IEEE Copyright Notice

Accepted to be published in: IEEE Transaction on Cognitive Communications and Networking

U-shaped Error Correction Code Transformers

Dang-Trac Nguyen, Sunghwan Kim

***Abstract*—In this work, we introduce two variants of the U-shaped error correction code transformer (U-ECCT) in combination with weight-sharing to improve the decoding performance of the error correction code transformer (ECCT) for moderate-length linear codes. The proposed models are inspired by the well-known U-Net architecture to leverage residual information for faster error estimation based on the syndrome-based reliability decoding principle. As an effort to further improve the general decoding performance of the U-ECCT, we propose the variational U-ECCT (VU-ECCT), in which the process of learning the shortcut connections is treated as a generative problem, forming a variational autoencoder (VAE) that exists intertwined with the existing U-ECCT model. This design allows the extraction of mutual information between the different levels of the U-shaped architecture, thus enhancing the performance of large syndrome sequences for low-rate codes. Additionally, to further reduce the model size, a new weight-sharing strategy, called mirror-sharing, is proposed to compress the model size as well as compliment the mechanism of the proposed U-shaped architecture. In experiments, it has been demonstrated that our proposed models achieve significantly better performance than baseline conventional algorithms and other learning-based models.**

***Index Terms*—Transformer, error correction code transformer (ECCT), mirror-sharing, variational autoencoder (VAE), U-Net**

I. Introduction

RELIABILITY of data transmission is one of the most important aspects of wireless communications. Among the leading technologies of modern communication systems, error correction codes (ECCs) have long been recognized as one of the most important factors contributing to the enhancement of reliable transmission due to their noise resistance and error correction capability. To achieve this, ECCs utilize likelihood decoding algorithms such as belief propagation (BP) decoding, min-sum (MS) decoding, and maximum likelihood (ML) decoding for low-error probability estimation of original messages. While these decoding algorithms have been proven to be powerful and optimal in certain code settings, further optimizing them has been considered highly challenging, posing difficulties for the advancement of ECCs. Other efforts to improve conventional ECCs have focused on situational code designs, which are mostly effective for their specific applications.

With the rise of deep learning applications across major aspects of society, learning-based approaches for ECC adaptation have gained favor due to achieving better performance compared to traditional methods. Model-based designs adopted the BP algorithm [1]–[6] or its variants, and the min-sum algorithm [7] for neural decoding designs. Although these works were able to achieve significant improvements over BP, their models have still been restricted by the complex decoding rules involving Tanner-graph. On the other hand, model-free approaches [8]–[14], [48], which were pioneered by the authors of [8], were not subject to the complexity of the Tanner-graph but instead hindered by the curse of dimensionality, especially with significantly large code length. Recently, the authors of [15] proposed the transformer-based architecture for effective model-free decoding of ECCs. In [15], different components from the originally proposed transformer in [16] were modified to achieve a code-aware transformer that operates on the basis of the syndrome-based reliability decoding principle [17]. The transformer-based model in [15] was able to achieve a significant reduction in error probability compared to the state-of-the-art learning-based approaches. However, similar to previous model-free research, the error correction code transformer (ECCT) in [15] suffered from scalability problems, where the model performed sub-optimally on moderately large code lengths with a shallow architecture (few stacked layers) and required a deeper architecture (moderate number of stacked layers) to maintain the same level of performance gain. This resulted in a significant increase in the model size for ECCT, which greatly affected the time complexity of the decoding process. Therefore, a structural change that reduces the dependency of performance on model size is needed to effectively address the problem.

The research on neural network architectural design has been highly successful, yielding renowned models such as ResNet [19], [20], DenseNet [21], [22], and MobileNet [23]. Among the popular architectural designs, U-Net [24] is known as the state-of-the-art image segmentation convolutional model. While it is well-known for its application in the field of computer vision, many researchers have successfully adapted this design to time-series data and other fields of research, as well as combining it with models other than convolutional neural networks.

Driven by the need for an architectural change to resolve the limitation of the ECCT in [15] when decoding larger code-length, in this work, we propose U-shaped ECCT (U-ECCT) models. The contributions of this work are as follows:

- A design that combines a U-shaped architecture with ECCT is proposed to provide flexibility in scaling the size of the transformer-based decoder architecture and to improve performance for decoding moderate-length linear block codes. This design offers an efficient learning-based decoder, contributing to the advancement of intelligent communication systems. The proposed U-shaped architecture consists of two major network flows: the main flow and the shortcut connections. While the decoding process of the main network flow is similar to the baseline ECCT, the shortcut connections interact with the main flow through a multiplicative operator. With this mechanism, the proposed model learns to generate and combine two different segments of the embedding

dimension, retaining common information between the two flows, which significantly enhances the decoding process, particularly targeting large code-length cases. Furthermore, the U-ECCT is designed with multi-level scaling with four different levels of hidden dimension, providing flexibility and reducing model size compared to the baseline ECCT [15].

- In order to improve the general performance of the U-ECCT at all code rates, the U-shaped architecture is combined with the variational autoencoder (VAE) model to form a variational U-ECCT (VU-ECCT). In our improved model, the interaction between the main network flow and the shortcut connections is modified to resemble the VAE. This modified U-ECCT model provides a way to estimate the common distribution of the shortcut connections. Through this effect, the large syndrome length is fully utilized, enabling significant improvements in low-rate codes.
- Finally, in an effort to enable an efficient U-ECCT design, a new weight-sharing strategy is proposed, called mirror-sharing. Our weight-sharing strategy is applied so that the second half of the U-shaped model reuses the parameters of its first half in a reflective manner. This method effectively compresses our proposed U-ECCT models, enabling a deep, light-weight model that performs better than the baseline ECCT.
- In experimentation, all U-ECCT models achieve significantly better performance than the baseline ECCT for moderate code-length cases (from 256 to 1024), and still provide noticeable improvement for short code-length cases (less than 256). In terms of decoding performance, the basic U-ECCT variant with optimal size provides improvements that vary between 0.3dB and 0.5dB for moderate code-length cases, depending on the code structure and code rates. The VU-ECCT model supports moderate improvement, especially for low code rate cases, achieving improvements between 0.5dB and 0.65dB. Regarding efficiency, through mirror-sharing strategy, our models provide effective compression, achieving up to 73% reduction in model size compared to the baseline ECCT [15].

The rest of this paper is organized as follows: Section II introduces the related knowledge adopted for this work. Section III consists of detailed explanations for the system model and our proposed methods. The experimental results and discussions are described in Section IV. Finally, conclusion to this work is provided in Section V.

II. Background and Related Works

In this section, necessary background knowledge is provided for transformers [16], [25]–[34] and key features of the ECCT [15], U-Net architecture [24], [35]–[38], and variational autoencoder (VAE) [39]–[43].

A. Transformer

The transformer was first introduced in [16] as a successor to the recurrent neural networks in sequence-to-sequence learning tasks, consisting of two major components: the positional encoding and multi-head attention mechanism. Since then, the transformer not only dominates the performance scale in the field of natural language processing but also becomes one of the leading models in multiple mainstream topics, including time series data [25]–[27] and image classification [28]. Several researchers have also adopted the design of the transformer to enhance the communication systems, such as 6G intelligent network designs [29], attention-inspired feedback codes [30], [31], signal modulation detection [32], [33], and BP-inspired semantic communications for text [34].

The heart of the transformer is a multi-head attention mechanism. By employing a learnable scaled dot-product mapping of a query-key vector pair, the model generates attention weights capturing the relation between each element to every other element within the same sequence (self-attention) or between a reference query sequence and another sequence projected into key and value vectors (cross-attention). The scaled dot-product attention can be formulated as

$$A(Q,K,V) = Softmax(\frac{QK^T}{\sqrt{d_h}})V, \tag{1}$$

where d_h is the dimension of the learned linear projections of query and key vectors, which is originally calculated by d_{model}/h in which d_{model} denotes the embedding dimension after performing positional encoding on the input sequence; h denotes the number of parallel heads during the linear projection of query, key, and value vectors.

