p(\mathbf{x})p(\hat{\mathbf{z}}|\mathbf{x}) = p(\mathbf{x})p_{\boldsymbol{\theta_C}}(\hat{\mathbf{z}}|\mathbf{z}), \tag{5}$$

where we assume that transmitter provides a deterministic mapping from $\mathbf{x}$ to $\mathbf{z}$. However, in an unknown channel scenario, the conditional density $p_{\boldsymbol{\theta_C}}(\hat{\mathbf{z}}|\mathbf{z})$, and in turn $p(\hat{\mathbf{z}}|\mathbf{x})$, is unknown.

A. Graphical model for communication problem

The problem of reliable communication can be cast into a graphical model as shown in Fig. 2. Here, $\mathbf{x}$ is the data and $\mathbf{z}$ is the corresponding representation to be transmitted over the channel. Here, we follow the standard plate notation of graphical models; variables $(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{z}, \hat{\mathbf{z}})$ are repeated N times, while the parameters $(\boldsymbol{\phi}, \boldsymbol{\theta})$ takes only a single realization in the problem. We use $\boldsymbol{\phi} = \{\boldsymbol{\theta}_T, \boldsymbol{\theta}_C\}$ to represent the parameters of the encoding process. In the graphical model, this is represented as $\mathbf{z}$ being influenced by $\mathbf{x}$ and $\boldsymbol{\phi}$. The decoder with parameters $\boldsymbol{\theta} = \boldsymbol{\theta}_R$ acts on received representation $\hat{\mathbf{z}}$ and produces a reconstruction of data.

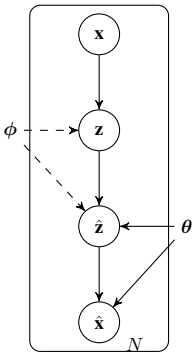


Fig. 2: Graphical model of proposed system

The main aim of a communication system is to identify the stochastic mapping of channel, from $\mathbf{z}$ to $\hat{\mathbf{z}}$, and develop methods to retrieve the data $\mathbf{x}$. In practical systems, it is often the case that the stochastic mapping of channel is unknown and the distribution is difficult to compute.

Variational Inference (VI) is a method from statistical learning for approximating difficult to compute probability densities [19]. VI deals with finding the conditional distribution of latent variables $\hat{\mathbf{z}}$ given $\mathbf{x}$. Considering the joint density $p(\mathbf{x}, \hat{\mathbf{z}}) = p(\mathbf{x})p(\hat{\mathbf{z}}|\mathbf{x})$, *inference* in a Bayesian model amounts to conditioning on data and computing the posterior $p(\hat{\mathbf{z}}|\mathbf{x})$. Variational Inference applies optimization techniques to approximate this conditional density.

Recent developments in deep learning proposed the use of variational inference for generative modeling. Generative modeling refers to the process of producing valid samples from $p(\mathbf{x})$. Consider the graphical model given in Fig. 3. Here, samples of $\mathbf{x}$ are generated from a latent variable $\hat{\mathbf{z}}$ and associated parameters represented by $\boldsymbol{\theta}$. The solid lines denote

the generative model $p_{\boldsymbol{\theta}}(\hat{\mathbf{z}})p_{\boldsymbol{\theta}}(\mathbf{x}|\hat{\mathbf{z}})$. To generate valid samples of $\mathbf{x}$, we first sample $\hat{\mathbf{z}}$ and then use $\hat{\mathbf{z}}$ and $\boldsymbol{\theta}$ to generate $\mathbf{x}$. The dashed lines represent the inference procedure with variational approximation of the intractable posterior $p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\mathbf{x})$.

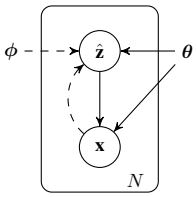


Fig. 3: Graphical model of relationship between variables

In [18], a stochastic optimization based method is proposed applying deep learning to first approximate the inference $p(\hat{\mathbf{z}}|\mathbf{x})$ with appropriate prior on $p(\hat{\mathbf{z}})$ using an encoder network $q_\phi(\hat{\mathbf{z}}|\mathbf{x})$. Then, a decoder network $p_\theta(\mathbf{x}|\hat{\mathbf{z}})$ is used to compute the reconstruction $\hat{\mathbf{x}}$ of message $\mathbf{x}$ from $\hat{\mathbf{z}}$. Here $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are the neural network parameters that will be learned during the training phase. Given high capacity model (ie., neural networks with sufficient learning capability), and good prior distribution $p(\hat{\mathbf{z}})$, the model will approximate the posterior ie., $q_\phi(\hat{\mathbf{z}}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\hat{\mathbf{z}}|\mathbf{x})$. Because of the encoder-decoder structure present in the model, this method is generally known as *Auto-encoding Variational Bayes (AVB)*.

The expected marginal likelihood $p_\theta(\mathbf{x})$ of datapoint $\mathbf{x} \in \mathcal{X}$, under an encoding function, $q_\phi(\cdot)$, can be computed as

$$\mathbb{E}_{p(\mathbf{x})} \log p_\theta(\mathbf{x}) = \mathcal{L}_{\theta,\phi}(\mathbf{x}) + \mathbb{E}_{p(\mathbf{x})}\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p_\theta(\hat{\mathbf{z}}|\mathbf{x})), \tag{6}$$

where

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}\left(\log \frac{p_\theta(\mathbf{x},\hat{\mathbf{z}})}{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}\right) \tag{7}$$

is commonly referred as *Evidence Lower Bound* (ELBO) or *Variational Lower Bound* and

$$\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p_\theta(\hat{\mathbf{z}}|\mathbf{x})) = \mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}\left(\log \frac{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}{p_\theta(\hat{\mathbf{z}}|\mathbf{x})}\right) \tag{8}$$

is the KL-divergence between the approximating and actual distributions. Please see Appendix A for details on (6). By re-arranging (6) and noting that $D_{KL}(Y_1||Y_2) \geq 0$ for any two random variables $\{Y_1, Y_2\}$, we can see that $\log p_\theta(\mathbf{x}) \geq \mathcal{L}_{\theta,\phi}(\mathbf{x})$. Therefore, the likelihood of reconstruction $\log p_\theta(\mathbf{x})$ is lower bounded by (7) (hence the name *ELBO*). Since it is difficult to compute the value of $p_\theta(\hat{\mathbf{z}}|\mathbf{x})$, Variational Inference tries to maximize this alternative quantity $\mathbb{E}_{p(\mathbf{x})} \log p_\theta(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x})}\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p_\theta(\hat{\mathbf{z}}|\mathbf{x}))$ by maximizing the ELBO $\mathcal{L}_{\theta,\phi}(\mathbf{x})$.

Following from (7), the maximization objective ELBO $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ can be re-arranged as

$$\begin{aligned}
&\mathcal{L}_{\theta,\phi}(\mathbf{x}) \\
&\quad= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{x},\hat{\mathbf{z}}) - \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log q_\phi(\hat{\mathbf{z}}|\mathbf{x}) \\
&\quad= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log \left(p_\theta(\mathbf{x}|\hat{\mathbf{z}})p(\hat{\mathbf{z}})\right) - \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log q_\phi(\hat{\mathbf{z}}|\mathbf{x}) \\
&\quad= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{x}|\hat{\mathbf{z}}) - \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \left(\log \frac{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}{p(\hat{\mathbf{z}})}\right) \\
&\quad= \underbrace{\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{x}|\hat{\mathbf{z}})}_{\text{reconstruction likelihood}} - \underbrace{\mathbb{E}_{p(\mathbf{x})}\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p(\hat{\mathbf{z}}))}_{\text{KL loss}}.
\end{aligned} \tag{9}$$

Hence, the objective of maximizing ELBO is equivalent to maximizing the penalized likelihood of reconstruction of $\mathbf{x}$ from $\hat{\mathbf{z}}$ where is the penalty is the KL-divergence between the inference density approximation $q_\phi(\hat{\mathbf{z}}|\mathbf{x})$ and assumed prior $p_\theta(\hat{\mathbf{z}})$.

From Fig. 1, $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_R$ are the only learnable parameters in this system and $\boldsymbol{\theta}_C$ represents the unknown parameters of the channel along with the stochastic channel function $h(\cdot)$. From the model presented in Fig. 2, we have $\boldsymbol{\phi} = \{\boldsymbol{\theta}_T, \boldsymbol{\theta}_C\}$ and $\boldsymbol{\theta} = \boldsymbol{\theta_R}$. In AVB, the encoder network $\mathcal{E}_\phi$ is used to learn the parameters to compute $\mathbf{z}$ from given symbol $\mathbf{x}$. Then, a stochastic channel function is applied on $\mathbf{z}$ to sample $\hat{\mathbf{z}}$ which is used by the decoder network $\mathcal{D}_\theta$ to recreate $\mathbf{x}$. Hence,

$$\mathbf{z} = f(\mathbf{x};\boldsymbol{\theta}_T), = \mathcal{E}_\phi(\mathbf{x}) \tag{10}$$

$$q_\phi(\hat{\mathbf{z}}|\mathbf{x}) = h(\mathbf{z};\boldsymbol{\theta_C}) \tag{11}$$

$$p_\theta(\hat{\mathbf{x}}|\hat{\mathbf{z}}) = g(\hat{\mathbf{z}};\boldsymbol{\theta}_R) \ = \mathcal{D}_\theta(\hat{\mathbf{z}}). \tag{12}$$

The effect of the encoder $\mathcal{E}_\phi$ and the stochastic channel function which together transform the message $\mathbf{x}$ to a representation $\hat{\mathbf{z}}$ which suffered corruption from the channel is approximated by $q_\phi(\hat{\mathbf{z}}|\mathbf{x})$. The output of the decoder $\mathcal{D}_\theta$ is a distribution over all the possible messages computed after observing $\hat{\mathbf{z}}$ and is represented as $p_\theta(\hat{\mathbf{x}}|\hat{\mathbf{z}})$.

Finally, the objective of the optimization problem (4) to train end to end communication system having the model discussed above can be written as

$$\boldsymbol{\theta}_T, \boldsymbol{\theta}_R = \underset{\boldsymbol{\theta}_T,\boldsymbol{\theta}_R}{\arg\max}\, \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}), \tag{13}$$

over all $\mathbf{x} \in \mathbf{X}$, the set of available training points.

