Autoencoder-Based Unequal Error Protection Codes

Autoencoder-Based Unequal Error Protection Codes

Vukan Ninkovic, *Student Member, IEEE,* Dejan Vukobratovic, *Senior Member, IEEE,* Christian Häger, *Member, IEEE,* Henk Wymeersch, *Senior Member, IEEE,* Alexandre Graell i Amat, *Senior Member, IEEE*

***Abstract*—We present a novel autoencoder-based approach for designing codes that provide unequal error protection (UEP) capabilities. The proposed approach, based on a generalization of an autoencoder loss function, provides a versatile framework for the design of message-wise and bit-wise UEP codes. Using an associated weight vector, the generalized loss function can be used to trade off error probabilities corresponding to different importance classes and to explore the region of achievable error probabilities. For message-wise UEP, we compare the proposed autoencoder-based UEP codes with a union of random coset codes. For bit-wise UEP, the proposed codes are compared with UEP rateless spinal codes and the superposition of random Gaussian codes. In all cases, the autoencoder-based codes show superior performance while providing design simplicity and flexibility in trading off error protection among different importance classes.**

***Index Terms*—Autoencoders, deep learning, unequal error protection.**

I. Introduction

Learning transmitters and receivers for a given channel model using deep autoencoders (AE) optimized for a specific loss function has recently been investigated in [1]–[3]. These works consider AE-based encoders and decoders that provide equal error protection across the set of transmitted messages. However, in many communication scenarios, one is interested in the design of unequal error protection (UEP) codes [4].

UEP codes are commonly investigated in two different scenarios: message-wise UEP and bit-wise UEP [5], [6]. In message-wise UEP, the set of source messages is divided into disjoint subsets or importance classes, each of which may be provided with a different level of error protection. In bit-wise UEP, a message is encoded into a sequence of bits and different subblocks of bits represent different importance classes that are protected differently [6]. Practical applications of bit-wise UEP codes, such as improved header protection or scalable multimedia communications, led to bit-wise UEP designs of popular coding schemes such as LDPC [7], fountain [8], or spinal codes [9]. Message-wise UEP codes have been investigated in applications such as alert message transmission and in certain joint source-channel coding scenarios [5].

In this work, we propose a novel AE-based approach to design UEP codes. We introduce a new AE compound loss function that comprises a weighted contribution of each importance class. The proposed approach offers a single and versatile framework to design both message-wise and bit-wise UEP codes that, as we demonstrate, expand the achievable error probability region compared to the UEP codes in the literature. For the case of message-wise UEP, AE-based codes significantly outperform the construction considered in [5] based on the union of coset codes. For bit-wise UEP, the proposed AE-based codes outperform UEP rateless spinal codes [9] and UEP codes based on the superposition of random Gaussian codes [10]. Moreover, the AE-based approach provides a flexible procedure of tuning the weight parameters to trade off error probabilities of different importance classes and explore the region of achievable error probabilities.

V. Ninkovic and D. Vukobratovic are with the Department of Power, Electronics and Communications Engineering, University of Novi Sad, 21000, Novi Sad, Serbia (e-mail: {ninkovic, dejanv}@uns.ac.rs).

C. Häger, H. Wymeersch, and A. Graell i Amat are with the Department of Electrical Engineering, Chalmers University of Technology, SE–41296 Gothenburg, Sweden (e-mail: {christian.haeger, henkw, alexandre.graell}@chalmers.se).

This work has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement number 856967.

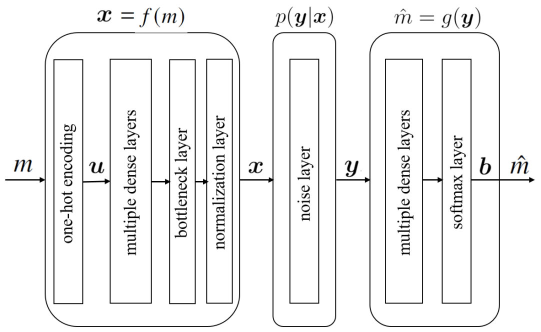

Fig. 1. Communication system represented as a deep autoencoder [1].

II. Background

A. System Model

We consider the problem of communicating a message m from a set of messages $\mathcal{M} = \{1, 2, \dots, M\}$ over a noisy channel. Each message is represented as a sequence of bits $\boldsymbol{s} = (s_1, s_2, \dots, s_k)$, where $k = \log_2(M)$ is the message length. We define the encoder mapping $f : \mathcal{M} \to \mathbb{R}^n$ that encodes the message m into a codeword $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ of length n. The transmitted codewords $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_2^2 = n\}$, i.e., the codewords $\boldsymbol{x}$ obey a total energy constraint. The code rate is $R = k/n$ (in bits per channel use). The channel $\mathcal{W}$ transforms the input codeword $\boldsymbol{x} \in \mathbb{R}^n$ into the output sequence $\boldsymbol{y} \in \mathbb{R}^n$ following the probabilistic channel law $p(\boldsymbol{y}|\boldsymbol{x})$. Finally, the decoder mapping $g : \mathbb{R}^n \to \mathcal{M}$ produces an estimate $\hat{m}$ of the transmitted message m. Under the above setup, the goal is to design a pair (f, g) for the channel $\mathcal{W}$ to minimize the average message error probability

$$P_\text{e} = \frac{1}{M} \sum_{m \in \mathcal{M}} \mathbb{P}\{\hat{m} \neq m | m\}. \tag{1}$$