From the basis of the scaled-dot product attention, the multi-head self-attention mechanism is then formulated as $MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O$, where $head_i = A(QW_i^Q, KW_i^K, VW_i^V)$. The linear projections of Q,K,V, and the attention output are parameterized by matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}}\times d_h}$, $W_i^K \in \mathbb{R}^{d_{\text{model}}\times d_h}$, $W_i^V \in \mathbb{R}^{d_{\text{model}}\times d_h}$, and $W_i^O \in \mathbb{R}^{hd_h\times d_{\text{model}}}$. In [16], the masked self-attention module takes the summation of causal masks and the dot product of the query-key vector pair. This mechanism was proposed to prevent the language decoder from attending to subsequent positions in the sequence, thereby preventing the network from learning only to copy the input. However, many researchers later modified this mask operation either to enhance general performance or to address domain-specific problems.

B. Error Correction Code Transformer

The authors of [15] introduced two key features, namely positional reliability encoding and code-aware self-attention, which will be discussed in this subsection.

1) Positional reliability encoding: In [15], the authors proposed a unique positional encoding method that pays more attention to the magnitude of the input sequence (i.e., the pair of channel corrupted codewords and their respective syndromes). Each dimension of $\{\tilde{\mathbf{y}}_i\}_{i=1}^{2n-k}$ in the positional encoding is regarded as the pre-processed version of the channel output $\mathbf{y}$. The encoding process projects the vector $\tilde{\mathbf{y}}$

to a d_{model} dimensional embedding $\{\boldsymbol{\phi}_i\}_{i=1}^{2n-k}$ which is defined by

$$\boldsymbol{\phi}_i = \begin{cases} |\mathbf{y}_i| W_i, & \text{if } i \leq n, \\ (1 - 2(s(\mathbf{y}))_{i-n+1}) W_i, & \text{otherwise}, \end{cases} \tag{2}$$

where $\{W_i \in \mathbb{R}^d\}_{i=1}^{2n-k}$ denotes the one-hot encoding vector corresponding to each bit position of $\tilde{\mathbf{y}}$. As was shown in [15], this embedding method yields a desirable effect during the self-attention dot product operation such that the unreliable information (i.e., low magnitude positions) would collapse to the origin, whereas the syndrome bits yield negative scaling. The proposed encoding can be viewed as a positional encoding that corresponds to the input reliability, hence the name positional reliability encoding.

2) Code-aware self-attention: Similar to the conventional decoding process, which requires the parity-check matrix, the transformer counterpart, introduced in [15], is designed to incorporate unique sparse masks capturing the patterns of the parity-check matrix for each respective code type. Since each bit of the input vector is not necessarily related to all other bits, traditional causal masking as well as non-masking would yield sub-optimal performance.

Algorithm 1 Pseudo-code of the binary mask generating algorithm

Input: Parity-check matrix $\mathbf{H}$
Output: Mask based on $\mathbf{H}$
$\mathbf{M} \leftarrow I(2n-k)$
for i = *1, 2, …,* $n-k$ **do**
  $idx \leftarrow$ where($\mathbf{H}[i] == 1$)
  for $j \in idx$ **do**
    $\mathbf{M}[n+i, j] \leftarrow 1$
    $\mathbf{M}[j, n+i] \leftarrow 1$
    for $h \in idx$ **do**
      $\mathbf{M}[j, h] \leftarrow 1$
      $\mathbf{M}[h, j] \leftarrow 1$
    end
  end
end
$\mathbf{M} \leftarrow (-\infty(\neg\mathbf{M}))$

Thus, for a given algebraic code corresponding to a parity-check matrix $\mathbf{H}$, the attention mask is denoted by $g(\mathbf{H})$: $\{0,1\}^{(n-k)\times n} \rightarrow \{-\infty, 0\}^{(2n-k)\times(2n-k)}$. First, the mask is initialized with an identity matrix. Taking reference to each row of the parity-check matrix $\mathbf{H}$ at a time, for every bit *one* in $\mathbf{H}$, a pair of positions (simultaneously considering row and column) in $g(\mathbf{H})$ is unmasked, i.e., assigned with *one*. Consequentially, this process provides a symmetric mask containing information about every pairwise bit relation. The construction of the mask $g(\mathbf{H})$ is explained by Algorithm 1.

C. U-Net

In the field of computer vision, a popular convolutional network architecture called U-Net was proposed in [24] for semantic segmentation in images. This architecture is designed with one side for down-sampling and the other side for up-sampling operations. The features extracted by the down-sampling side are also forwarded and learned by the up-sampling side through skip connections, concatenation, and re-scaling modules. As a result of its success in the initial proposed problem set, U-Net was later combined with other neural network families to provide efficient solutions for a variety of tasks, including U-shape swin transformers (SUNet [35]) and U-Net VAE [36] for image denoising, U-Net transformer (Ds-transunet [37]) for enhanced medical image segmentation, and one-dimension convolutional U-Net [38] for times series learning. In the majority of its applications, U-Net was proposed due to its highly effective capability for segmenting different parts of the superimposed input data and extracting hidden patterns that enable restoration of distorted information. In this context, we note that U-Net architecture has great potential in noise segmentation problems such as channel decoding.

D. Variational Autoencoder

Initially introduced in [39], VAE is one of the successful deep generative models. As suggested by its naming convention, this variant of the autoencoder belongs to the family of methods called variational Bayes methods. In general, this family of methods is used to solve the variational inference problem, where a posterior distribution of unobserved latent variables given some input data is approximated by a variational distribution. VAE operates as a parameterized model to optimize the process of discovering the variational distribution across all input data points.

This process is achieved by using an encoder neural network to map the input data to a latent space. Then, a separate parameterized model learns the mean and variance of the encoded latent space to generate new samples from the prior distribution. Finally, an decoder network approximates the distribution of the input data (the variational distribution) by extracting the features from the newly generated samples. A combination of VAE and transformers had also been proposed for generative tasks [40]. In the communication point of view, VAE has also been proposed for channel coding [41], [42] and joint source-channel coding [43].

III. System Model

Under the assumption of a standard wireless communication system, the transmission process uses a linear code C to encode an input message $\mathbf{m} \in \{0,1\}^k$ into a codeword $\mathbf{x} \in \{0,1\}^n$ by taking the Galois field matrix product with a generator matrix $\mathbf{G}$, in which a summation operation is replaced by the modulo-2 operation. The linear code is defined by a binary generator matrix $\mathbf{G}$ of size $k \times n$ and a binary parity-check matrix $\mathbf{H}$ of size $(n-k) \times n$, such that $\mathbf{G}\mathbf{H}^{\mathrm{T}} = 0$, where T denotes the matrix transposition. A codeword $\mathbf{x}$, satisfying $\mathbf{H}\mathbf{x} = 0$, is then transmitted via a binary-input-symmetric-output, additive white Gaussian noise (AWGN) channel. The effect of the channel can be modeled as $\mathbf{y} = \mathbf{x}_{\mathrm{s}} + \mathbf{z}$, where $\mathbf{x}_{\mathrm{s}}$ is the modulated codeword (commonly considered binary phase shift keying (BPSK)), and $\mathbf{z}$, defined

by $\mathbf{z} \sim \mathcal{N}(0, \boldsymbol{\sigma}^2)$, represents the AWGN which is independent of the modulated codeword $\mathbf{x}_\mathrm{s}$. Without loss of generality, the modulated codeword $\mathbf{x}_\mathrm{s}$ can be obtained by $\mathbf{x}_\mathrm{s} = bipolar(\mathbf{x})$, where $bipolar(\cdot)$ denotes the conversion by $0 \to 1, 1 \to -1$, and is also obtainable as $bipolar(a) = 1 - 2a$ with input $a \in \{0, 1\}$.