B. Reconstruction likelihood

The first term in maximizing objective ELBO (9) accounts for the capability of the end to end system to successfully reproduce the intended message $\mathbf{x}$ at the receiver end. The exact expression for reconstruction likelihood $\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{x}|\hat{\mathbf{z}})$ depends on how the message $\mathbf{x}$ is represented in the system.

Previous works on end to end design of communication systems [4]–[6], [8] used one-hot encoding to represented each message $\mathbf{x} \in \mathcal{X}$. With $|\mathcal{X}| = M$, one-hot encoding uses a vector of length M with all entries 0 except a 1 for the position corresponding to the message. The softmax output layer of the receiver also produces a M length vector, which sums to 1. If this representation of $\mathbf{x}$ is used, the reconstruction term in (9)

takes the form of negative categorical cross entropy and can be written as

$$\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}\log p_\theta(\mathbf{x}|\hat{\mathbf{z}}) = \sum_{\mathbf{x}\in\mathbf{X}}\log(p_{\mathbf{x}}), \tag{14}$$

where $p_{\mathbf{x}}$ corresponds to the normalized (to 1) score given to the message $\mathbf{x}$ by the receiver $\mathcal{D}_\theta(\cdot)$'s softmax output layer.

Another way of representing the message is to directly use the binary representation of the message. For $|\mathcal{X}| = M$, we need a block length of atleast $d = \lceil \log_2 M \rceil$ to represent (uncoded) message $\mathbf{x}$. Under this representation, $\mathbf{x}$ is a vector of length d with multiple entries of 0s and 1s. The output layer of decoder should also be properly modified to output the corresponding values. In this case, a popular choice for output layer activation function is to use sigmoid activation, which assigns a value between 0 and 1 for each of the entries in reconstruction. Hence $p_\theta(\mathbf{x}|\hat{\mathbf{z}})$ becomes a multivariate Bernoulli distribution of length b with element probabilities computed from $\hat{\mathbf{z}}$. The reconstruction likelihood becomes negative of binary cross entropy as in [18] and can be computed as,

$$\begin{aligned}
&\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}\log p_\theta(\mathbf{x}|\hat{\mathbf{z}}) \\
&\quad= \sum_{\mathbf{x}\in\mathbf{X}}\sum_{i=1}^{d}\log p_\theta(x_i|\hat{\mathbf{z}}) = \sum_{\mathbf{x}\in\mathbf{X}}\sum_{i=1}^{d}\log p(x_i;\hat{x}_i) \\
&\quad= \sum_{\mathbf{x}\in\mathbf{X}}\sum_{i=1}^{d}\left(x_i\log\hat{x}_i + (1-x_i)\log(1-\hat{x}_i)\right). \qquad (15)
\end{aligned}$$

While one-hot representation with categorical cross entropy is a popular choice of loss function for classification tasks, the binary message representation with binary cross entropy is scalable to a learn for a very large number of messages [1]. One should select the appropriate representation for messages while keeping these constraints in mind. In Sec IV, we show that by using (15) instead of (14) on (9), the models can be taught the concept of *Gray Coding* without any other explicit criterion.

Note that (9) composes of two terms and in the succeeding subsections, we discuss the second term and its impact. Also, note that when the second term in (9) is a constant, the first term will be the optimization objective and we recover the results in [4]–[6], [8].

C. KL-loss for AWGN channel

The Additive White Gaussian Noise (AWGN) channel is a widely used channel model to represent the corruption incurred to the transmitted signal in communication systems. For a $\mathbf{z}$ of dimensions m, Gaussian corruption with noise power σ_n^2 per component is modeled as

$$\hat{\mathbf{z}} = \mathbf{z} + \boldsymbol{n}, \tag{16}$$

where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}_m, \sigma_n^2\boldsymbol{I}_m)$; $\mathbf{0}$ is an all zero vector of dimension m and $\boldsymbol{I}$ is an identity matrix of dimension $m \times m$. Taking a Gaussian prior of $p(\hat{\mathbf{z}}) = \mathcal{N}(\mathbf{0}_m, \sigma_0^2\boldsymbol{I}_m)$, the KL Loss in (9) for AWGN channel can be computed as

$$\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p(\hat{\mathbf{z}})) = \frac{1}{2\sigma_0^2}\sum_{j=1}^{m} z_j^2 - \frac{m}{2}\left(1 - \frac{\sigma_n^2}{\sigma_0^2} + \log\frac{\sigma_n^2}{\sigma_0^2}\right). \tag{17}$$

Please refer Appendix B for the derivation. Depending on the representation used for symbols in the model, (17) can be combined with (14) (in case of one-hot representation) or with (15) (in case of binary representation) to get appropriate objective function for training the model in AWGN channel.

Considering the case of one-hot encoding as in [4]–[6], [8], the ELBO objective to be maximized ie., (9) can then be computed as

$$\sum_{\mathbf{x}\in\mathbf{X}}\left(\log(p_{\mathbf{x}}) - \frac{1}{2\sigma_0^2}\sum_{j=1}^{m} z_j^2 + \frac{m}{2}\left(1 - \frac{\sigma_n^2}{\sigma_0^2} + \log\frac{\sigma_n^2}{\sigma_0^2}\right)\right) \tag{18}$$

As the noise power per component σ_n^2 and the prior variance σ_0^2 are constant in the problem, the final objective to maximize can be written as

$$\max_{\boldsymbol{\theta}_T,\boldsymbol{\theta}_R}\left\{\sum_{\mathbf{x}\in\mathbf{X}}\left(\log(p_{\mathbf{x}}) - \frac{1}{2\sigma_0^2}\sum_{j=1}^{m} z_j^2\right)\right\}. \tag{19}$$

The first term in the derived objective (19) is negative of the categorical cross entropy. Previous works in [4]–[6], [8] considered only this term for optimization at a constant training SNR[2]. The second term connects the signal power $\sum_{j=1}^{m} z_j^2$ and noise power to the design. At a specified noise power σ_n^2 per component, maximization of the above objective brings in the concept of using less power for signaling. Hence, the derived objective optimizes the signaling such that a tradeoff is achieved between minimizing the transmit power and maximizing the reconstruction likelihood. If we assume a constant training SNR scenario, the second term becomes a constant and we recover the objective used in [4]–[6], [8].

Comparing the derived objective (19) to the objective used in AE based communication systems design popularized by [4] points to some interesting observations. The main difference between the proposed method and AE based design are that AE based designs use a normalization layer as the last layer in transmitter to control the power used for signaling. By choosing a particular SNR, γ, to train at, the objective of these models is to maximize the reconstruction likelihood alone. Let σ_n^2 be the noise power per component of the transmission from the channel and m be the number of components. Then the objective to optimize, with power constraint from the normalization layer, becomes,

$$\begin{aligned}
\max_{\boldsymbol{\theta}_T,\boldsymbol{\theta}_R} \quad & \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q(\hat{\mathbf{z}}|\mathbf{x})}\log p_{\boldsymbol{\theta}_R}(\mathbf{x}|\hat{\mathbf{z}}) \\
\text{sub. to} \quad & \mathbb{E}_{p(\mathbf{x})}\mathbf{z}^T\mathbf{z} = m\sigma_n^2\gamma. \qquad (20)
\end{aligned}$$

[1] While one-hot encoding requires M nodes at the inputs layer, binary representation only requires only $\lceil \log_2 M \rceil$ nodes at inputs.

[2] The SNR in this case is defined as $SNR = \frac{1}{m\sigma_n^2}\sum_{j=1}^{m} z_j^2$.

Introducing Lagrangian multiplier, we can rewrite the above optimization objective as,

$$\max_{\boldsymbol{\theta}_T,\boldsymbol{\theta}_R} \quad \left\{\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q(\hat{\mathbf{z}}|\mathbf{x})}\log p_{\boldsymbol{\theta}_R}(\mathbf{x}|\hat{\mathbf{z}}) - \lambda_L\mathbb{E}_{p(\mathbf{x})}\mathbf{z}^T\mathbf{z} - \lambda_L m\sigma_n^2\gamma\right\}, \tag{21}$$

where λ_L is the Lagrangian multiplier. Removing the problem independent constants, this can be re-written as,

$$\max_{\boldsymbol{\theta}_T,\boldsymbol{\theta}_R} \quad \left\{\mathbb{E}_{p(\mathbf{x})}\left(\mathbb{E}_{q(\hat{\mathbf{z}}|\mathbf{x})}\log p_{\boldsymbol{\theta}_R}(\mathbf{x}|\hat{\mathbf{z}}) - \lambda_L\mathbf{z}^T\mathbf{z}\right)\right\}. \tag{22}$$

Comparing this with the objective derived in (19), we can observe that AE based models [4] are also following a similar objective function to maximize with $\lambda_L = \frac{1}{2\sigma_0^2}$. In other words, while the works in [4]–[6], [8] impose hard constraints, this work imposes a soft constraint.

Recent developments in research to incorporate hard constraints to deep learning problems suggest that imposing a hard constraint on a deep learning problem may not lead to desired performance. The work in [20] suggests that hard constraints should mostly be avoided and instead proposes to use differentiable penalties in loss functions, similar to our approach. In [16], authors show that even though hard constraints bring in nice theoretical benefits, the resulting method can end up being computationally complex (like the addition of normalization layer at the output of the encoder as done in existing works [4]–[6], [8]). Further, the promised benefits may not manifest in practical problems. Later, in the Results section, we show that the proposed method of soft constraints on loss function yields faster training that the hard constraint approach adopted in previous works.