B. Autoencoder-Based Code Design

From a deep learning perspective, the above communication system can be represented as an AE [1]. An AE consists of a set of encoder layers representing the encoder mapping $\boldsymbol{x} = f(m)$, the noise layer modeling the channel $\mathcal{W}$ that transforms $\boldsymbol{x}$ into $\boldsymbol{y}$, and a set of decoder layers representing the decoder mapping $\hat{m} = g(\boldsymbol{y})$, as shown in Fig. 1.

At the input of the encoder layers, the message m is encoded as a one-hot vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_M) \in \{0,1\}^M$, i.e., it is represented as an M-dimensional vector with the m-th element equal to one and the others equal to zero. The set of encoder layers represents a feed-forward neural network with H hidden layers, followed by a bottleneck layer of width n. The goal of the encoder neural network is to find the most suitable representation of the information so that it is robust to the channel perturbations. At the output of the bottleneck layer, normalization ensures that $\boldsymbol{x} \in \mathcal{X}$.

Next, the codewords $\boldsymbol{x}$ are passed through a noise layer that represents the channel $\mathcal{W}$. In this paper, we consider an additive white Gaussian noise (AWGN) channel, thus at the output of the noise layer we have $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{z}$, where $\boldsymbol{z}$ contains n independent and identically distributed (i.i.d.) samples of a Gaussian random variable $\mathcal{N}(0, \sigma^2)$ of variance σ^2.

The output of the noise layer $\boldsymbol{y}$ is fed into the decoder layers representing the receiver. The receiver is implemented in the same way as the transmitter via a feed-forward neural network, except that the last layer has a softmax activation function with output $\boldsymbol{b} = (b_1, b_2, \ldots, b_M) \in (0,1)^M$, where the ℓ_1 norm $\|\boldsymbol{b}\|_1 = 1$. The decoded message is $\hat{m} = \arg\max_i \{b_i\}$. Except for the last layer at the transmitter and the receiver that have a linear and a softmax activation function, respectively, all others layers are activated by a rectified linear unit (ReLU).

The AE is trained in an end-to-end manner by using stochastic gradient descent (SGD) with the Adam optimizer [11] on the set of all possible messages $m \in \mathcal{M}$. The minimization of the cross-entropy loss between $\boldsymbol{u}$ and $\boldsymbol{b}$ is used as a surrogate for minimizing the error probability P_{e}, which cannot be used directly as it is not differentiable. The AE is trained using batches of training data by minimizing the cross-entropy loss

$$\ell(\boldsymbol{u}, \boldsymbol{b}) = -\sum_{i=1}^{M} u_i \log b_i, \tag{2}$$

averaged across a batch of training samples.

C. Unequal Error Protection Code Design

We consider both message-wise UEP and bit-wise UEP, as detailed next.

1) Message-wise UEP: We assume that the message set $\mathcal{M}$ containing M messages is partitioned into $C \leq M$ disjoint subsets, referred to as message classes, having different error protection requirements. Message class $\mathcal{M}_i$ contains $|\mathcal{M}_i| = M_i$ messages, with $M = \sum_{i=1}^{C} M_i$. For a given encoder-decoder pair (f, g), we define the per-class probability of error

$$P_{\mathrm{e}}^{(i)} = \frac{1}{M_i} \sum_{m \in \mathcal{M}_i} \mathbb{P}\{\hat{m} \neq m | m\}. \tag{3}$$

We collect the per-class error probabilities of a message-wise UEP code in a vector $\boldsymbol{P}_{\mathrm{e}} = (P_{\mathrm{e}}^{(1)}, P_{\mathrm{e}}^{(2)}, \ldots, P_{\mathrm{e}}^{(C)})$. We use the term message-wise UEP code to refer to the triple $(\{\mathcal{M}_i\}_{i=1}^{C}, f, g)$ [5]. We denote as $\mathcal{P}_{\mathcal{W}}(\{\mathcal{M}_i\}_{i=1}^{C}, n) \subset [0,1]^C$ the region of achievable $\boldsymbol{P}_{\mathrm{e}}$-values for message-wise UEP codes of length n and the message classes $\{\mathcal{M}_i\}_{i=1}^{C}$ over the channel $\mathcal{W}$.