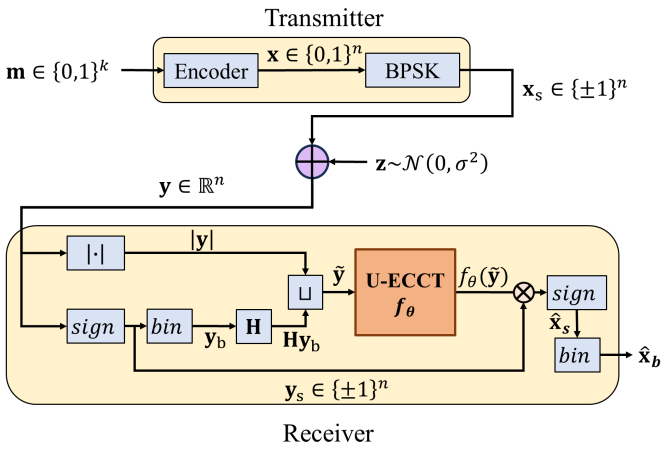


Fig. 1. System model for syndrome-based decoding framework.

At the decoder, the main objective is to provide a soft estimation $\hat{\mathbf{x}}$ of the original codeword. Thus, the decoding function can be viewed as a function of $\mathbf{y}$, which is $\hat{\mathbf{x}} = f(\mathbf{y})$. Similar to [15], the pre-processing and post-processing methods proposed by [17] are considered in this work. These methods are regarded as an extended version of symdrome-based decoding that includes channel reliabilities, aiming to overcome overfitting by eliminating the need to simulate codewords during training.

The pre-processing step modifies channel output $\mathbf{y}$ with a vector of length $2n-k$, defined as $\tilde{\mathbf{y}} = |\mathbf{y}| \sqcup s(\mathbf{y})$, where $\sqcup$ denotes the vector concatenation, $|\mathbf{y}|$ represents the element-wise magnitude of $\mathbf{y}$, and $s(\mathbf{y}) = \mathbf{H}\mathbf{y}_\mathrm{b} \in \{0, 1\}^{n-k}$ denotes the syndrome vector as a result of the Galois field matrix product between the hard-decision mapping $\mathbf{y}_\mathrm{b}$ of $\mathbf{y}$ and the parity-check matrix $\mathbf{H}$. The binary hard-decision transformation of $\mathbf{y}$ can be obtained by $\mathbf{y}_\mathrm{b} = bin(sign(\mathbf{y})) = \frac{1}{2}(1 - sign(\mathbf{y}))$, where $bin(\cdot)$ denotes the inverse of the $bipolar(\cdot)$ operation, and $sign(\cdot)$ denotes a sign function. An inverse BPSK operation, denoted as $bin(\cdot)$ is defined by $bin(b) = (1-b)/2$, where input $b \in \{\pm 1\}$ is the result of the BPSK bipolar conversion. The sign function, denoted as $sign(\cdot)$, is defined by

$$sign(c) = \begin{cases} 1, & c > 0, \\ 0, & c = 0, \\ -1, & c < 0. \end{cases}$$

The post-processing step multiplies the bipolar mapping of the channel output $\mathbf{y}$ with the decoder output to create the estimation $\hat{\mathbf{x}}$ of the original codeword $\mathbf{x}$. To be more specific, the prediction takes the form of $\hat{\mathbf{x}} = \mathbf{y} \otimes f_\theta(\tilde{\mathbf{y}})$, where $f_\theta(\cdot)$ represents the neural decoder parameterized by θ and $\otimes$ is the component-wise multiplication. The decoding process is visualized in Fig. 1.

The objective of the neural decoder is to generate a form of estimation that will reduce the effect of channel noise and recover the original codeword when multiplied with the channel output $\mathbf{y}$. With such an approach, the syndrome-based decoding proposed method in [17] focused on prediction of the multiplicative noise $\tilde{\mathbf{z}}$. Let $\tilde{\mathbf{z}}_\mathrm{s}$ denotes the soft multiplicative noise such that $\mathbf{y} = \mathbf{x}_\mathrm{s} \otimes \tilde{\mathbf{z}}_\mathrm{s}$ (proof of Lemma 1 in [18]), thereby obtaining the following expression: $\tilde{\mathbf{z}}_\mathrm{s} = \tilde{\mathbf{z}}_\mathrm{s} \otimes \mathbf{x}_\mathrm{s}^2 = \mathbf{y} \otimes \mathbf{x}_\mathrm{s}$. Hence, the target data is the binary multiplicative noise, defined by $\tilde{\mathbf{z}}_\mathrm{b} = bin(sign(\mathbf{y} \otimes \mathbf{x}_\mathrm{s}))$. Given the decoder output and its target data, the loss function is chosen as the binary cross-entropy (BCE) loss between the decoder output $f_\theta(y)$ and the binary multiplicative noise $\tilde{\mathbf{z}}_\mathrm{b}$, which is formalized as

$$\mathcal{L}_\mathrm{BCE}(\tilde{z}_\mathrm{b}, f_\theta(\tilde{y})) = -\sum_{i=1}^{n} (\tilde{z}_i \log(f_\theta(\tilde{y})) + (1 - \tilde{z}_i \log(1 - f_\theta(\tilde{y})))). \tag{3}$$

Thus, the estimation of the original codeword $\mathbf{x}$ is obtained by $\hat{\mathbf{x}} = bin(sign(f_\theta(\tilde{\mathbf{y}}) \otimes sign(\mathbf{y})))$.

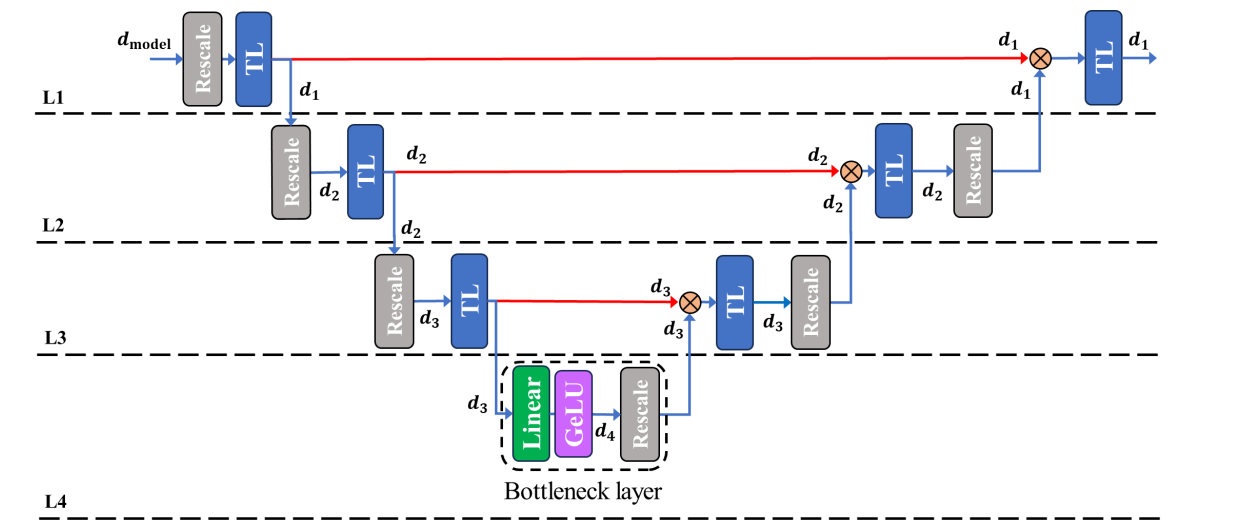
IV. Proposed Architecture

In this section, the proposed U-ECCT designs are explained in detail. To elaborate on our proposal, the subsections include the description of U-ECCT for decoding high-rate, moderate-length codes, VU-ECCT for decoding moderate-length codes at lower code rates, and the mirror-sharing strategy for model size compression.