D. KL-loss for Rayleigh Block Fading (RBF) channel

One of the most widely used model to capture the fading effects during signal transmission is Rayleigh Block Fading. Under Rayleigh Block Fading (RBF) model, the corrupted signal $\hat{\mathbf{z}}$ can be modeled as

$$\hat{\mathbf{z}} = h\mathbf{z} + \boldsymbol{n}, \tag{23}$$

where $h \sim \mathcal{CN}(0,1)$ and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}_m, \sigma_n^2\boldsymbol{I}_m)$ or equivalently [6]

$$\hat{\mathbf{z}} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{2}\left(\mathbf{z}\mathbf{z}^T - \mathbf{J}\mathbf{z}\mathbf{z}^T\mathbf{J}\right) + \sigma_n^2\boldsymbol{I}_m\right), \tag{24}$$

where $\mathbf{J}$ is the matrix defined by $\mathbf{J} = \begin{bmatrix}\mathbf{0}_{m/2} & -\boldsymbol{I}_{m/2}\\ \boldsymbol{I}_{m/2} & \mathbf{0}_{m/2}\end{bmatrix}$ with $\mathbf{0}_{m/2}$ is square zero matrix of dimension $m/2$ and $\boldsymbol{I}_{m/2}$ identity matrix of dimension $m/2$ [3]. If the only knowledge we have about the channel is that it can be well modeled by a distribution with finite variance, then the prior choice should reflect this information. In this context, a normal prior is the maximum entropy prior. Hence, taking a prior of $p(\hat{\mathbf{z}}) = \mathcal{N}(\mathbf{0}_m, \sigma_0^2\boldsymbol{I}_m)$, the KL Loss in (9) for this case can be computed as

$$\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p(\hat{\mathbf{z}})) = \frac{1}{2\sigma_0^2}\sum_{j=1}^{m} z_j^2 - \frac{m}{2}\left(1 - \frac{\sigma_n^2}{\sigma_0^2} + \log\frac{\sigma_n^2}{\sigma_0^2}\right) - \log\left(1 + \frac{1}{2\sigma_n^2}\sum_{j=1}^{m} z_j^2\right). \tag{25}$$

Please refer to Appendix C for the detailed derivation. Depending on the representation used for symbols in the model, (25) can be combined with (14) (in case of one-hot representation) or with (15) (in case of binary representation) to get appropriate objective function for training the model in RBF channel.

Considering one-hot encoding and removing the constant terms in the problem, the final ELBO objective (9) to maximize for training an end to end communication system in an RBF channel can be written as

$$\max_{\boldsymbol{\theta}_T,\boldsymbol{\theta}_R}\left\{\sum_{\mathbf{x}\in\mathbf{X}}\left(\log(p_{\mathbf{x}}) - \frac{1}{2\sigma_0^2}\sum_{j=1}^{m} z_j^2 + \log\left(1 + \frac{1}{2\sigma_n^2}\sum_{j=1}^{m} z_j^2\right)\right)\right\}. \tag{26}$$

This objective is slightly different from the AWGN objective (19) due to an additional term similar to capacity. Similar to the case of AWGN channel objective, we can see that at constant SNR condition, we recover the objective function used in [4]–[6], [8]. Interestingly, in the special case of $m = 2$, the new term in this objective (the third term in (26)) is equivalent to the AWGN channel capacity. Maximizing this objective optimizes the system to improve the channel capacity (third term) while minimizing the signaling energy (second term) and at the same time improving reconstruction loss (first term). This intuitively fits with the objective of communication systems - maximize the capacity while using minimum signaling power.

E. Constellation learning and mutual information

Mutual information between the data to send $\mathbf{x}$ and the received symbol $\hat{\mathbf{z}}$ can be written as,

$$I(\mathbf{X};\hat{\mathbf{Z}}) = \mathbb{E}_{q_\phi(\mathbf{x},\hat{\mathbf{z}})}\log\frac{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}{q_\phi(\hat{\mathbf{z}})}$$

Following this definition, a lower bound on the mutual information can be obtained as

$$\begin{aligned} I(\mathbf{X};\hat{\mathbf{Z}}) &= \mathbb{E}_{q_\phi(\mathbf{x},\hat{\mathbf{z}})}\log\frac{q_\phi(\mathbf{x}|\hat{\mathbf{z}})}{p(\mathbf{x})} \\ &= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}\log\frac{p_\theta(\mathbf{x}|\hat{\mathbf{z}})}{p(\mathbf{x})} + \mathbb{E}_{q_\phi(\hat{\mathbf{z}})}\mathbb{E}_{q_\phi(\mathbf{x}|\hat{\mathbf{z}})}\log\frac{q_\phi(\mathbf{x}|\hat{\mathbf{z}})}{p_\theta(\mathbf{x}|\hat{\mathbf{z}})} \\ &= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}\log p_\theta(\mathbf{x}|\hat{\mathbf{z}}) + \mathbb{E}_{p(\mathbf{x})}\log\frac{1}{p(\mathbf{x})} + \mathbb{E}_{q_\phi(\hat{\mathbf{z}})}\mathcal{D}_{KL}\left(q_\phi(\mathbf{x}|\hat{\mathbf{z}})||p_\theta(\mathbf{x}|\hat{\mathbf{z}})\right) \\ &\geq \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}\log p_\theta(\mathbf{x}|\hat{\mathbf{z}}). \end{aligned} \tag{27}$$

[3]Note that while implementing in DNN, we split complex $\mathbf{z}$ in to real and imaginary components and stack them into a column vector of dimension m. Hence m is always even in the model.

Similarly, an upper bound on mutual information can be obtained as,

$$\begin{aligned} I(\mathbf{X};\hat{\mathbf{Z}}) &= \mathbb{E}_{q_\phi(\mathbf{x},\hat{\mathbf{z}})} \log \frac{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}{q_\phi(\hat{\mathbf{z}})} \\ &= \mathbb{E}_{q_\phi(\mathbf{x},\hat{\mathbf{z}})} \log \frac{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}{p(\hat{\mathbf{z}})} + \mathbb{E}_{q_\phi(\mathbf{x},\hat{\mathbf{z}})} \log \frac{p(\hat{\mathbf{z}})}{q_\phi(\hat{\mathbf{z}})} \\ &= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log \frac{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}{p(\hat{\mathbf{z}})} - \mathbb{E}_{q_\phi(\hat{\mathbf{z}})} \log \frac{q_\phi(\hat{\mathbf{z}})}{p(\hat{\mathbf{z}})} \\ &= \mathbb{E}_{p(\mathbf{x})}\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p(\hat{\mathbf{z}})) - \mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}})||p(\hat{\mathbf{z}})) \\ &\leq \mathbb{E}_{p(\mathbf{x})}\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p(\hat{\mathbf{z}})). \end{aligned} \tag{28}$$

Comparing the ELBO objective used to train the proposed system (9) and the bounds derived in (27) and (28), we can observe that the objective (9) simultaneously tries to maximize a lower bound of mutual information and minimize an upper bound of the same. The weight given to the objective of minimizing the upper bound can be controlled by the parameter σ_0^2 and hence, in the training process, more importance can be given to maximizing the lower bound. As discussed previously, the AE-based end-to-end communication systems also have similar objective function as ours and hence they also follow similar procedure of maximizing lower bound on mutual information and minimizing a weighted upper bound. Note, it is the upper bound minimizing term which brings in the concept of power control to the model. By controlling the mutual information between $\hat{\mathbf{z}}$ and $\mathbf{x}$, the models avoid trivially scaling the transmit symbols to avoid the effect of channel distortion. However, AE based methods [4]–[6] having hard constraint place more emphasis on minimizing the upper bound on mutual information.

F. Discussion

In this section, we presented an approach for end to end designing of communication systems based on the principles of variational inference and the recent developments in generative modeling with deep neural networks. We showed how any prior information about the channel, either in the form of channel parameters or in the functional form of the channel, can be appropriately incorporated for designing the objective function for optimization through (9). We also provided two examples, with the case of AWGN and RBF channel models. Previous works had to include an additional normalization layer at the transmitter output to control the power of the transmit symbols, which otherwise can become very high. This is because, the objective functions used by the learning agents in those works have no incentive for controlling the transmit power. However, our proposed method yields objective functions which implicitly take care of transmit power control and hence eliminate the need for an additional normalization layer at the transmitter output.

Generalizing beyond AWGN and RBF channel models, the method we proposed in this section can be applied to additive non-Gaussian noise channels as well as other generalized fading channel models using suitable prior. In the scenarios where such additional knowledge of the channel is available, the KL-loss in (9) has to be computed with appropriate prior to obtain the objective function for training. As an example for other noise scenarios, we derive the loss function for Additive Independent Laplace Noise (AILN) environments in Appendix D, for Additive Independent Cauchy Noise (AICN) in Appendix E and give performance results for the same.

IV. Results

In this section, we describe and report results based on simulation studies to validate the analysis and design of the proposed method. We compare the proposed method with existing works in both traditional and deep learning based approaches. For the purpose of evaluation, we consider three cases:

1) 2 bit block with one complex channel use ($M = 4, m = 2$). This scheme is similar to the QPSK scheme which uses one constellation point in complex channel plane to represent 2 bits.
2) 4 bit block with two complex channel uses ($M = 16, m = 4$).
3) 8 bit block with four complex channel uses ($M = 256, m = 8$).

All the schemes are evaluated in both AWGN and RBF channel models. We compare the performance of trained models with traditional methods of QAM and Agrell sphere packing [21] and deep learning based method proposed in [4]. For deep learning based methods, 100 models are trained and the results are reported.

We use two metrics to compare the capabilities of the schemes.