2) Bit-wise UEP: In bit-wise UEP, we consider an equivalent representation $\mathcal{S}$ of the message set $\mathcal{M}$ that consists of the binary representation ($\boldsymbol{s}$, see Section II-A) of the messages $m \in \mathcal{M}$. Further, we assume that $\boldsymbol{s}$ consists of C submessages representing disjoint subsequences of bits, i.e., $\boldsymbol{s} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_C)$, where the length of submessage $\boldsymbol{s}_i$ is k_i bits and $k = \sum_{i=1}^{C} k_i$. We denote by $\mathcal{S}$ (respectively, $\mathcal{S}_i$) the set of all possible binary messages $\boldsymbol{s}$ (submessages $\boldsymbol{s}_i$), with $|\mathcal{S}| = 2^k$ ($|\mathcal{S}_i| = 2^{k_i}$). In the case of bit-wise UEP, the different submessages represent the different message classes and are assigned different error protection requirements.

We are interested in the probability of error associated with a particular message class. For a submessage $\boldsymbol{s}_i \in \mathcal{S}_i$, we denote by $\mathcal{M}_{\boldsymbol{s}_i}$ the set of all messages $m \in \mathcal{M}$ whose i-th submessage in the corresponding representation $\boldsymbol{s}$ equals $\boldsymbol{s}_i$. For a given encoder-decoder pair (f, g), we define the per-class probability of error

$$P_{\mathrm{e}}^{(i)} = \frac{1}{|\mathcal{S}_i|} \sum_{\boldsymbol{s}_i \in \mathcal{S}_i} \mathbb{P}\{\hat{m} \notin \mathcal{M}_{\boldsymbol{s}_i} | m \in \mathcal{M}_{\boldsymbol{s}_i}\}, \tag{4}$$

and define $\boldsymbol{P}_{\mathrm{e}} = (P_{\mathrm{e}}^{(1)}, P_{\mathrm{e}}^{(2)}, \ldots, P_{\mathrm{e}}^{(C)})$. We use the term bit-wise UEP code to refer to the triple $(\{\mathcal{S}_i\}_{i=1}^{C}, f, g)$.

Example 1: For clarity, we consider an example for $|\mathcal{M}| = 16$ messages corresponding to the set $\mathcal{S}$ of all binary sequences $\boldsymbol{s}$ of length $k = 4$. Let $\boldsymbol{s} = (\boldsymbol{s}_1, \boldsymbol{s}_2)$ be divided into $C = 2$ submessages, where both $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are of length $k_1 = k_2 = 2$ bits. Then, if $\boldsymbol{s}_1 = (1,0)$ is transmitted, it is considered correctly decoded as long as the decoded sequence is consistent with $\boldsymbol{s}_1$, i.e., it belongs to $\mathcal{M}_{\boldsymbol{s}_1 = (1,0)} = \{(1,0,0,0), (1,0,0,1), (1,0,1,0), (1,0,1,1)\}$.

We now discuss an extension of the bit-wise UEP scenario called progressive bit-wise UEP. For some practical applications that call for bit-wise UEP, the above definition of per-class error probabilities does not capture the code design requirements. For example, of great interest is the case where the messages are encoded in such a way that the importance of submessages progressively decreases from $\mathcal{S}_1$ to $\mathcal{S}_C$ and there exists an interdependence between importance classes. More precisely, in such applications, the i-th message block $\boldsymbol{s}_i$ is considered as correctly received if and only if this block as well as all blocks $\boldsymbol{s}_j$ for $j < i$ are decoded correctly [12]. To accommodate for this scenario, we redefine the per-class error probability as

$$P_{\mathrm{e}}^{(i)} = \frac{1}{|(\mathcal{S}_1, \ldots, \mathcal{S}_i)|} \cdot \sum_{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_i \in (\mathcal{S}_1, \ldots, \mathcal{S}_i)} \mathbb{P}\{\hat{m} \notin \mathcal{M}_{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_i} | m \in \mathcal{M}_{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_i}\}, \tag{5}$$

where $\mathcal{M}_{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_i}$ is the set of all messages $m \in \mathcal{M}$ whose binary representation $\boldsymbol{s}$ is consistent with the first i submessages $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_i$. We refer to this case as progressive bit-wise UEP.

III. Autoencoder-Based UEP Codes

A. UEP Autoencoders

In this section, we present a flexible and efficient method to design encoders and decoders for both message-wise and bit-wise UEP codes by training deep AEs. The key idea of the proposed AE-based design is to define an appropriate *compound loss function* that generalizes the cross-entropy loss function defined in (2) to the UEP case.

1) Message-wise UEP: Let $\ell_{\mathcal{M}_j}(\boldsymbol{u},\boldsymbol{b})$ be the loss function associated to the j-th message class defined as

$$\ell_{\mathcal{M}_j}(\boldsymbol{u},\boldsymbol{b}) = -\sum_{i\in\mathcal{M}_j} u_i \log b_i. \tag{6}$$

We redefine the loss function for the case of message-wise UEP as the weighted sum of the loss functions $\ell_{\mathcal{M}_j}(\boldsymbol{u},\boldsymbol{b})$ as

$$\ell(\boldsymbol{u},\boldsymbol{b}) = \sum_{j=1}^{C} \lambda_j \ell_{\mathcal{M}_j}(\boldsymbol{u},\boldsymbol{b}), \tag{7}$$

where $\boldsymbol{\lambda} = (\lambda_1,\lambda_2,\dots,\lambda_C)$ is a weight vector associated to the message classes, $\sum_{j=1}^{C}\lambda_j = 1$, and $\lambda_j \geq 0$. The rest of the training procedure follows the same steps as for the standard (equal error protection) AE-based codes.