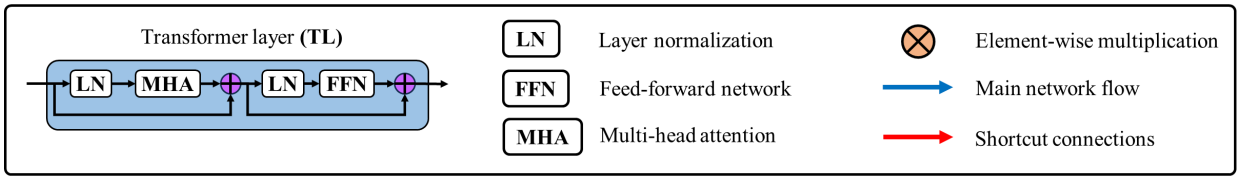
A. U-ECCT

Inpsired by the U-Net from [24], we propose a U-shaped architecture for the ECCT model with specific modifications to enable both performance improvement and parameter quantity reduction. In Fig. 2, the architecture of U-ECCT is presented by a specific network with four scaling levels. This architecture is designed as a counterpart to the standard six-layer ECCT model in [15]. The proposed model consists of the main network flow of information and the shortcut connections. The main flow is a network of sequentially connected components, including the bottleneck layer and pairs of scaling linear layer and transformer layer. The bottleneck layer is positioned at the center of the network, dividing it into two halves. Prior to the bottleneck layer, the first half of the network (left side of architecture Fig. 2) takes as input the post-embedding vector with the hidden dimension of d_model. Each pair of the scaling and transformer layers is designed with a hidden dimension that increases as the information flows toward the bottleneck layer. The second half of network (right side of architecture after the bottleneck layer in Fig. 2) then performs a reverse scaling in terms of hidden dimension.

As presented in Fig. 2, the scaling of the hidden dimension is divided into four levels: L1, L2, L3, and L4, with their hidden dimensions denoted by d_1, d_2, d_3, and d_4, respectively. The hidden dimension follows the order $d_1 < d_2 < d_\mathrm{model} < d_3 < d_4$, three of which are shared between the first half and the second half of the architecture. Similar to the U-Net of [24], the shared levels in our design also comprise the interaction between the main flow and the shortcut connections. At each level, a copy of the output on the first



(a) Overall architecture



(b) Sub-layer architecture

Fig. 2. Overview of the four-level U-ECCT architecture. (a) Overall architecture of the U-ECCT. (b) Sub-layer architecture of the transformer layers.

TABLE I
AN EXAMPLE OF PARAMETERS QUANTIFICATION FOR U-ECCT.

Levels	**Hidden dimension**	P_{LN}	P_{MHA}	P_{FFN}	$P_{\text{Transformer}}$
L1 (P_{L1})	$d_1 = \frac{1}{2}d_{\text{model}}$	d_{model}	$d_{\text{model}}^2 + 2d_{\text{model}}$	$2d_{\text{model}}^2 + \frac{5}{2}d_{\text{model}}$	$3d_{\text{model}}^2 + \frac{13}{2}d_{\text{model}}$
L2 (P_{L2})	$d_2 = \frac{3}{4}d_{\text{model}}$	$\frac{3}{2}d_{\text{model}}$	$\frac{9}{4}d_{\text{model}}^2 + 3d_{\text{model}}$	$\frac{9}{2}d_{\text{model}}^2 + \frac{15}{4}d_{\text{model}}$	$\frac{27}{4}d_{\text{model}}^2 + \frac{39}{4}d_{\text{model}}$
L3 (P_{L3})	$d_3 = \frac{5}{4}d_{\text{model}}$	$\frac{5}{2}d_{\text{model}}$	$5d_{\text{model}}^2 + 5d_{\text{model}}$	$\frac{25}{2}d_{\text{model}}^2 + \frac{25}{4}d_{\text{model}}$	$\frac{75}{4}d_{\text{model}}^2 + \frac{65}{4}d_{\text{model}}$
Transformer size (P_{L123})	$2P_{\text{L1}} + 2P_{L2} + 2P_{L3} = 57d_{\text{model}}^2 + 65d_{\text{model}}$				
L4 (P_{L4})	$d_4 = \frac{3}{2}d_{\text{model}}$	$d_3 \times d_4 + d_4 = \frac{15}{8}d_{\text{model}}^2 + \frac{3}{2}d_{\text{model}}$			
Rescale Embedding - L1 ($P_{\text{Emb-L1}}$)	$d_{\text{model}} \times d_1 + d_1 = \frac{1}{2}d_{\text{model}}^2 + \frac{1}{2}d_{\text{model}}$				
Rescale L1 - L2 (P_{R12})	$d_1 \times d_2 + d_2 = \frac{3}{8}d_{\text{model}}^2 + \frac{3}{4}d_{\text{model}}$				
Rescale L2 - L3 (P_{R23})	$d_2 \times d_3 + d_3 = \frac{15}{16}d_{\text{model}}^2 + \frac{5}{4}d_{\text{model}}$				
Rescale L4 - L3 (P_{R43})	$d_4 \times d_3 + d_3 = \frac{15}{8}d_{\text{model}}^2 + \frac{5}{4}d_{\text{model}}$				
Rescale L3 - L2 (P_{R32})	$d_3 \times d_2 + d_2 = \frac{15}{16}d_{\text{model}}^2 + \frac{3}{4}d_{\text{model}}$				
Rescale L2 - L1 (P_{R21})	$d_2 \times d_1 + d_1 = \frac{3}{8}d_{\text{model}}^2 + \frac{1}{2}d_{\text{model}}$				
Non-transformer size ($P_{\text{Non-transformer}}$)	$P_{L4} + P_{\text{Emb-L1}} + P_{R12} + P_{R23} + P_{R34} + P_{R43} + P_{R32} + P_{R21}$ $= \frac{55}{8}d_{\text{model}}^2 + \frac{11}{2}d_{\text{model}} \approx 7d_{\text{model}}^2 + 6d_{\text{model}}$				
Total size	$P_{L123} + P_{\text{Non-transformer}} = 64d_{\text{model}}^2 + 71d_{\text{model}}$				

half is used as a shortcut connection to provide supportive information to the second half of the network. In the U-Net architecture of [24], the shortcut connections are combined with the main network flow through concatenation. However, through experimentation, it is found that the concatenation is inefficient for our design in terms of dimensionality. Instead, element-wise multiplication is utilized to combine the shortcut connections with the second half of the network. Furthermore, through multiplication, the information from the first half acts as positional weights that help U-ECCT effectively narrow down the regions in the embedding space corresponding to the erroneous bits in the channel output.

The original U-Net was designed for convolutional networks, in which the data is processed in terms of height, width, and number of channels. Hence, the scaling of dimensionality is different from our design. As an example, in [24], the first half of the U-Net downscaled the height-width dimensions by half their size after each level but doubles the number of channels accordingly. Since the dimensionality of our problem is similar to that of a sequence-based problem, the scaling of U-ECCT only involves a change to the last dimension (the embedding dimension) of the data array. In the baseline ECCT [15], the hidden dimension for each sub-layer was scaled according to the embedding dimension d_{model}. Similarly, our proposed scaling design also has the hidden dimension of each level derived from the embedding dimension d_{model}. In ascending order, the hidden dimensions of U-ECCT are defined as $d_1 = \frac{1}{2}d_{\text{model}}$, $d_2 = \frac{3}{4}d_{\text{model}}$, $d_3 = \frac{5}{4}d_{\text{model}}$, and $d_4 = \frac{3}{2}d_{\text{model}}$. This design supports better parameter utilization efficiency and decoding performance compared to the baseline ECCT.