1) Block Error Rate (BLER): The block error rate performance over a wide range of SNR of the schemes will show the usefulness of the schemes in delivering the information over the channel.
2) Packing Density: Another metric to compare the efficiency of multiple signaling methods is to compare the packing density of the transmit signals over the dimensions specified by the number of channel uses. Normalized second moment (E_n) of the transmit symbols $\mathbf{z}$ is defined as [21]

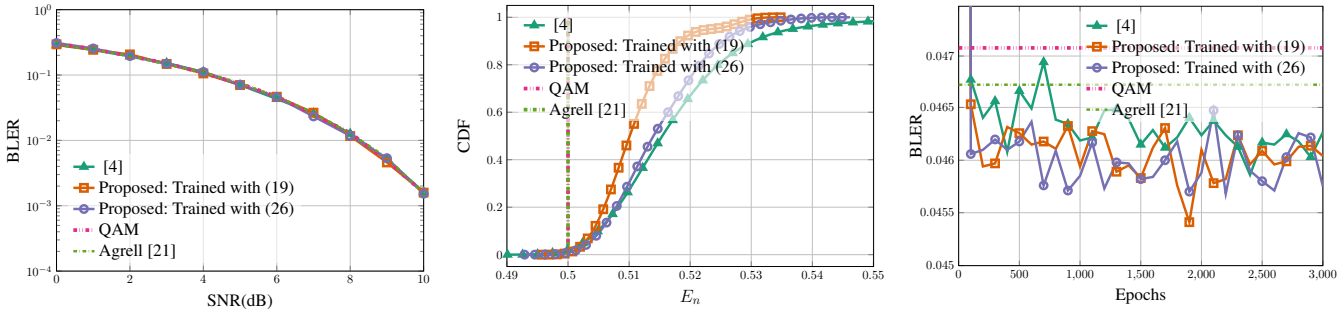
$$E_n = \frac{1}{M}\frac{1}{d_{min}^2}\sum_{i=1}^{M} \mathbf{z}_i^T\mathbf{z}_i, \tag{29}$$

where $d_{min}^2 = \min_{i\neq j}(\mathbf{z}_i - \mathbf{z}_j)^T(\mathbf{z}_i - \mathbf{z}_j)$ is the square of minimum euclidean distance between transmit points. This metric is insensitive to scaling and hence useful to compare packing densities. Smaller the value of E_n, better the packing density.

Please refer Appendix F for more details about the simulation setup and the training procedure.

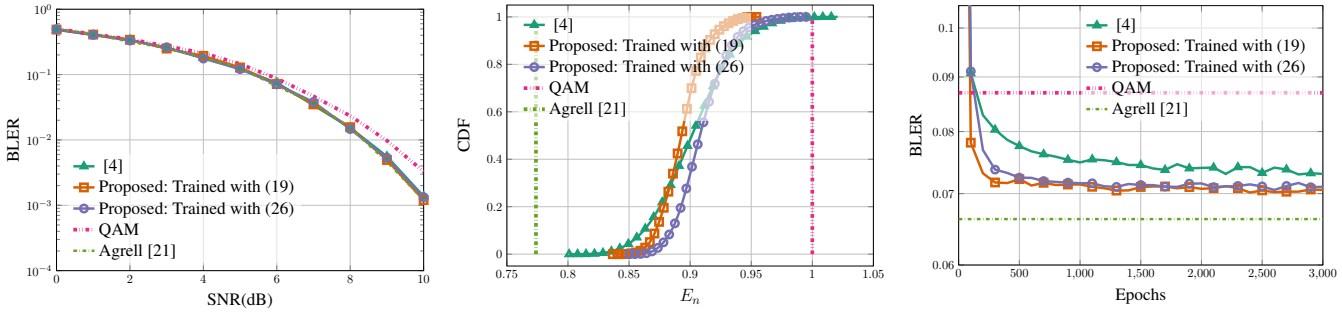
A. DNN architecture

We consider a feedforward autoencoder architecture with three hidden dense layers for encoder network and three hidden dense layers for decoder for all the experiments and both the DL methods under comparison for fairness. The network architecture details are given in Table II.



(a) BLER of best model (b) Distribution of E_n (c) Evolution of BLER at $6dB$

Fig. 4: Results for model with $M = 4, m = 2$ in AWGN channel



(a) BLER of best model (b) Distribution of E_n (c) Evolution of BLER at $6dB$

Fig. 5: Results for model with $M = 16, m = 4$ in AWGN channel

TABLE II: Details of DNN architecture

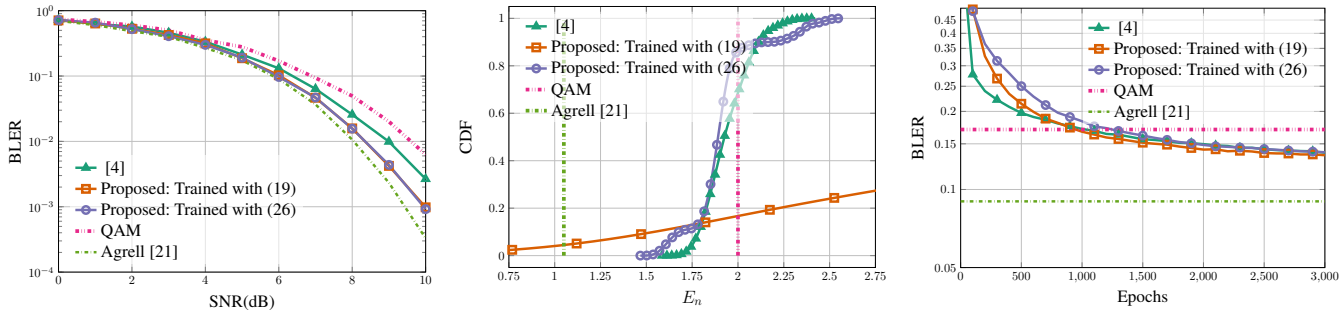
	Layer Name	Size	Activation Function
Transmitter (Encoder)	Input Layer	M	-
	Hidden $E1$	64	ReLU
	Hidden $E2$	32	ReLU
	Hidden $E3$	16	ReLU
	Transmit Layer	m	Linear for Proposed Linear + BN for [4]
Channel			(16) for AWGN channel (23) for RBF channel
Receiver (Decoder)	Hidden $D1$	16	ReLU
	Hidden $D2$	32	ReLU
	Hidden $D3$	64	ReLU
	Ouput Layer	M	Softmax

Selection of activation functions for the network layers impact both the quality of the solution as well as the convergence properties of the model. Traditional activation functions including sigmoid, tanh restrict the activations to be in the range of $[0, 1]$ and $[-1, +1]$ respectively with saturating effects near the boundaries. These saturation effects can hinder gradient propagation through the layers. Recent works applying deep learning for communication systems modeling advocate the use of advanced activation functions like Rectified Linear Units (ReLU) [4], [5], Exponential Linear Units (ELU) [6] etc. We use ReLU for activation at our hidden layers, linear activation at the output of the encoder network and a softmax layer for output of the decoder network. The works in [4]–[6], [8] used a Batch Normalization (BN) layer at the output of the transmitter to control the power of the transmitted constellation. If this layer is not included, the model will try to transmit at uncontrollably higher powers to minimize the cross-entropy loss. However, the objective functions presented in this work, (19) and (26), includes an additional term to minimize the transmit power. Hence, the deep learning model is incentivized for doing power control at the learning phase and will control the constellation power according to the noise it observed and reconstruction likelihood during training. We used $\sigma_0^2 = 1.0$ and $\sigma_n^2 = 0.1$ while training the proposed model. Adam optimizer [22] with learning rate 0.01, $\beta_1 = 0.99$ and $\beta_2 = 0.999$ is used for training all models and each model is trained for 3000 epochs. The models using [4] are trained at an SNR of $10dB$.

B. *Evaluation in AWGN channel*

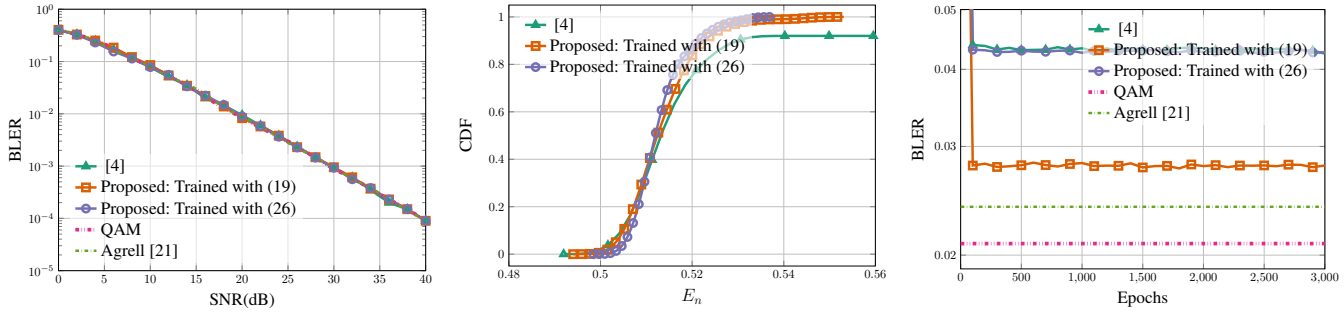
The proposed method is evaluated in AWGN channel model given by (16). In this case, the objective function to optimize is given in (19). However, in a practical scenario, we would like to train the model without any assumptions on the channel model. To cover this case, we also provide results using the objective function developed assuming RBF channel (26). The results for different configurations under test are given in Fig. 4 - 6.

The BLER vs SNR performance of the models are given in Fig. 4a, Fig. 5a and Fig. 6a. Agrell [21] being the optimized sphere packing scheme found using search is able to perform better in all cases. Note that, in the case of one complex channel use, both Agrell and QAM scheme are the same. As the number of channel uses increases, the dimension of the sphere packing problem also increases, and it can be seen



(a) BLER of best model (b) Distribution of E_n (c) Evolution of BLER at $6dB$

Fig. 6: Results for model with $M = 256, m = 8$ in AWGN channel



(a) BLER of best model (b) Distribution of E_n (c) Evolution of BLER at $16dB$

Fig. 7: Results for model with $M = 4, m = 2$ in RBF channel

that the QAM scheme does not perform as good as the other methods in comparison and the gap between the performance of Agrell scheme and QAM scheme widens with an increase in the number of channel uses.

In all the cases, we can see that deep learning methods perform better than traditional QAM methods and are able to perform very close to the optimized Agrell schemes. Even with deep learning models, the performance compared to Agrell scheme widens as the dimension increases. Interestingly, both (19) and (26) provides equally good BLER performance in AWGN channel.

The distribution of surrogate metric for packing density ,E_n given by (29), for the trained models are given in Fig. 4b, Fig. 5b and Fig. 6b. We use kernel density estimation to smoothen the empirical histogram for packing density. In the case of single channel use ($M = 4, m = 2$), traditional QAM and Agrell schemes are the optimal sphere packing schemes (with $E_n = 0.5$) and DL methods are able to reach close to this. In the case of higher dimensions (Fig. 6b) , we can observe that the proposed objective function (26) is able to produce models with better E_n than traditional QAM approximately 90% of the instances while the procedure in [4] managed to produce such models only 70% of the time. From all the results, we can conclude that even though (26) is developed for RBF channel, it can used in AWGN channel as well.