2) Bit-wise UEP: We use the same approach of modifying the loss function using a weighting vector. In this case, however, some updates in the message representation are needed beforehand. We extend the definition of one-hot vector $\boldsymbol{u}$ so that it indicates a subset of messages in $\mathcal{M}$ whose binary symbol representation $\boldsymbol{s}$ is consistent with a given submessage $\boldsymbol{s}_j \in \mathcal{S}_j$. More precisely, for every submessage $\boldsymbol{s}_j \in \mathcal{S}_j$, we define the corresponding vector $\boldsymbol{u}_{\boldsymbol{s}_j} = (u_1,u_2,\dots,u_M)$, with the m-th position set to one if the message m is such that the j-th submessage of its binary sequence representation is $\boldsymbol{s}_j$. Note that $\boldsymbol{u}_{\boldsymbol{s}_j}$ is now a binary vector with 2^{k-k_j} ones. Let $\ell(\boldsymbol{u}_{\boldsymbol{s}_j},\boldsymbol{b})$ be the loss function associated to the j-th submessage,

$$\ell(\boldsymbol{u}_{\boldsymbol{s}_j},\boldsymbol{b}) = -\sum_{i=1}^{M} u_i \log b_i. \tag{8}$$

Given the binary sequence representation $\boldsymbol{s} = (\boldsymbol{s}_1,\boldsymbol{s}_2,\dots,\boldsymbol{s}_C)$ of a message $m \in \mathcal{M}$, we define a set of C vectors $\mathcal{U} = \{\boldsymbol{u}_{\boldsymbol{s}_1},\boldsymbol{u}_{\boldsymbol{s}_2},\dots,\boldsymbol{u}_{\boldsymbol{s}_C}\}$. With the above definition, the compound loss function for bit-wise UEP is defined as

$$\ell(\mathcal{U},\boldsymbol{b}) = \sum_{j=1}^{C} \lambda_j \ell(\boldsymbol{u}_{\boldsymbol{s}_j},\boldsymbol{b}), \tag{9}$$

where $\boldsymbol{\lambda} = (\lambda_1,\lambda_2,\dots,\lambda_C)$ is a weight vector associated to message classes, $\lambda_j > 0$, and $\sum_{j=1}^{C}\lambda_j = 1$. The rest of the training procedure follows the same steps as for the standard (equal error protection) AE-based codes.

The case of progressive bit-wise UEP can be treated in a similar way. For every submessage $\boldsymbol{s}_i \in \mathcal{S}_i$, we define the binary vector $\boldsymbol{u}_{\boldsymbol{s}_1,\dots,\boldsymbol{s}_i} = (u_1,u_2,\dots,u_M)$, with a 1 in the m-th entry if the message $m \in \mathcal{M}_{\boldsymbol{s}_1,\dots,\boldsymbol{s}_i}$. Given a binary sequence representation $\boldsymbol{s} = (\boldsymbol{s}_1,\boldsymbol{s}_2,\dots,\boldsymbol{s}_C)$ of a message $m \in \mathcal{M}$, we define a set of C vectors $\mathcal{U} = \{\boldsymbol{u}_{\boldsymbol{s}_1},\boldsymbol{u}_{\boldsymbol{s}_1,\boldsymbol{s}_2},\dots,\boldsymbol{u}_{\boldsymbol{s}_1,\boldsymbol{s}_2,\dots,\boldsymbol{s}_C}\}$. Finally, we can reuse (8) and (9) by inserting the appropriate $\mathcal{U}$ and $\boldsymbol{u}_{\boldsymbol{s}_1,\dots,\boldsymbol{s}_j}$.