The parameter utilization is determined by the total number of parameters in the entire model. This refers to the sum of the number of weights and biases from different types of layers trained for the given task. In the sub-layer architecture of both our proposed design and the baseline ECCT in [15], three main sub-layer types are used, namely layer normalization (LN), multi-head attention (MHA), and feed-forward network (FFN). For ECCT [15], the number of parameters used by each type of sub-layer can be calculated as a function of the embedding dimension d_{model}. The building block of all sub-layers is referred to as the dense layer or fully connected layer, with d_{in} and d_{out} denoting its input and output dimensions, respectively. Its number of parameters, P_{dense}, is calculated by

$$P_{\text{dense}}(d_{\text{in}}, d_{\text{out}}) = d_{\text{in}} \times d_{\text{out}} + d_{\text{out}}. \tag{4}$$

The number of parameters for layer normalization, P_{LN}, is defined as

$$P_{\text{LN}} = 2d_{\text{model}}. \tag{5}$$

From (4) and (5), the number of parameters for MHA and FFN sublayers can be defined as

$$\begin{aligned} P_{\text{MHA}} &= 4P_{\text{dense}}(d_{\text{model}}, d_{\text{model}}) \\ &= 4 \times (d_{\text{model}}^2 + d_{\text{model}}) \\ &= 4d_{\text{model}}^2 + 4d_{\text{model}}, \end{aligned} \tag{6}$$

$$\begin{aligned} P_{\text{FFN}} &= P_{\text{dense}}(d_{\text{model}}, d_{\text{FFN}}) + P_{\text{dense}}(d_{\text{FFN}}, d_{\text{model}}) \\ &= 4d_{\text{model}}^2 + 4d_{\text{model}} + 4d_{\text{model}}^2 + d_{\text{model}} + 2d_{\text{model}} \\ &= 8d_{\text{model}}^2 + 5d_{\text{model}}. \end{aligned} \tag{7}$$

where d_{FFN} is the dimensionality of the FFN module inner-layer, which was equal to four times the embedding dimension d_{model} in [15] and [16]. The FFN component, was originally proposed in [16] along with the MHA module, is composed of a neural layer with GELU activation [44] followed by a linear layer. In this pair of layers, the output dimension of the first layer and the input dimension of the second layer are equal to d_{FFN}.

By accumulating the sub-components parameters, the total number of parameters for a transformer layer is calculated by

$$\begin{aligned} P_{\text{Transformer}} &= P_{\text{MHA}} + P_{\text{FFN}} + 2P_{\text{LN}} \\ &= 4d_{\text{model}}^2 + 4d_{\text{model}} + 8d_{\text{model}}^2 + 5d_{\text{model}} + 4d_{\text{model}} \\ &= 12d_{\text{model}}^2 + 13d_{\text{model}}. \end{aligned} \tag{8}$$

Let M denote the number of transformer layers in the model. The size of an M-layers ECCT model is determined by multiplying $P_{\text{Transformer}}$ number of parameters of each transformer layer by M number of layers used in the model architecture, which results in the following form as

$$P_{\text{ECCT}} = M \times P_{\text{Transformer}} = 12Md_{\text{model}}^2 + 13Md_{\text{model}}. \tag{9}$$

Based on ECCT parameter calculation, the size of the U-ECCT architecture can be determined by substituting d_{model} with the hidden dimension of each respective scaling level. However, due to the difference between levels, additional layers (rescale blocks in Fig. 2) are assigned between the levels to facilitate inter-level transitions. For example, a rescale layer positioned between levels d_1 and d_2 will have an input dimension of d_1 and an output dimension of d_2. Therefore, by (4), the size of the rescale layer between levels L1 and L2 is $P_{\text{R12}} = d_1 \times d_2 + d_2 = d_2 \times (d_1 + 1)$. The functions to determine size for the U-ECCT and its sub-components are summarized in Table I.

With the proposed scaling scheme, the dimension of each level is adjustable, enabling flexibility in parameter assignment. This multi-level scaling aspect of the U-shaped architecture is crucial for learning to separate combined distributions. Another crucial aspect of the proposed scaling scheme is the utilization of fewer parameters while achieving better performance than the baseline ECCT at high code rates with moderate length. In particular, for models with $M = 6$ layers, our model utilizes a total of $64d_{\text{model}}^2 + 71d_{\text{model}}$ parameters, which is fewer than the $72d_{\text{model}}^2 + 78d_{\text{model}}$ parameters used by the baseline ECCT. The model setting presented in Table I is empirically chosen for its superior performance compared to the referenced six-layer ECCT model.

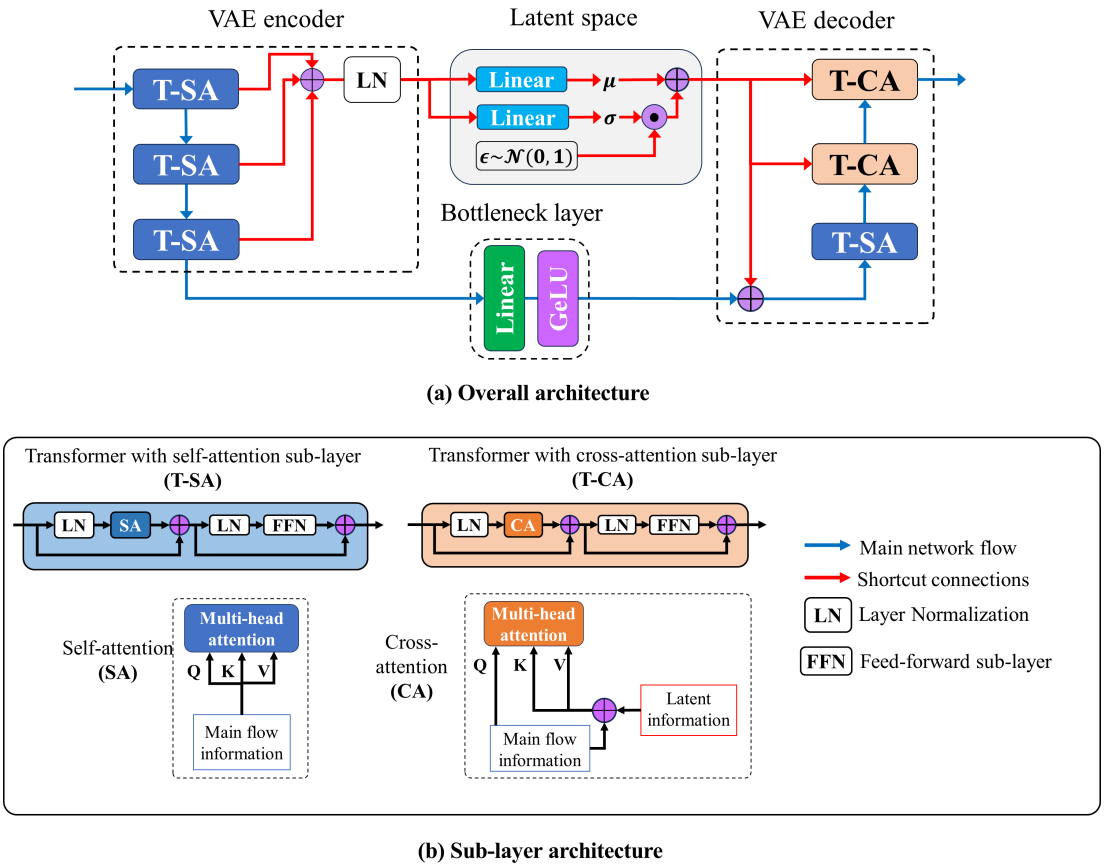


Fig. 3. Diagrams for the VU-ECCT architecture. (a) Overall architecture of the VU-ECCT. (b) Sub-layer architecture for individual transformer layer type.

B. Variational U-ECCT

The VU-ECCT architecture is proposed to improve the performance of the initially proposed model at lower code rates. In the new architecture, three modifications are applied to the U-shaped architecture regarding the shortcut connection, the dimensionality scaling, and the transformer layers on the second half of the network. Fig. 3 presents the overall and sub-layer architecture for the VU-ECCT architecture.