Interesting observations emerge when we analyze the evolution of BLER of the models during the training phase (Fig. 4c, 5c and 6c). It can be observed that, in lower dimensions, the proposed loss functions are able to train the models faster than the method in [4], achieving lower BLERs early in the training. However, at higher dimensions (Fig. 6c), the proposed loss functions slightly lags behind the method in [4] but is eventually able to provide better BLER. These advantages during the training phase can be conclusively attributed to the new loss function developed in this work.

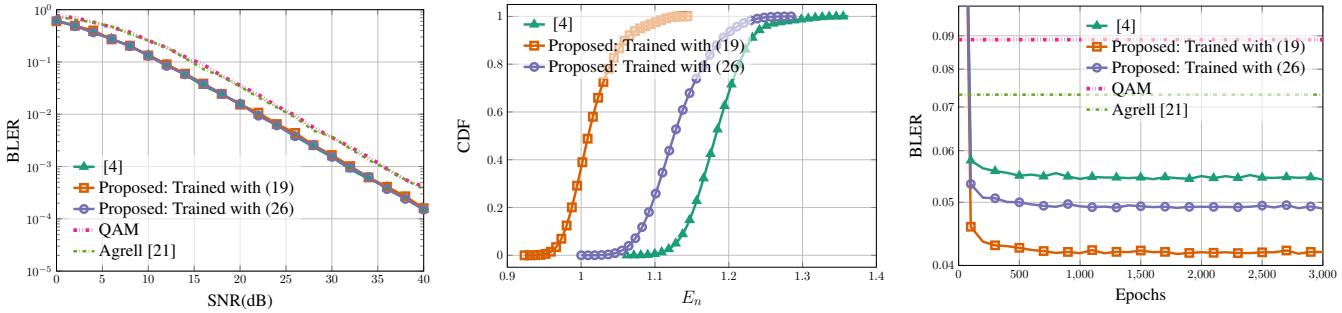
At high dimensions, usage of (19) results in high variability of packing density among trained models, as seen from Fig. 6b. Even though this presents a difficulty in using these method at high dimensions, objective discussed in (26) is able to train better models consistently when compared with [4].

C. Evaluation in RBF channel

For verifying the performance of the methods in the RBF channel model, the model described in (23) is used. We provide the results of optimizing the DNN using both the objectives, (19) and (26), in Fig. 7 - 9.

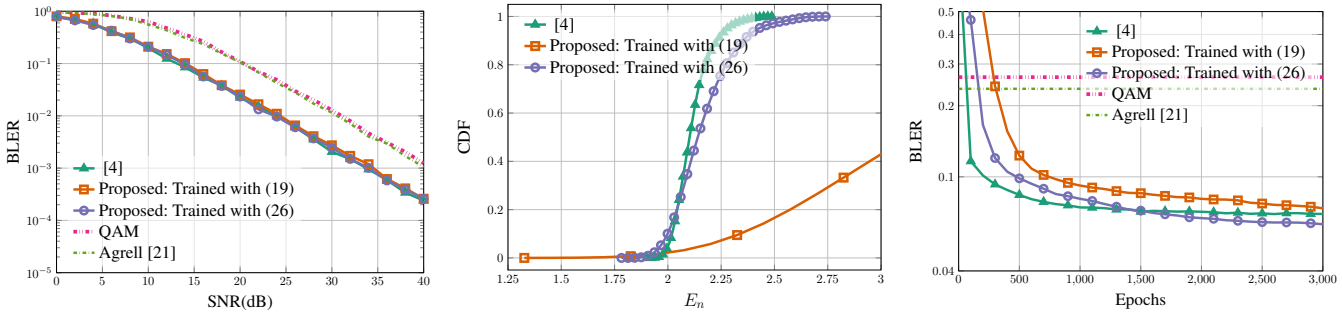
We need to used pilot symbols to obtain an estimate of channel coefficient h and the equalization is done prior to decoding as done in [6], [10]. The estimate of h obtained from pilot symbols affects decoding performance through noisy equalization. We used the same power per component as the constellation points to transmit the pilot symbol such that both the pilot components and the symbol components in the block experience the same SNR during transmission.

Traditional QAM and Agrell schemes are not optimized for RBF channels. As the number of channel uses increases, we can see that the DL methods are able to perform better than QAM and Agrell. The improvement in the case of DL



(a) BLER of best model (b) Distribution of E_n (c) Evolution of BLER at $16dB$

Fig. 8: Results for model with $M = 16, m = 4$ in RBF channel



(a) BLER of best model (b) Distribution of E_n (c) Evolution of BLER at $16dB$

Fig. 9: Results for model with $M = 256, m = 8$ in RBF channel

methods can be attributed to the function approximation power of neural networks which learns to neutralize the effects of noisy channel equalization. Surprisingly, in RBF channel, the models trained with objective derived for AWGN model (19) is able to give performance close to the models using (26). However, the difference when one uses (26) is visible in the packing density of the learned models. At higher dimension (Fig. 9b), (26) is able to consistently produce better models when compared to (19). Although the method in [4] is able to produce models with less variation at higher dimensions, in lower dimensions (Fig. 7b and Fig. 8b), it suffers with high variability. The evolution of BLER of the trained models at $16dB$ is given in Fig. 7c, 8c and 9c. Interestingly, at low dimensions, the objective derived for AWGN channel model performs better that the objective for RBF model. Also, in all cases, the derived loss functions are able to provide a better BLER than the method in [4]. From all these results, we can conclude that using the objective (26) derived for RBF channel model can be expected to produce desired results consistently across different dimensions.

Based on the above, it can be inferred that the proposed method for end to end communication system design

1) Provides a solution which accounts for noise corrupted latent codes with a theoretical backing.
2) Consistently trains better models when compared to existing AE based methods.

Now we investigate the impact of hyperparameter σ_0^2 and the input encoding for better insights into models and constellation labels.

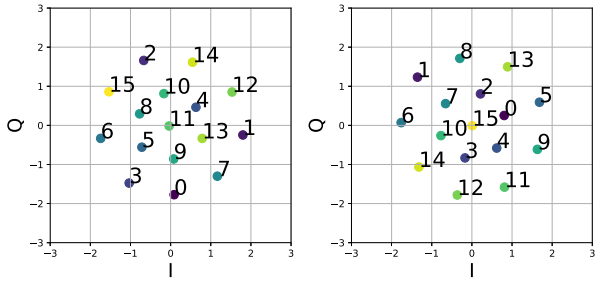
D. Effect of σ_0^2

We used a prior of $p(\hat{\mathbf{z}}) = \mathcal{N}(\mathbf{0}, \sigma_0^2 \boldsymbol{I})$ with variance per component σ_0^2 during the derivation of objective functions (19) and (26). It can be easily seen from these objective functions that σ_0^2 affects the weight given to the transmit symbol power term $\sum_{j=1}^{m} z_j^2$. When prior variance σ_0^2 is low, more weight is given to the transmit symbol power control term to reduce the transmit power and vice versa. However, a very low value of σ_0^2 will aggressively optimize the transmit power such that the constellations learned will have transmit power close to 0. This affects the decoding process and increases the BLER.

Further, the numerical value of σ_0^2 is also related to the noise power σ_n^2. When noise power σ_n^2 is very high, the received symbol $\hat{\mathbf{z}}$ will be heavily distorted and hence a meaningful reconstruction of the transmitted symbol is difficult. This requires the models to transmit at higher power for learning to proceed which can be achieved by using higher numerical value for σ_0^2. When noise power is low, the magnitude of σ_0^2 can be set to low value enabling one to use low power designs. Hence, one is required set the value of σ_0^2 proportional to σ_n^2 with $\sigma_0^2 > \sigma_n^2$ to enable learning.

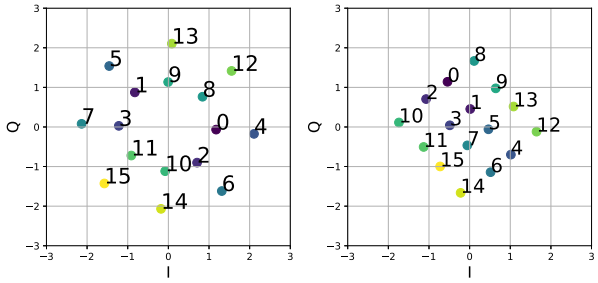
E. Recovering Gray codes

In all the experiments discussed above, we used one-hot encoding to represent the symbols as done in previous works [4]–[6], [8] for comparability. In order to study the structure of constellations, we trained models in AWGN channel for



(a) Sample 1 (b) Sample 2

Fig. 10: Constellations trained for $M = 16, m = 2$ using [4].



(a) Sample 1 (b) Sample 2

Fig. 11: Constellations trained for $M = 16, m = 2$ with (30).

$M = 16$ and $m = 2$ using the method in [4][4]. Two sample constellations learned by the method is given in Fig. 10. It can be observed that the symbols are well arranged in concentric circles maintaining sufficient distance between constellation points. This type of design is useful in optimizing BLER of the system. However, as close-by symbols change by multiple bit positions, this may not be optimized way to design if the system requirement is to improve BER. This constellation characteristic is the effect of choosing one-hot encoding for representing symbols at both input and output as in one-hot encoding, there is no incentive for the model to place symbols with only one bit changes near to each other.