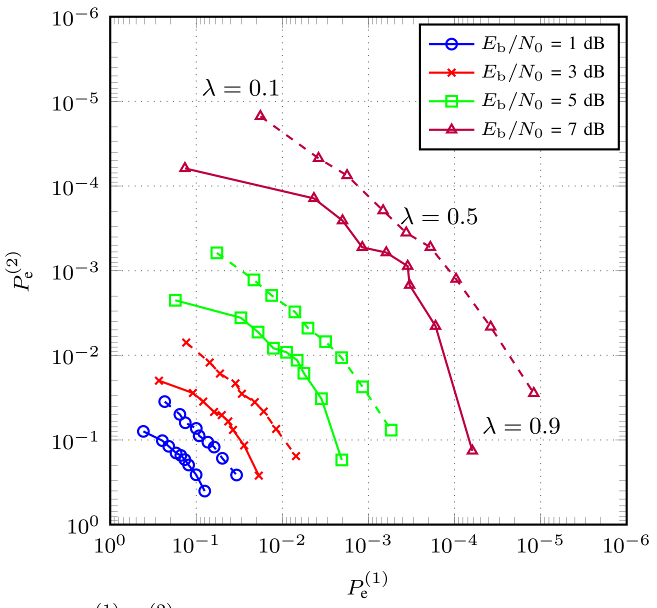

Fig. 2. $(P_e^{(1)}, P_e^{(2)})$ performance of AE-based message-wise (solid curves) and bit-wise UEP codes (dashed curves) with $C = 2$, $|\mathcal{M}_1| = 8$ and $|\mathcal{M}_2| = 8$ for message-wise and $k_1 = 2$ and $k_2 = 2$ for bit-wise codes, $n = 7$, $E_b/N_0 = \{1, 3, 5, 7\}$ dB and $\lambda = \{0.1, 0.2, \dots, 0.9\}$.

B. Flexible AE-Based UEP Code Design

We exemplify the AE-based UEP code design for $M = 16$ messages transmitted using $n = 7$ channel uses over the AWGN channel for $C = 2$ error-protection classes. For both message-wise and bit-wise UEP, we design pairs (f, g) that explore the trade-off between per-class error probabilities $(P_e^{(1)}, P_e^{(2)})$. In the proposed AE-based design, varying the weight vector $\boldsymbol{\lambda}$ is a flexible mechanism to trade off the values within $\boldsymbol{P}_e$ and thus explore the region of achievable $\boldsymbol{P}_e$-values (for $C = 2$, $\boldsymbol{\lambda} = (\lambda_1,\lambda_2) = (\lambda, 1-\lambda)$, where $\lambda \in [0, 1]$).

1) Message-wise UEP: For message-wise UEP, we partition the set of $M = 16$ messages into two classes, each containing half of the messages, i.e., $|\mathcal{M}_1| = 8$ and $|\mathcal{M}_2| = 8$. In Fig. 2 (solid lines), we plot the error-probability profile $\boldsymbol{P}_e = (P_e^{(1)}, P_e^{(2)})$ of the trained AE-based encoder-decoder pair (f, g) for $E_b/N_0 = 1, 3, 5,$ and 7 dB, where E_b denotes the energy per bit, N_0 is the noise power spectral density, and $\sigma^2 = (2RE_b/N_0)^{-1}$. Each curve is obtained by evaluating $(P_e^{(1)}, P_e^{(2)})$ for the weight parameter $\lambda = \{0.1, 0.2, \dots, 0.9\}$. As in [1], we consider both encoder and decoder layers consisting of a single hidden layer with $M = 16$ neurons, while the bottleneck layer has $n = 7$ neurons. The value of E_b/N_0 applied during the training can be understood as a hyperparameter. We set $E_b/N_0 = 3$ dB, as for $E_b/N_0 > 3$ dB, $P_e^{(1)}$ significantly deteriorates, while the same holds for $P_e^{(2)}$ when $E_b/N_0 < 3$. The AE is optimized using SGD with Adam optimizer, applying the learning rate $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

First, note that for $\lambda = 0.5$, the values of $(P_e^{(1)}, P_e^{(2)})$ coincide, as our AE code boils down to the equal error protection AE code presented in [1]. Secondly, note that the curves $(P_e^{(1)}, P_e^{(2)})$ are symmetric with respect to $\lambda = 0.5$. Finally, as we increase λ from 0.5 toward 1, for any fixed

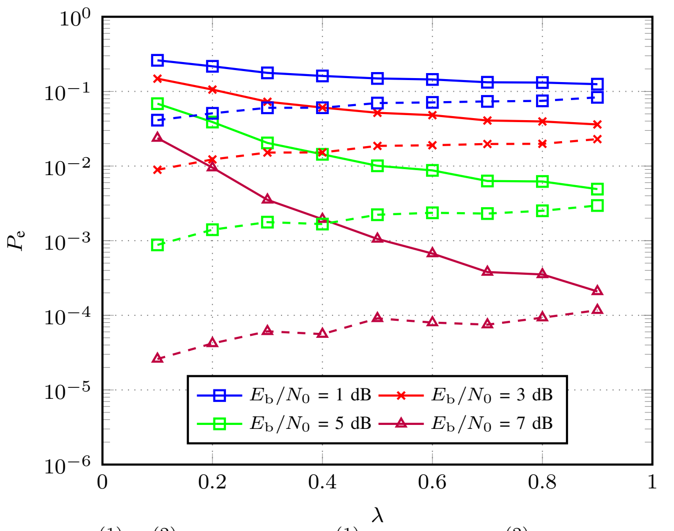

Fig. 3. $(P_e^{(1)}, P_e^{(2)})$ performance ($P_e^{(1)}$ solid curves, $P_e^{(2)}$ dashed curves) of AE-based progressive bit-wise UEP codes ($C = 2$, $k_1 = 2$, $k_2 = 2$, $n = 7$) for different values of λ and $E_b/N_0 = \{1, 3, 5, 7\}$ dB.