Firstly, the shortcut connection in the VU-ECCT architecture is not a direct forward pass as in the basic design. Instead, the variational Bayes inference method is adopted to provide a combined approximation for the learned features from the first half of the network. As shown in Fig. 3(a), the outputs of transformer layers in the first half are combined through summation and normalized before feeding to the approximation module, forming an encoder-like structure that encodes the outputs of the first half into a common latent space. The approximation mechanism adopts the re-parameterization trick proposed in [39], where the distribution of the latent space is approximated by a deterministic function, defined by

$$\begin{aligned} \tilde{\mathbf{s}} &= \boldsymbol{\mu} + \boldsymbol{\sigma} \otimes \boldsymbol{\epsilon}, \\ \boldsymbol{\epsilon} &\sim \mathcal{N}(0,1), \end{aligned} \tag{10}$$

where $\tilde{\mathbf{s}}$ is the latent vector, $\boldsymbol{\mu}$ is the learned mean vector of the latent distribution, $\boldsymbol{\sigma}$ is the learned variance vector of the latent distribution, and $\boldsymbol{\epsilon}$ is the random vector drawn from the standard normal distribution. In this case, the reparameterization trick formula is designed to generate random vectors from the location-scale distribution family. The latent vector $\tilde{\mathbf{s}}$ is fed to the second half of the network. In Fig. 3(a), the second half is structured with two types of transformer layers, denoted by T-SA (transformer with self-attention sub-layer) and T-CA (transformer with cross-attention sub-layer). First introduced by the authors of [16], the two types of attention modules were swhon to be effective in learning inter-positional relations within a sequence in a teach-forcing manner. The input to the T-SA is either the previous T-SA output (from the first half of the network) or the sum of the bottleneck layer output and latent information (at the bottom level of the second half). For the T-CA, as shown in Fig. 3(b), the query vector Q is directly taken from the preceding transformer layer, whereas the key and value vectors, K and V, are obtained by taking in the summation between the main flow information and the latent approximation. By this mechanism, the second half of the U-shaped architecture operates as a VAE decoder, extracting reliable information from the combined distribution between the main flow and the latent approximation. In particular, the latent approximation captures the common patterns across the T-SA layers on the encoder side, mitigating the adverse effect upon combining the shortcuts with the second half of the network for low rate codes when the syndrome length is $n - k \geq \frac{n}{2}$. However, this mechanism is less effective when

the syndrome length is small, rendering the performance of the VU-ECCT closer to the basic U-ECCT design.

Secondly, in the VU-ECCT architecture, the dimensionality is uniform across all layers. In the case of the basic U-ECCT, the multi-level scaling provides flexibility to adjust the model size as well as effectively distribute the parameters to important parts of the network. On the contrary, the VU-ECCT architecture becomes highly complex when applying multi-level scaling due to the requirement for additional rescale layers before and after the latent space approximation step. As a result, the model size increases significantly compared to the basic U-ECCT while performance slightly degrades compared to the VU-ECCT with a uniform hidden dimension, losing both the benefits of model reduction and performance improvement.

Finally, as part of the VAE, the objective function of the model is modified to include the KL-divergence term in addition to the cross-entropy loss. This combination can be deduced from the evidence lower bound, which was detailed in [39]. The new loss function is defined as

$$\mathcal{L}_\theta = \mathcal{L}_{\mathrm{BCE}}(\tilde{\mathbf{z}}_b, f_\theta(\check{\mathbf{y}})) + D_{\mathrm{KL}}(\mathcal{N}(\mu, \sigma^2 I) \parallel \mathcal{N}(0, \sigma_{\mathbf{y}}^2 I), \quad (11)$$

where $\mathcal{L}_{\mathrm{BCE}}$ is the binary cross-entropy (BCE) term which is obtained by (3), D_{KL} is the KL-divergence term, and $\sigma_{\mathbf{y}}$ is the standard deviation of the channel output $\mathbf{y}$. In our design, the target standard normal distribution in the KL-divergence is modified to the normal distribution with a zero mean and the standard deviation $\sigma_{\mathbf{y}}$ calculated from the observed channel output $\mathbf{y}$. The KL-divergence term between two normal distribution is defined by

$$\begin{aligned} D_{\mathrm{KL}}(\mathcal{N}(\mu_1, \Sigma_1) \parallel \mathcal{N}(\mu_2, \Sigma_2)) = \frac{1}{2} \big[tr(\Sigma_2^{-1}\Sigma_1) \\ + (\mu_2 - \mu_1)^{\mathrm{T}} \Sigma_2^{-1} (\mu_2 - \mu_1) + r + \log \frac{|\Sigma_2|}{|\Sigma_1|}, \end{aligned} \quad (12)$$

where r is the size of the random vector, which, in this case, is the hidden dimension d_{model} of the latent vector. Leveraging the formula derived by the authors of [41] (Appendix B, [41]) and the considered distributions in our case, the KL-divergence loss term can be calculated as

$$\begin{aligned} D_{\mathrm{KL}}(\mathcal{N}(\mu, \sigma^2 I) \parallel \mathcal{N}(0, \sigma_{\mathbf{y}}^2 I)) = \frac{1}{2\sigma_{\mathbf{y}}^2} \sum_{j=1}^{r} \mu_j^2 \\ - \frac{r}{2} (1 - \frac{\sigma_{\mathbf{y}}^2}{\sigma^2} + \log \frac{\sigma_{\mathbf{y}}^2}{\sigma^2}). \end{aligned} \quad (13)$$

In this formulation, the latent distribution is designed to have a similar power factor to the channel output, further enhancing the estimation of the multiplicative noise. Furthermore, during training, the derived form in (13) is tractable during back-propagation.

The mechanism of the VU-ECCT provides significant resistance to the unwanted noise provided by the extensively large syndrome length in low code rate cases. Furthermore, it is able to take advantage of the additional syndrome information to enhance the multiplicative noise estimation process. Therefore, the VU-ECCT exhibits considerably better performance in low code rate cases compared to the basic U-ECCT design. However, as a trade-off, the multi-level scaling of the basic design is not applicable for the variational model, losing much of the scaling flexibility of the U-ECCT model.

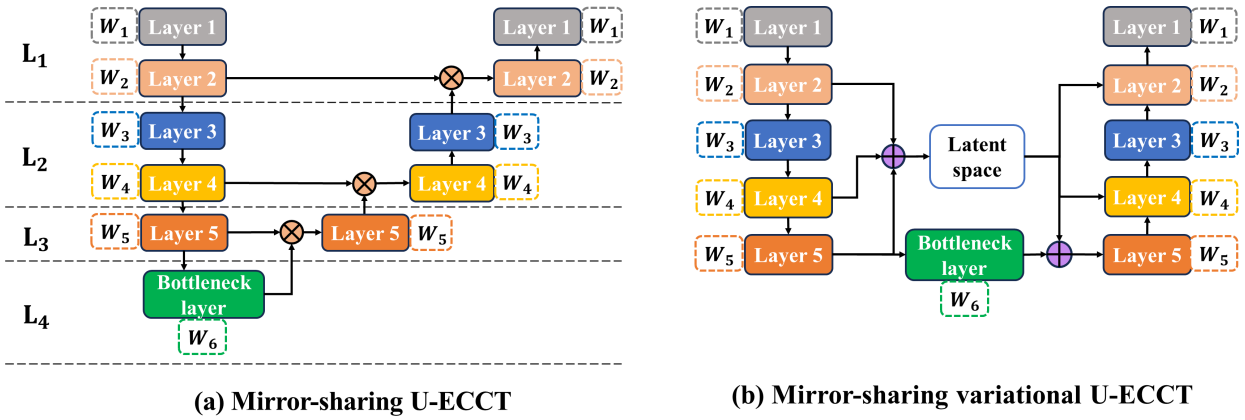
C. Mirror-sharing for model compression

In this sub-section, a new model compression strategy, named mirror-sharing, is introduced. This strategy is based on the weight-sharing method, which increases learning efficiency. While there were numerous effective weight-sharing strategies proposed in previous works, we came to the conclusion that, in order to effectively apply weight-sharing to the setting of U-ECCT, a new weight-sharing scheme is required.

In our weight-sharing strategy, each transformer layer in the first half of the U-shaped architecture is assigned an independent set of parameters. For the second half of the architecture, we reuse the parameter sets from the first half. In this structure, the proposed weight-sharing method is called mirror-sharing since one side of the network mimics the parameters of the other side in an U-shaped architecture that has reflective characteristics regarding dimensionality. During training, the first half and the second half of the U-shaped model share the same set of parameters, or, in other words, this set of parameters is trained twice during the forward and backward propagation. This means that when considering an M-layers model, only $\frac{M}{2}$ sets of parameters are used, which results in a new model with only half the number of parameters. Nevertheless, depending on the computational power required for specific cases, the sharing rate needs to be adjusted. Furthermore, the weight-sharing technique cannot be applied to rescale layers since they are not reflective in dimensionality. Each rescale layer has a unique dimensionality, which is highly complex when applying the weight-sharing technique.