However, by using the binary representation of the symbols and the reconstruction likelihood introduced in (15), the models will be able learn the concept of nearby symbols as the penalization forces all the bit positions to be correct. In this case, the objective function to train models in AWGN channel can be obtained by combining (15) and (17) and can be written as

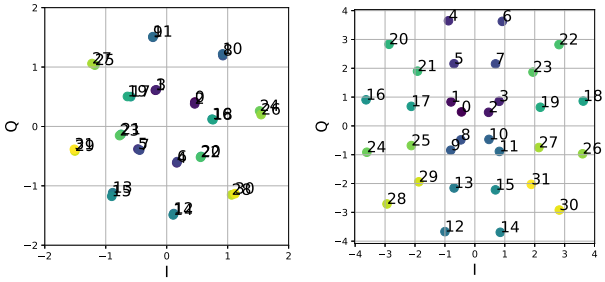
$$\max \left\{ \sum_{i=1}^{d} \left(x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i)\right) - \frac{1}{2\sigma_0^2} \sum_{j=1}^{m} z_j^2 \right\}. \tag{30}$$

As the input layer dimension is now reduced from 16 to 4, we used a small network with hidden layers in encoder having 32 and 16 nodes, decoder having hidden layers with 16 and 32 nodes and finally an output layer of 4 nodes with sigmoid activation function. Training is done for 500 epochs with other settings being similar to the one used in previous experiments. Sample constellations learned by this model is given in Fig. 11.

From the constellations given in Fig 11, it can be easily observed that both the models *learned* the concept of *gray coding*. Symbols are placed in the constellation in such a way that near-by symbols vary by only one bit. After training multiple models, we observed that constellations with concentric circle structure as in Fig. 11a is the most commonly learned structure and the traditional grid-like structure as given in Fig. 11b occurs rarely. This shows that the loss function we use is having multiple local minima resulting in concentric structure and very few local minima resulting in a grid-like structure.

The use of explicit batchnormalization for constraining constellation energy in [4] results in one symbol being placed at point $(0, 0)$ as visible in Fig. 10. This may produce practical difficulties during transmission as a symbol close to $(0, 0)$ is similar to no signal at all. As the method proposed in this work includes constraining the constellation energy into the objective function (30), this problem is not observed in the trained models (As seen in Fig. 11). The placement of a symbol at $(0, 0)$ will result in constellation with center symbol differing in multiple bit positions from the symbols in first concentric circle and suffering a higher reconstruction likelihood with (15). Hence the models learn to avoid such a placement and instead places all symbols on concentric circles in the gray coding scheme.

Interestingly, when the number of symbols increased while keeping the $m = 2$, the model learns to cheat the system by placing two symbols which differ by only one bit top of each other and hence maintaining two concentric circles of the constellation but suffering a higher BLER. A sample constellation when the model is trained using $M = 32$ is given in Fig. 12a.



(a) Learns to cheat ($\sigma_0^2 = 1.0$) (b) With $\sigma_0^2 = 5.0$

Fig. 12: Cheating behavior of model with variation in σ_0^2.

In Fig. 12a, the model learns to place symbols differing in second last bit (Eg: $(0, 2), (29, 31)$). We used a value of $\sigma_0^2 = 1.0$ for this experiment. This cheating behavior can be attributed to the symbol energy control term in objective (30). As discussed before, a low value of σ_0^2 will give more

[4] We chose $m = 2$ for the simplicity of visualization as $m > 2$ is difficult to visualize in 2D. Even though advanced methods like t-SNE can be used for high dimensional visualization and analyzing clustering behavior as done in [4], [6], it is a projection to a 2D plane and may not efficiently covey the placements of points in high dimensional space which we are trying to analyze here.

importance to limiting the constellation transmit power and hence the model learns to place symbols on top of each other while sacrificing reconstruction likelihood. During our experiments, we observed that increasing the value of σ_0^2 also improved the BER performance (as expected, because of more spread out constellation). By adjusting the value of $\sigma_0^2 = 5.0$, the model learns spread out the symbols while maintaining gray coding scheme as shown in Fig.12b. For this particular configuration under test, $M = 32, m = 2$, we observed that increasing σ_0^2 beyond 5.0 does not help in improving bit error rate.

This behavior is observed when more bits are squeezed to transmit per channel use. It can be inferred that σ_0^2 acts as a *honesty* parameter and when forcing the model to pack more bits per channel use, the model needs to have a high value for this parameter to avoid cheating behavior. As both M and σ_0^2 are hyperparameters to be chosen during the system specification, this behavior can be easily handled by appropriately setting the value of σ_0^2 at the design phase.

V. Concluding Remarks

This work proposed a method to perform end to end modeling of communication systems based on the principles of variational inference. Compared to the AE based systems existing in the literature, the proposed method explicitly accounts for the noise corruption of latent codes (transmitted symbols). Further, unlike the AE based works which have a normalization layer leading to hard constraints, we have adopted a soft constraint based approach. The proposed soft constraint approach enables models to explore better during the optimization process. Numerical simulation results show that the proposed method is able to train faster, provides competitive BLER performance and consistently better packing density compared to AE based designs. By modifying the loss function, it is shown that the concepts of gray coding can be learned.

Appendix A
Derivation of log-likelihood of data

This derivation is based on [18]. Noting that $p_\theta(\mathbf{x})$ is constant with respect to $q_\phi(\cdot)$, we have

$$\begin{aligned}
\mathbb{E}_{p(\mathbf{x})} \log p_\theta(\mathbf{x}) &= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log p_\theta(\mathbf{x}) \\
&= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log \left(\frac{p_\theta(\mathbf{x}, \hat{\mathbf{z}})}{p_\theta(\hat{\mathbf{z}}|\mathbf{x})} \right) \\
&= \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log \left(\frac{p_\theta(\mathbf{x}, \hat{\mathbf{z}})}{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \right) + \\
&\qquad \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\phi(\hat{\mathbf{z}}|\mathbf{x})} \log \left(\frac{q_\phi(\hat{\mathbf{z}}|\mathbf{x})}{p_\theta(\hat{\mathbf{z}}|\mathbf{x})} \right) \\
&= \mathcal{L}_{\theta,\phi}(\mathbf{x}) + \mathbb{E}_{p(\mathbf{x})} \mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x}) || p_\theta(\hat{\mathbf{z}}, \mathbf{x})).
\end{aligned}$$

Appendix B
Derivation of objective function for AWGN channel

The KL-divergence between two normal distributions with $\boldsymbol{\mu} \in \mathbb{R}^m$ is given by

$$\begin{aligned}
&\mathcal{D}_{KL}\left(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) || \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\right) \\
&\quad = \frac{1}{2}\left[tr(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) - \right. \\
&\qquad \left. m + \log \frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} \right].
\end{aligned} \tag{31}$$

For AWGN model, we have $q_\phi(\hat{\mathbf{z}|\mathbf{x}}) = \mathcal{N}(\mathbf{z}, \sigma_n^2 \boldsymbol{I}_m)$ and $p(\hat{\mathbf{z}}) = \mathcal{N}(\mathbf{0}_m, \sigma_0^2 \boldsymbol{I}_m)$. Hence the KL Loss term in (9) can be computed as

$$\begin{aligned}
\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x}) || p(\hat{\mathbf{z}})) &= \mathcal{D}_{KL}\left(\mathcal{N}(\mathbf{z}, \sigma_n^2 \boldsymbol{I}_m) || \mathcal{N}(\mathbf{0}_m, \sigma_0^2 \boldsymbol{I}_m)\right) \\
&= \frac{1}{2}\left[Tr\left((\sigma_0^2 \boldsymbol{I}_m)^{-1}\sigma_n^2 \boldsymbol{I}_m\right) + \mathbf{z}^T(\sigma_0^2 \boldsymbol{I}_m)^{-1}\mathbf{z} \right. \\
&\qquad \left. - m + \log \frac{|\sigma_0^2 \boldsymbol{I}_m|}{|\sigma_n^2 \boldsymbol{I}_m|} \right] \\
&= \frac{1}{2}\left[m\frac{\sigma_n^2}{\sigma_0^2} + \frac{1}{\sigma_0^2}\mathbf{z}^T\mathbf{z} - m + m\log\frac{\sigma_0^2}{\sigma_n^2} \right] \\
&= \frac{1}{2\sigma_0^2}\sum_{j=1}^{m} z_j^2 - \frac{m}{2}\left(1 - \frac{\sigma_n^2}{\sigma_0^2} + \log\frac{\sigma_n^2}{\sigma_0^2}\right).
\end{aligned} \tag{32}$$

Appendix C
Derivation of objective function for RBF channel

Noting that $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{0}_m$, $\boldsymbol{\Sigma}_1 = \frac{1}{2}\left(\mathbf{z}\mathbf{z}^T - \mathbf{J}\mathbf{z}\mathbf{z}^T\mathbf{J}\right) + \sigma_n^2 \boldsymbol{I}_m$ and $\boldsymbol{\Sigma}_2 = \sigma_0^2 \boldsymbol{I}_m$, we have

$$\begin{aligned}
&\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x}) || p(\hat{\mathbf{z}})) \\
&\quad = \mathcal{D}_{KL}\left(\mathcal{N}\left(\mathbf{0}_m, \frac{1}{2}\left(\mathbf{z}\mathbf{z}^T - \mathbf{J}\mathbf{z}\mathbf{z}^T\mathbf{J}\right) + \sigma_n^2 \boldsymbol{I}_m \right) || \right. \\
&\qquad \left. \mathcal{N}(\mathbf{0}_m, \sigma_0^2 \boldsymbol{I}_m) \right).
\end{aligned}$$

Simplifying,

$$\begin{aligned}
tr(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) &= tr\left((\sigma_0^2 \boldsymbol{I}_m)^{-1}\left(\frac{1}{2}\left(\mathbf{z}\mathbf{z}^T - \mathbf{J}\mathbf{z}\mathbf{z}^T\mathbf{J}\right) + \sigma_n^2 \boldsymbol{I}_m \right) \right) \\
&= \frac{1}{2\sigma_0^2} tr\left(\mathbf{z}\mathbf{z}^T - \mathbf{J}\mathbf{z}\mathbf{z}^T\mathbf{J}\right) + \frac{\sigma_n^2}{\sigma_0^2} tr(\boldsymbol{I}_m) \\
&= \frac{1}{2\sigma_0^2}\left[tr\left(\mathbf{z}\mathbf{z}^T\right) - tr\left(\mathbf{J}\mathbf{z}\mathbf{z}^T\mathbf{J}\right)\right] + m\frac{\sigma_n^2}{\sigma_0^2} \\
&= \frac{1}{2\sigma_0^2}\left[tr\left(\mathbf{z}\mathbf{z}^T\right) - tr\left(\mathbf{J}\mathbf{J}\mathbf{z}\mathbf{z}^T\right)\right] + m\frac{\sigma_n^2}{\sigma_0^2} \\
&= \frac{1}{2\sigma_0^2}\left[tr\left(\mathbf{z}\mathbf{z}^T\right) + tr\left(\mathbf{z}\mathbf{z}^T\right)\right] + m\frac{\sigma_n^2}{\sigma_0^2} \\
&= \frac{1}{\sigma_0^2}\sum_{j=1}^{m} z_j^2 + m\frac{\sigma_n^2}{\sigma_0^2}.
\end{aligned} \tag{33}$$