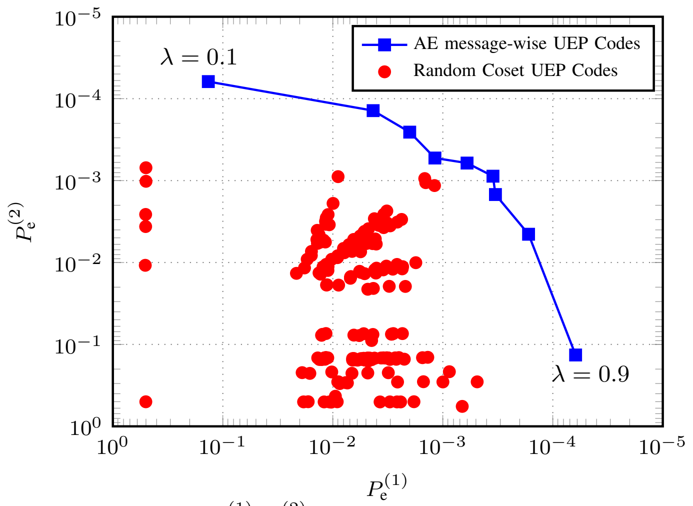

Fig. 4. Comparison of $(P_e^{(1)}, P_e^{(2)})$ performance of AE-based message-wise UEP codes vs random coset UEP codes ($C = 2$, $|\mathcal{M}_1| = 8, |\mathcal{M}_2| = 8$, $n = 7$) for $E_b/N_0 = 7$ dB.

E_b/N_0, the AE-based UEP code error-probability profile $\boldsymbol{P}_e = (P_e^{(1)}, P_e^{(2)})$ sweeps through a sequence of pairs where $P_e^{(1)}$ gradually improves, while $P_e^{(2)}$ gradually deteriorates. Thus, the parameter λ offers a flexible “fine-tuning knob” that allows for the selection of the desired trade-off within the region $\mathcal{P}_{\mathcal{W}}(\mathcal{M}, n)$.

2) Bit-wise UEP: In this scenario, $M = 16$ messages are represented as $k = 4$-bit sequences $\boldsymbol{s} \in (\mathcal{S}_1, \mathcal{S}_2)$, where submessage $\boldsymbol{s}_1 \in \mathcal{S}_1$ contains the first two bits ($k_1 = 2$), while submessage $\boldsymbol{s}_2 \in \mathcal{S}_2$ contains the remaining two bits ($k_2 = 2$). Fig. 2 (dashed lines) plots the error-probability profile $\boldsymbol{P}_e = (P_e^{(1)}, P_e^{(2)})$ of a trained AE-based encoder-decoder pair (f, g) for the same values of E_b/N_0 and for the same sequence of λ values as in the message-wise UEP example. The encoder and decoder pairs (f, g) are obtained using the same architecture and training methodology as for the message-wise UEP. Notably, the error-probability profile $\boldsymbol{P}_e$ shows a similar behavior as for the case of message-wise UEP, despite fundamental difference between the two scenarios.

For the progressive bit-wise UEP, where the success in decoding the second message subblock is conditioned on successful decoding of the first subblock, the error-probability profile $\boldsymbol{P}_e$ changes as illustrated in Fig. 3. Informally, as $\lambda \to 0$, the codewords corresponding to $\mathcal{M}_{\boldsymbol{s}_1}$ (for any $\boldsymbol{s}_1$) converge to each other, while as $\lambda \to 1$ they diverge from each other, approaching equal error protection. Note that the resulting code design is related to the problems of superposition coding for degraded broadcast channels [13] and designing UEP modulation constellations [14].

IV. Performance of AE-based UEP Codes

In this section, we compare the AE-based UEP codes against selected classes of UEP codes.