Fig. 4 shows the mirror-sharing strategy for U-ECCT architecture with $M = 10$. A fully mirror-sharing strategy is shown in Fig. 4, where both sides of the U-shaped model share the same five sets of parameters for transformer layer, denoted by W_1, W_2, W_3, W_4, and W_5. The parameters set W_6 of the bottleneck layer in Fig. 4 is not subject to sharing since this layer is used only once in the entire model. Five sets of parameters are specifically chosen for sharing to reduce model size and obtain optimal performance.

Table II presents the model size comparison models with and without mirror-sharing for basic and VU-ECCT. Here, three different scaling settings are presented for basic U-ECCT, which are Type I, Type II and Type III, and a mirror-sharing setting for VU-ECCT. Among the three settings for U-ECCT, Type I is the same setting presented in Table. I, whereas Type II and Type III are settings with reduced size and increased size, respectively. Without mirror-sharing, only Type II provides a significant reduction to the overall model size. However, when mirror-sharing is applied, the model size is significantly reduced. Especially for the Type III model, the model size is significantly improved with mirror-sharing, as it now uses fewer parameters than the baseline ECCT as opposed to the model before applying the proposed sharing scheme. Similarly, the mirror-sharing strategy is also applicable for the



(a) Mirror-sharing U-ECCT

(b) Mirror-sharing variational U-ECCT

Fig. 4. The proposed mirror-sharing strategy for the U-ECCT variants. (a) Mirror-sharing is applied to the basic U-ECCT design. (b) Mirror-sharing applied to VU-ECCT

TABLE II
MODEL SIZE COMPARISON BETWEEN MODELS WITH AND WITHOUT MIRROR-SHARING. MODELS WITHOUT MIRROR-SHARING ARE CONFIGURED WITH $M = 6$ LAYERS. THE MODELS WITH MIRROR-SHARING ARE CONFIGURED WITH $M = 10$ LAYERS WITH $\frac{M}{2}$ UNIQUE PARAMETER SETS SHARED BETWEEN THE TWO HALVES OF THE U-SHAPED ARCHITECTURE AS SHOWN IN FIG. 4.

Setting indicator	Model setting	Without mirror-sharing	With mirror-sharing
U-ECCT Type I	$d_1 = \frac{1}{2}d_{\text{model}}$ $d_2 = \frac{3}{4}d_{\text{model}}$ $d_3 = \frac{5}{4}d_{\text{model}}$ $d_4 = \frac{3}{2}d_{\text{model}}$	$P_{\text{U-ECCT-I}} = 64d^2_{\text{model}} + 71d_{\text{model}}$ $P_{\text{U-ECCT-I}} < P_{\text{ECCT}}$ $\Delta P = 8d^2_{\text{model}} + 7d_{\text{model}}$	$P'_{\text{U-ECCT-I}} = 46d^2_{\text{model}} + 55d_{\text{model}}$ $P'_{\text{U-ECCT-I}} < P_{\text{U-ECCT-I}} < P_{ECCT}$ $\Delta P' = 26d^2_{\text{model}} + 23d_{\text{model}}$
U-ECCT Type II	$d_1 = \frac{1}{2}d_{\text{model}}$ $d_2 = \frac{3}{8}d_{\text{model}}$ $d_3 = \frac{3}{4}d_{\text{model}}$ $d_4 = d_{\text{model}}$	$P_{\text{U-ECCT-II}} = 25d^2_{\text{model}} + 47d_{\text{model}}$ $P_{\text{U-ECCT-II}} < P_{\text{ECCT}}$ $\Delta P = 47d^2_{\text{model}} + 31d_{\text{model}}$	$P'_{\text{U-ECCT-II}} = 19d^2_{\text{model}} + 38d_{\text{model}}$ $P'_{\text{U-ECCT-II}} < P_{\text{U-ECCT-II}} < P_{ECCT}$ $\Delta P' = 53d^2_{\text{model}} + 40d_{\text{model}}$
U-ECCT Type III	$d_1 = \frac{1}{2}d_{\text{model}}$ $d_2 = \frac{3}{4}d_{\text{model}}$ $d_3 = \frac{3}{2}d_{\text{model}}$ $d_4 = \frac{7}{4}d_{\text{model}}$	$P_{\text{U-ECCT-III}} = 83d^2_{\text{model}} + 79d_{\text{model}}$ $P_{\text{U-ECCT-III}} > P_{\text{ECCT}}$ $\Delta P = 11d^2_{\text{model}} + d_{\text{model}}$	$P'_{\text{U-ECCT-III}} = 56d^2_{\text{model}} + 60d_{\text{model}}$ $P'_{\text{U-ECCT-III}} < P_{\text{ECCT}} < P_{\text{U-ECCT-III}}$ $\Delta P' = 16d^2_{\text{model}} + 18d_{\text{model}}$
Variational U-ECCT	$d_1 = d_2 = d_3 = d_4 = d_{\text{model}}$	$P_{\text{VU-ECCT}} = 74d^2_{\text{model}} + 80d_{\text{model}}$ $P_{\text{VU-ECCT}} \approx P_{\text{ECCT}}$ $\Delta P = 2d^2_{\text{model}} + 2d_{\text{model}}$	$P'_{\text{VU-ECCT}} = 60d^2_{\text{model}} + 65d_{\text{model}}$ $P'_{\text{VU-ECCT}} < P_{\text{ECCT}} < P_{\text{VU-ECCT}}$ $\Delta P' = 12d^2_{\text{model}} + 13d_{\text{model}}$

VU-ECCT, as shown in Fig. 4(b) and the last row of Table II. However, since the hidden dimension is uniform across all transformer layers of the VU-ECCT architecture, the mirror-sharing strategy provides less reduction compared to sharing model for the U-ECCT.

V. EXPERIMENTATION

A. Setup and training

The experimentation involves comparing the performance of the proposed models with the baseline ECCT [15] and BP [45]. Similar to [15], our design is evaluated with three classes of codes, namely low-density parity-check (LDPC) codes, Bose-Chaudhuri-Hocquenghem (BCH) codes and polar codes. The parity-check matrices for all codes are taken from [49]. For polar codes exclusively, successive cancellation list (SCL) [47] and CRISP [48] are included for comparison. For LDPC exclusively, min-sum (MS) algorithm [46] and the state-of-the-art model-driven normalized min-sum neural decoder (NNMS) [7] are included for comparison. The model from [7] used for comparison is the shared NNMS with Leaky

TABLE III
HYPERPARAMETERS SETTING FOR ALL METHODS. (L: THE LIST SIZE OF THE SCL DECODING, b: NUMBER OF CYCLIC REDUNDANCY CHECK (CRC) BITS ADDED TO THE POLAR CODEWORD IN THE CRC-AIDED SCL DECODING SCHEME, I: THE NUMBER OF ITERATIONS FOR BP-BASED METHODS, M: THE NUMBER OF LAYERS IN NEURAL NETWORK MODELS, d: THE HIDDEN DIMENSION OF THE RNN MODEL UTILIZED IN CRISP [48])

Model	Hyperparameter	Codes
SCL / CA-SCL [47]	$L = 8, 128$ $b = 8$	Polar
CRISP [48]	$M = 2, 6$ $d = 512$	
MS [46]	$I = 50$	LDPC
SNNMS-LR-Q [7]	$I = 10$ $M = 22$	
BP [45]	$I = 50$	LDPC BCH Polar
ECCT [15]	$d_{\text{model}} = 128$ $d_{\text{ff}} = 4 \times d_{\text{model}}$ $h = 8$ $M = 6$	
U-ECCT VU-ECCT	$d_{\text{model}} = 128$ $d_{\text{ff}} = 4 \times d_{\text{model}}$ $h = 8$ $M = 6$ (no mirror-sharing) $M = 6, 10$ (mirror-sharing)	

ReLU and a 12-bit quantizer (SNNMS-LR-Q). The baseline ECCT [15] model is trained and tested using the transformer architecture with $M = 6$ layers, $h = 8$ attention heads, and embedding dimension $d_{\text{model}} = 128$. Our designs and the baseline ECCT have the same number of attention heads and embedding dimension. For the models without mirror-sharing, our models are trained with configurations similar to those of the baseline. For mirror-sharing cases, our models are configured with either $M = 6$ layers or $M = 10$ layers, with $\frac{M}{2}$ sets of parameters shared between the two halves of the U-shaped model. Details of the hyperparameters settings for all comparing methods are shown in Tab. III. All simulation tasks are performed on a workstation equipped with an AMD Ryzen Threadripper Pro 5995WX processor, 512GB of RAM, and two NVIDIA RTX 4090-24GB GPUs. However, training and testing are executed on individual GPU independently. All simulations are programmed using the PyTorch framework [50].