Also, we have $\log|\mathbf{\Sigma}_2| = \log|\sigma_0^2\boldsymbol{I}_m| = m\log(\sigma_0^2)$ and

$$\begin{aligned}\log|\mathbf{\Sigma}_1| &= \log\left|\frac{1}{2}\left(\mathbf{z}\mathbf{z}^T - \mathbf{J}\mathbf{z}\mathbf{z}^T\mathbf{J}\right) + \sigma_n^2\boldsymbol{I}_m\right| \\ &= \log\left|\frac{1}{2}\left(\mathbf{z}\mathbf{z}^T + \mathbf{J}\mathbf{z}(\mathbf{J}\mathbf{z})^T\right) + \sigma_n^2\boldsymbol{I}_m\right| \quad (\because \mathbf{J}^T = -\mathbf{J}) \\ &= \log\left(\left(\sigma_n^2\right)^m\left(1 + \frac{1}{\sigma_n^2}\mathbf{z}\mathbf{z}^T + \frac{1}{4\sigma_n^4}\left(\mathbf{z}\mathbf{z}^T\right)^2\right)\right) \\ &= \log\left(\left(\sigma_n^2\right)^m\left(1 + \frac{1}{2\sigma_n^2}\mathbf{z}\mathbf{z}^T\right)^2\right) \\ &= m\log(\sigma_n^2) + 2\log\left(1 + \frac{1}{2\sigma_n^2}\sum_{j=1}^{m} z_j^2\right). \end{aligned} \tag{34}$$

Combining (31), (33) and (34), we get

$$\begin{aligned}&\mathcal{D}_{KL}\left(\mathcal{N}\left(\mathbf{0}_m, \frac{1}{2}\left(\mathbf{z}\mathbf{z}^T - \mathbf{J}\mathbf{z}\mathbf{z}^T\mathbf{J}\right) + \sigma_n^2\boldsymbol{I}_m\right)||\mathcal{N}(\mathbf{0}_m, \sigma_0^2\boldsymbol{I}_m)\right) \\ &= \frac{1}{2}\left[\frac{1}{\sigma_0^2}\mathbf{z}\mathbf{z}^T + m\frac{\sigma_n^2}{\sigma_0^2} - m + m\log(\sigma_0^2)\right. \\ &\qquad \left. -m\log(\sigma_n^2) - 2\log\left(1 + \frac{1}{2\sigma_n^2}\mathbf{z}\mathbf{z}^T\right)\right] \\ &= \frac{1}{2}\left[\frac{1}{\sigma_0^2}\mathbf{z}\mathbf{z}^T - m + m\frac{\sigma_n^2}{\sigma_0^2} - m\log\frac{\sigma_n^2}{\sigma_0^2}\right. \\ &\qquad \left. -2\log\left(1 + \frac{1}{2\sigma_n^2}\mathbf{z}\mathbf{z}^T\right)\right] \\ &= \frac{1}{2\sigma_0^2}\sum_{j=1}^{m} z_j^2 - \frac{m}{2}\left(1 - \frac{\sigma_n^2}{\sigma_0^2} + \log\frac{\sigma_n^2}{\sigma_0^2}\right) \\ &\qquad - \log\left(1 + \frac{1}{2\sigma_n^2}\sum_{j=1}^{m} z_j^2\right). \end{aligned} \tag{35}$$

Appendix D
Training Models in Laplace Noise Environments

Laplace noise distribution is one of the popular noise models used in communication systems to capture the non-Gaussian impulsive behavior of signal corruption [23]–[25]. Laplace noise model has been found useful modeling the signal corruption in cases of indoor and outdoor communications, ultra-wideband wireless systems, multi-user interference, etc (see [24] and references therein). The probability density function of a Laplace random variable with mean μ and variance $2\sigma_n^2$ is defined as [24]

$$\mathcal{L}(x;\mu,\sigma_n) = \frac{1}{2\sigma_n}\exp\left(-\frac{|x-\mu|}{\sigma_n}\right). \tag{36}$$

The KL-divergence between two Laplace distributions can be derived as

$$\begin{aligned}\mathcal{D}_{KL}(\mathcal{L}(x;\mu_1,\sigma_1)||\mathcal{L}(y;\mu_2,\sigma_2)) &= \frac{\sigma_1}{\sigma_2}\left(\exp\left(-\frac{|\mu_1-\mu_2|}{\sigma_1}\right)\right. \\ &\left. - \left(1 - \frac{|\mu_1-\mu_2|}{\sigma_1}\right)\right) + \frac{\sigma_1}{\sigma_2} - 1 - \log\left(\frac{\sigma_1}{\sigma_2}\right). \end{aligned} \tag{37}$$

Following the model in [24], the KL-loss term in (9) can be computed as

$$\begin{aligned}\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p(\hat{\mathbf{z}})) &= \mathcal{D}_{KL}(\mathcal{L}(\hat{\mathbf{z}};\mathbf{z},\sigma_n^2)||\mathcal{L}(\hat{\mathbf{z}};\mathbf{0},\sigma_n^2)) \\ &= \frac{\sigma_n}{\sigma_0}\sum_{i=1}^{m}\exp\left(-\frac{|z_i|}{\sigma_n}\right) - \frac{1}{\sigma_0}\sum_{i=1}^{m}|z_i| - m\log\frac{\sigma_n}{\sigma_0} - m \\ &= \sum_{i=1}^{m}\left(\frac{\sigma_n}{\sigma_0}\left(\exp\left(-\frac{|z_i|}{\sigma_n}\right) - \left(1 - \frac{|z_i|}{m}\right)\right)\right. \\ &\qquad \left. + \frac{\sigma_n}{\sigma_0} - 1 - \log\frac{\sigma_n}{\sigma_0}\right). \end{aligned} \tag{38}$$

This can be upper bounded using [26, Lemma 2.5],

$$\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}}|\mathbf{x})||p(\hat{\mathbf{z}})) \leq \frac{\mathbf{z}^T\mathbf{z}}{2\sigma_n\sigma_0} + \frac{m}{7}\left(\frac{\sigma_0^2}{\sigma_n^2} - 1\right)^2\frac{\sigma_n^2}{\sigma_0^2}. \tag{39}$$

In our experiments, we found that using (39) instead of (38) helps the model to learn fast. We suspect this is because of the L_1 term in (38) making the loss surface difficult to optimize over, while the upper bound in (39) creates a smooth loss surface. Hence the objective function to train the models under Laplace noise can be written as,

$$\max_{\boldsymbol{\theta}_T,\boldsymbol{\theta}_R}\left\{\sum_{\mathbf{x}\in\mathbf{X}}\left(\log(p_{\mathbf{x}}) - \frac{1}{2\sigma_n\sigma_0}\sum_{j=1}^{m} z_j^2\right)\right\}. \tag{40}$$

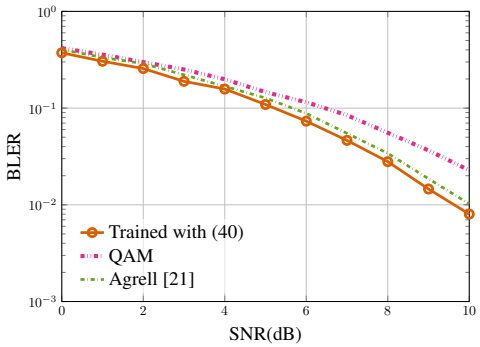


Fig. 13: BLER Performance in Laplace noise for $M = 16, m = 4$.

The BLER performance of the proposed method with traditional methods for $M = 16, m = 4$ scheme is given in Fig. 13. Agrell signalling scheme [21] is designed to be optimal for AWGN noise and in the case of additive Laplace noise, we can see that the proposed method is able to given better BLER performance.

Appendix E
Training Models in Cauchy Noise Environments.

Cauchy noise distribution is a popular choice for modeling impulsive noise in communication systems [27], [28]. However, the undefined nature of first and second moments of Additive Independent Cauchy Noise (AICN) makes the analysis of such systems extremely difficult. The probability

density function of AICN with location parameter δ and scale (dispersion) parameter γ is given by [29]:

$$\mathcal{C}(x;\delta,\gamma) = \frac{1}{\pi}\frac{\gamma}{\gamma^2+(x-\delta)^2}. \tag{41}$$

The KL divergence between two Cauchy distributions can be shown as [30]:

$$\mathcal{D}_{KL}(\mathcal{C}(x;\delta_1,\gamma_1)||\mathcal{C}(x;\delta_2,\gamma_2)) = \log\frac{(\gamma_1+\gamma_2)^2+(\delta_1-\delta_2)^2}{4\gamma_1\gamma_2}. \tag{42}$$

In this experiment, we consider non-isometric Cauchy noise in m-dimensions and the KL-divergence between the received symbol (with dispersion γ_n per component) $q_\phi(\hat{\mathbf{z}};\mathbf{z},\gamma_n) = \mathcal{C}(\hat{\mathbf{z}};\mathbf{z},\gamma_n)$ and 0-location prior with $p(\hat{\mathbf{z}}) = \mathcal{C}(\hat{\mathbf{z}};\mathbf{0},\gamma_0)$ can be derived as:

$$\mathcal{D}_{KL}(q_\phi(\hat{\mathbf{z}};\mathbf{z},\gamma_n)||p(\hat{\mathbf{z}})) = \sum_{i=1}^{m}\log\frac{(\gamma_n+\gamma_0)^2+z_i^2}{4\gamma_n\gamma_0}. \tag{43}$$

Combining this with the objective function derived in (9) and using one-hot encoding, the objective for training models in Cauchy noise can be shown as

$$\max_{\boldsymbol{\theta}_T,\boldsymbol{\theta}_R}\left\{\sum_{\mathbf{x}\in\mathbf{X}}\left(\log(p_{\mathbf{x}}) - \sum_{i=1}^{m}\log\frac{(\gamma_n+\gamma_0)^2+z_i^2}{4\gamma_n\gamma_0}\right)\right\}. \tag{44}$$

As the second moment of Cauchy distribution is undefined, the traditional definition of SNR is not applicable. Geometric SNR (G-SNR) is developed as an alternative to capture the noise strength and for a unit energy transmit symbol, G-SNR is defined as [31]

$$G-SNR = \frac{1}{2C_g}\frac{1}{\gamma^2}, \tag{45}$$

where $C_g \approx 1.78$.