1) Message-wise UEP: In [5], the authors consider a message-wise UEP construction based on a union of coset codes. For each message class $\mathcal{M}_i$, we generate a random binary generator matrix $\boldsymbol{G}_i$ of dimension $k_i \times n$ and a random binary shift vector $\boldsymbol{v}_i$ of length n. For each message $m \in \mathcal{M}_i$, the corresponding codeword $\boldsymbol{x}$ is obtained as the binary phase shift keying (BPSK) modulated version of $\boldsymbol{s}\boldsymbol{G}_i + \boldsymbol{v}_i$, where $\boldsymbol{s}$ is the binary sequence representation of m. For $C = 2$ and $n = 7$, in order to generate $\mathcal{M}_1 = \mathcal{M}_2 = 8$ codewords, we set $k_1 = 2$ and $k_2 = 2$. We compare the AE-based design with a randomly generated set of 200 random coset-based message-wise UEP codes. Note that, unlike the AE-based design where we use λ to control the trade-off between per-class error probabilities in $\boldsymbol{P}_e$, for random coset-based codes such a control of $\boldsymbol{P}_e$ is not trivial, thus we compare against a sample of 200 randomly generated codes. Although the coset-based design is asymptotically good for the BSC and the BEC [5], for the AWGN and short code lengths, the performance of the best selected candidates does not match that of AE-based designed codes (note that the encoding function of coset-based codes is restricted to $f : \mathcal{M} \to \{+1, -1\}^n \subset \mathcal{X}$). Fig. 4 demonstrates that, for $\lambda = \{0.1, 0.2, \ldots, 0.8, 0.9\}$, the error-probability pairs $(P_e^{(1)}, P_e^{(2)})$ for AE-based codes consistently outperform randomly-sampled set containing 200 coset-based codes (at $E_b/N_0 = 7$ dB). In other words, AE-based codes perform consistently closer to the boundary of the region $\mathcal{P}_{\mathcal{W}}(\{\mathcal{M}_i\}_{i=1}^{C}, n)$.

2) Bit-wise UEP: In Fig. 5, we compare the AE-based bit-wise UEP codes with the bit-wise UEP spinal codes [9]. The code parameters are $C = 2$, $k_1 = 4$, $k_2 = 10$, and $n = 32$. For the AE-based code, we apply the AE architecture with a single hidden layer with 500 neurons for both the transmitter and the receiver and $n = 32$ neurons for the bottleneck layer. The size of the hidden layer is selected experimentally as a compromise between error-correction performance and AE computation complexity. Furthermore, we consider $\lambda = \{0.7, 0.8, 0.9\}$. The training procedure is the same as described in Section III-B except that the training is done at $E_b/N_0 = 1$ dB. The AE-based codes perform comparably to spinal codes in terms of $P_e^{(1)}$, while significantly outperforming them for $P_e^{(2)}$.

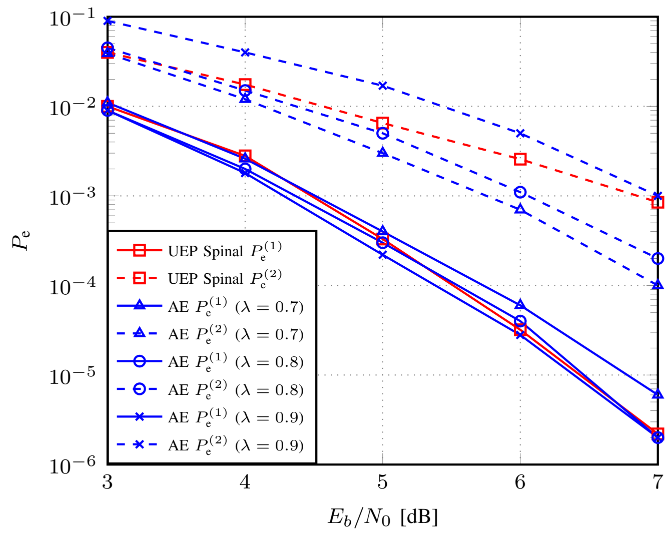

Fig. 5. Comparison of $(P_e^{(1)}, P_e^{(2)})$ vs E_b/N_0 performance of AE-based and spinal bit-wise UEP codes ($C = 2$, $k_1 = 4$, $k_2 = 10$, $n = 32$).

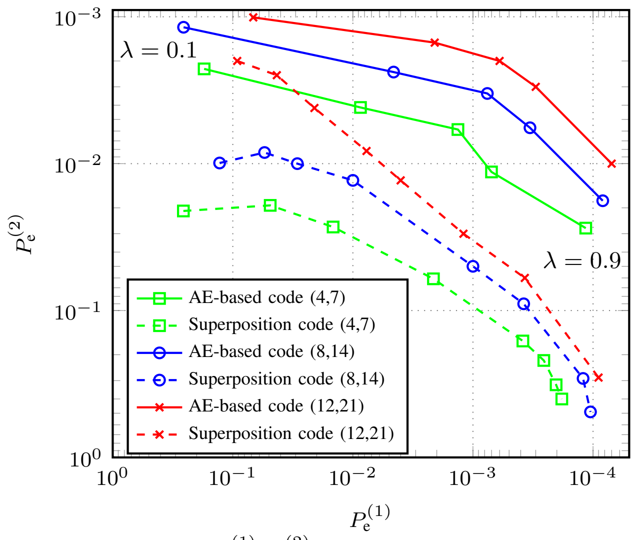

Fig. 6. Comparison of $(P_e^{(1)}, P_e^{(2)})$ performance of AE-based and superposition of random Gaussian codes for (k, n)-pairs: $(4, 7)$, $(8, 14)$ and $(12, 21)$, where $k_1 = \frac{1}{4}k$ and $k_2 = \frac{3}{4}k$ at $E_b/N_0 = 5$ dB.

Next, we compare the AE-based design with the bit-wise UEP codes based on the superposition of random Gaussian codes [10].[1] For $C = 2$, each codeword from a set of 2^{k_1} random Gaussian codewords whose symbols are drawn from $\mathcal{N}(0, \sigma_1^2)$ is superimposed by a set of 2^{k_2} random Gaussian codewords whose symbols are sampled from $\mathcal{N}(0, \sigma_2^2)$. The resulting set of $M = 2^k$ codewords is normalized to obey the total energy constraint. We consider three equal-rate (k, n)-pairs, $(4, 7)$, $(8, 14)$, and $(12, 21)$, where $k_1 = \frac{1}{4}k$ and $k_2 = \frac{3}{4}k$. We use a single parameter μ to design the superposition of random Gaussian codes by varying $(\sigma_1^2, \sigma_2^2) = (\mu, 1 - \mu)$, providing a similar control of the trade-off between error probabilities $(P_e^{(1)}, P_e^{(2)})$ as λ provides for the AE-based codes. Fig. 6 demonstrates superior performance of the AE-based codes for $\lambda = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ at $E_b/N_0 = 5$ dB compared to the best performing UEP codes based on the superposition of random Gaussian codes for $\mu = \{0.3, 0.4, 0.5, 0.6, 0.7\}$ (from a set of 200 randomly generated codes). Finally, although the results presented are restricted to $R = 4/7$ and $7/16$, our experiments show that the proposed AE-based UEP codes perform well across a wide range of code rates.

V. Conclusion

We presented a novel learning-based approach to the design of UEP codes using deep AEs. The design is based on a generalized AE loss function that accommodates both message-wise and bit-wise UEP code design. The proposed AE-based codes show superior performance to known UEP approaches, such as random coset codes or superposition of random Gaussian codes. They also outperform state-of-the-art UEP rateless spinal codes. Besides excellent performance, AE-based UEP codes provide a built-in flexible mechanism for weighting loss function components that results in a graceful trade-off between per-class error probabilities. The limitation of the proposed scheme—as that of other AE-based codes with one-hot encoding—is that the codeword dimension and length are severely limited. A binary representation of the messages may alleviate this problem at the expense of some performance penalty, but more efficient approaches are in general needed.

[1]We consider a special case of [10], where decoding of both message classes is attempted after all n codeword symbols are received.

References

[1] 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, Dec. 2017.
[2] T. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based MIMO communications," Jul. 2017., arXiv:1707.07980v1 [cs.IT] . [Online]. Available: https://arxiv.org/abs/1707.07980
[3] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.,* vol. 12, no. 1, pp. 132-143, Feb. 2018.
[4] B. Masnick and J. Wolf, "On linear unequal error protection codes," *IEEE Trans. Inf. Theory,* vol. 13, no. 4, pp. 600-607, Oct. 1967.
[5] Y. Y. Shkel, V. Y. Tan, and S. C. Draper, "Unequal message protection: Asymptotic and non-asymptotic tradeoffs," *IEEE Trans. Inf. Theory,* vol. 61, no. 10, pp. 5396-5416, Oct. 2015.
[6] S. Borade, B. Nakiboglu, and L. Zheng, "Unequal error protection: An information-theoretic perspective," *IEEE Trans. Inf. Theory,* vol. 55, no. 12, pp. 5511-5539, Dec. 2009.
[7] H. Pishro-Nik, N. Rahnavard, and F. Fekri, "Nonuniform error correction using low-density parity-check codes," *IEEE Trans. Inf. Theory,* vol. 51, no. 7, pp. 2702-2714, July 2005.
[8] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, and R. J. Piechocki, "Expanding window fountain codes for unequal error protection," *IEEE Trans. Commun.,* vol. 57, no. 9, pp. 2510-2516, Sep. 2009.
[9] X. Yu, Y. Li, W. Yang, and Y. Sun, "Design and analysis of unequal error protection rateless spinal codes," *IEEE Trans. Commun.,* vol. 64, no. 11, pp. 4461-4473, Nov. 2016.
[10] M. Karimzadeh and M. Vu, "Short Blocklength Priority-Based Coding for Unequal Error Protection in the AWGN Channel," in *Proc. 2019 IEEE Global Commun. Conf. (GLOBECOM),* Waikoloa, HI, USA, Dec. 9-13, 2019, pp. 1-6.
[11] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. on Learn. Representation,* San Diego, CA, USA, May 7-9, pp. 1-41, 2015.
[12] V. Chande and N. Farvardin, "Progressive transmission of images over memoryless noisy channels," *IEEE J. Sel. Areas Commun. ,* vol. 18, no. 6, pp. 850–860, June 2000.
[13] T. M. Cover, "Comments on broadcast channels," *IEEE Trans. Inf. Theory,* vol. 44, no. 6, pp. 2524-2530, Oct. 1998.
[14] L. F. Wei, "Coded modulation with unequal error protection," *IEEE Trans. Commun.,* vol. 41, no. 10, pp. 1439-1449, Oct. 1993.