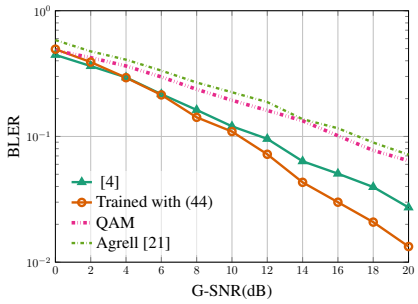


Fig. 14: BLER Performance in Cauchy noise for $M = 16, m = 4$.

The BLER performance of proposed method when compared to traditional constellation designs in non-isometric channel is given in Fig. 14 for the case of $M = 16$ and $m = 4$. The models are trained with $\gamma_0 = 5.00$ as we observed that lower value of prior dispersion adversely affect the learning process. As Cauchy noise is very impulsive in nature, we can see that the BLER is also quite higher than that in other channels like AWGN, RBF etc for traditional constellations of QAM and Agrell. However, we can see that the proposed deep learning method is able to provide a huge margin in BLER over the traditional methods in Additive Independent Cauchy Noise channel.

Appendix F
Details of Simulation Setup

We used Tensorflow-1.12 to implement deep learning models. All training is done in a desktop-class computer with Intel Core $i7@2.4GHz$ CPU and $16GB$ RAM and no GPU. For BLER results, the transmission of blocks are simulated until 500 block errors are observed, for both DL methods and traditional methods.

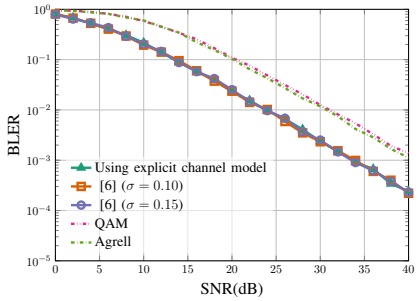
While training in AWGN channel, even though the training set of symbols remained the same, we added different noise samples to each training point at each epoch. Similarly, for RBF channel, we used different values of channel coefficients h and noise samples at each epoch. This technique can reduce model overfitting as well as reduce the chances of getting stuck in saddle points.

While training RBF models, we used equalization to condition the received symbol before feeding to the decoder network. We used a constant pilot symbol of $(1, 1)$ at transmission for equalization during the training phase. As the models trained by the proposed methods do not guarantee constellation of specific energy, we need to appropriately modify the pilot energy during the testing phase. During the testing phase, we maintained the per-component power of pilot to be equal to the average per component power of transmit symbols. This way, we can ensure that both pilot and data symbols experience the same SNR during testing. Pilot boosting can be used to improve that estimation of channel coefficients and hence BLER but is out of the scope of this work.

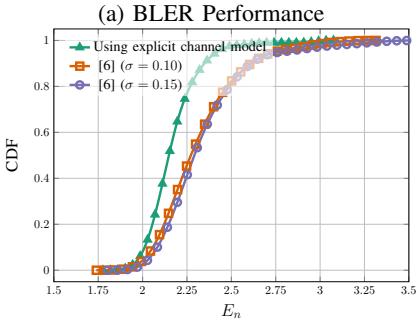
Appendix G
Training Models in Real Channels

Since we assumed the knowledge of the channel and used a model-based simulation system, we were able to train the system with actual gradients. However, in a real system, the channel impairments will be an unknown layer on the network and hence backpropagation of gradients from receiver to transmitter is not possible using traditional optimization techniques used by the deep learning community. Specific to the wireless communication domain, a few practical techniques are developed by the community to mitigate this problem and few of them are discussed below. This includes fine-tuning the receiver decoder with real channel [5], using GANs [32] to approximate the channel behavior [9], [10], approximating the channel gradients [8] by perturbation, perturbing the transmitter outputs [6] etc. We can replace the optimization objectives in these works with the objective function given in (13) and any of the following techniques can be used for model-free training with no further changes.

A comparison of the performance of models trained using the model-aware technique using explicit channel model (trained using Adam, with knowledge of channel function $h(\cdot)$ and $\boldsymbol{\theta}_C$) and model-free technique proposed in [6] with objective function (26) is given in Fig. 15. Here, σ is the standard deviation of the Gaussian perturbation applied at the transmitter output. We can observe that the BLER performance (Fig. 15a) of models trained using [6] is almost the same as the models which require channel knowledge (trained using



(a) BLER Performance



(b) Comparison of packing density

Fig. 15: Comparison of models trained with RBF objective function and different training methods for $M = 256$ and $m = 8$ in RBF channel.

Adam). However, in the case of packing density (Fig. 15b), the models trained with [6] is slightly worse than models trained using Adam. This could be explained as the added perturbation also acts a noise to the model. The only price we pay while using [6] to train is the slow convergence of the models.

REFERENCES

[1] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, May 2019.

[2] E. Balevi and J. G. Andrews, "One-bit OFDM receivers via deep learning," *IEEE Transactions on Communications*, 2019.

[3] Z. Qin, H. Ye, G. Y. Li, and B. F. Juang, "Deep learning in physical layer communications," *IEEE Wireless Communications*, pp. 1–7, 2019.

[4] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[5] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.

[6] F. A. Aoudia and J. Hoydis, "Model-free training of end-to-end communication systems," *arXiv preprint arXiv:1812.05929*, 2018.

[7] D. H. Ballard, "Modular learning in neural networks." in *AAAI*, 1987, pp. 279–284.

[8] V. Raj and S. Kalyani, "Backpropagating through the air: Deep learning at physical layer without channel models," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2278–2281, 2018.

[9] T. J. O'Shea, T. Roy, N. West, and B. C. Hilburn, "Physical layer communications system design over-the-air using adversarial networks," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 529–532.

[10] H. Ye, G. Y. Li, B. F. Juang, and K. Sivanesan, "Channel agnostic end-to-end learning based communication systems with conditional GAN," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec 2018, pp. 1–5.

[11] B. Karanov, M. Chagnon, F. Thouin, T. A. Eriksson, H. Bülow, D. Lavery, P. Bayvel, and L. Schmalen, "End-to-end deep learning of optical fiber communications," *Journal of Lightwave Technology*, vol. 36, no. 20, pp. 4843–4855, 2018.

[12] X. Qian, M. Di Renzo, and A. Eckford, "Molecular communications: Model-based and data-driven receiver design and optimization," *IEEE Access*, vol. 7, pp. 53 555–53 565, 2019.

[13] R. Jiang, X. Wang, S. Cao, J. Zhao, and X. Li, "Deep neural networks for channel estimation in underwater acoustic ofdm systems," *IEEE Access*, vol. 7, pp. 23 579–23 594, 2019.

[14] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2016, pp. 223–228.

[15] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, 1994, pp. 3–10.

[16] P. Marquez Neila, M. Salzmann, and P. Fua, "Imposing hard constraints on deep networks: Promises and limitations," in *CVPR Workshop on Negative Results in Computer Vision*, no. CONF, 2017.

[17] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations.*, 2014.

[19] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[20] H. Kervadec, J. Dolz, M. Tang, E. Granger, Y. Boykov, and I. B. Ayed, "Constrained-cnn losses for weakly supervised segmentation," *Medical image analysis*, vol. 54, pp. 88–99, 2019.

[21] E. Agrell, "Database of sphere packings," *Online: http://codes.se/packings*, 2014, accessed Mar. 1, 2019.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations.*, 2015.

[23] H. Soury and M.-S. Alouini, "On the symbol error rate of m-ary mpsk over generalized fading channels with additive laplacian noise," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 2879–2883.

[24] ——, "Symbol error rate of mpsk over egk channels perturbed by a dominant additive laplacian noise," *IEEE Transactions on Communications*, vol. 63, no. 7, pp. 2511–2523, 2015.

[25] O. S. Badarneh, "Error rate analysis of$m$-ary phase shift keying in$\alpha-\eta-\mu$fading channels subject to additive laplacian noise," *IEEE Communications Letters*, vol. 19, no. 7, pp. 1253–1256, July 2015.

[26] V. Feldman and T. Steinke, "Calibrating noise to variance in adaptive data analysis," *arXiv preprint arXiv:1712.07196*, 2017.

[27] G. A. Tsihrintzis and C. L. Nikias, "Incoherent receivers in alpha-stable impulsive noise," *IEEE Transactions on Signal Processing*, vol. 43, no. 9, pp. 2225–2229, 1995.

[28] K. Gulati, B. L. Evans, J. G. Andrews, and K. R. Tinsley, "Statistics of co-channel interference in a field of poisson and poisson-poisson clustered interferers," *IEEE Transactions on Signal Processing*, vol. 58, no. 12, pp. 6207–6222, 2010.

[29] J. Fahs and I. Abou-Faycal, "A cauchy input achieves the capacity of a cauchy channel under a logarithmic constraint," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 3077–3081.

[30] F. Chyzak and F. Nielsen, "A closed-form formula for the kullback-leibler divergence between cauchy distributions," *arXiv preprint arXiv:1905.10965*, 2019.

[31] J. G. Gonzalez, J. L. Paredes, and G. R. Arce, "Zero-order statistics: A mathematical framework for the processing and characterization of very impulsive signals," *IEEE Transactions on Signal Processing*, vol. 54, no. 10, pp. 3839–3851, 2006.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.