For training ECCT-based models, data samples are generated with v samples per batch for 1000 batches, which makes up $v \times 1000$ samples. Due to limitations in dedicated GPU memory, the number of samples is adjusted based on code length and code rates. Codes with $n \geq 256$ and $R \leq 0.5$ force the input dimension of the neural network to exceed the computational limit of our current available hardware. Therefore, the number of samples and embedding dimension are adjusted for specifically memory-consuming cases. For codes with $n \geq 256$ and $R \leq 0.5$, $v = 96$ samples are generated per batch. For all cases with $n \geq 512$ ($n = 512, 576, 1024, 1056$), the number of samples per batch is $v = 32$ and $v = 16$, respectively, with the exception of Polar (1024, 342) using $v = 8$ samples per batch and an embedding dimension of $d_{\text{model}} = 64$. Baseline ECCT and our proposed models are trained with all-zero codewords as it is shown to be sufficient in [15], [17]. For CRISP [48] and SNNMS-LR-Q [7], training are conducted follows the reported settings in their respective articles. As suggested in [7], [48], instead all-zero codewords, random binary messages encoded using the generator matrix of each code type are utilized. For CRISP, $v \times 1000$ batches with batch size of $v = 4096$ samples are used for training over 120000 epochs to obtained the reported results with $n \leq 64$. However, such number of samples are infeasible for simulation on our current hardware when $n \geq 128$. Therefore, for large code lengths, CRISP model is trained with $v = 512$ samples per batch with 300 batches over 1000 epochs to keep the total number of samples close to that of ECCT-based models while keeping the batch size large as suggested in [48]. All neural network models are trained on noise-corrupted codewords with the energy per bit to noise power spectral density ratio E_b/N_0 generated within the range of 2dB to 7dB. The Adam

TABLE IV
CALCULATED MODEL SIZE OF ECCT [15] AND U-ECCT VARIANTS WITH CHOSEN $d_{\text{model}} = 128$. ECCT MODELS AND PROPOSED MODELS WITHOUT MIRROR-SHARING USE $M = 6$ LAYERS, WHILE MIRROR-SHARING MODELS USE $M = 10$ LAYERS. THE SIZE DIFFERENCE COLUMNS CONSISTS OF MODEL SIZE DIFFERENCE IN PERCENTAGE BETWEEN ECCT AND THE MODELS IN THE PRECEDING COLUMN TO THE RIGHT.

Setting indicator	**Without mirror-sharing**	**Size difference (%)**	**With mirror-sharing**	**Size difference (%)**
ECCT [15]	1189632	N/A	N/A	N/A
U-ECCT Type I	1057664	↓ 11%	760704	↓ **36%**
U-ECCT Type II	415616	↓ 65%	316160	↓ **73%**
U-ECCT Type III	1370496	↑ 13%	925184	↓ **22%**
Variational U-ECCT	1222656	↑ 2.7%	991360	↓ **11%**

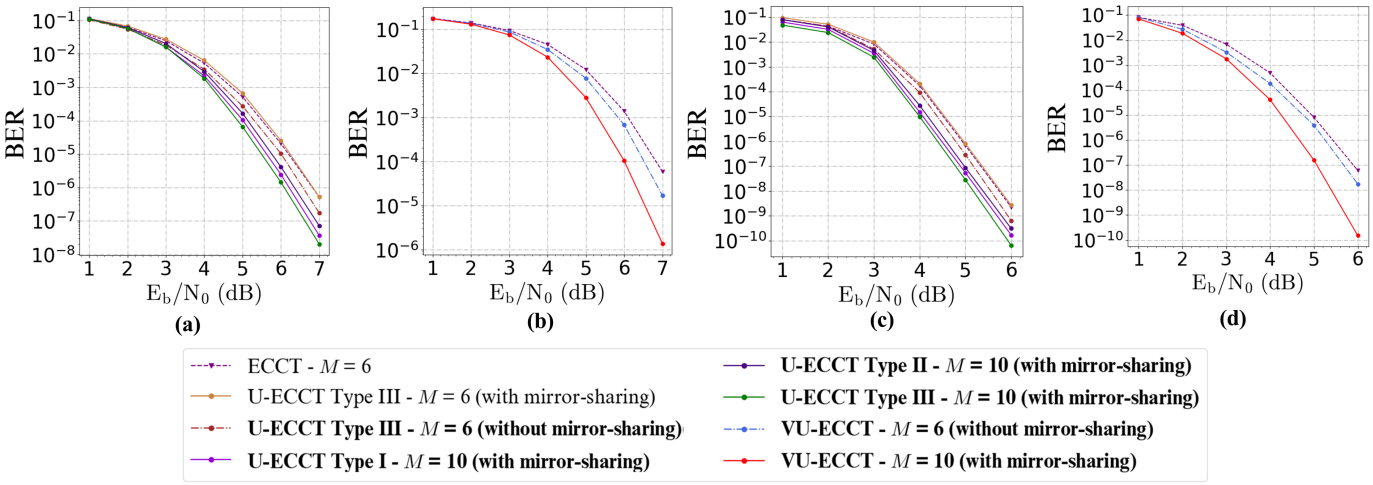


Fig. 5. BER performance for U-ECCT with different mirror-sharing strategies. (a) Mirror-sharing vs. no mirror-sharing for U-ECCT with Polar (256, 128). (b) Mirror-sharing vs. no mirror-sharing for VU-ECCT with Polar (512, 171). (c) Mirror-sharing vs. no mirror-sharing for U-ECCT with LDPC (576, 432). (d) Mirror-sharing vs. no mirror-sharing for VU-ECCT with LDPC (155, 64)

optimizer [35] is used with initial learning rate of 10^{-4}. If not stated otherwise, training is conducted over 1000 epochs using the cosine annealing weight decay method [38] without a warm restart until the learning rate reaches 5×10^{-7}.

For inference, the generator matrix $\mathbf{G}$ is obtained from the respective standard form parity-check matrix and used to generate the codewords through the Galois field matrix product, defined by $\mathbf{x} = \mathbf{mG}$. With this operation, v samples are generated for each test batch until the number of error frames reaches 100 frames to obtain accurate bit error rate (BER) results. Similarly, for BP and MS with 50 iterations, Monte-Carlo simulations are conducted with a total of 10^7 samples to ensure accurate results across all E_b/N_0 regions. All methods are tested over the E_b/N_0 range between 1dB and 7dB.

B. Significance of mirror-sharing in U-ECCT models

The proposed mirror-sharing strategy described in Section III provides a solution to configure deeper U-ECCT networks without increasing the number of parameters. In Figs. 5(a) and 5(c), the BER performance is shown for the three scaling types in Table II. Compared to the baseline ECCT and the non-sharing U-ECCT model, all of our mirror-sharing variants are superior in either decoding performance or model size. For Type II model, the number of parameters equates to 27% that of the ECCT [15] (a reduction of 73% total parameters), yet its performance is still superior to the baseline model. For the other two variants, not only do they provide model reduction (36% for Type I and 22% for Type III), but they also outperform the baseline ECCT design by a considerable difference. This demonstrates that the mirror-sharing U-ECCT enables significant improvements compared to the baseline ECCT. Furthermore, the superior results highlight the significance of neural networks depth (number of layers) to performance as opposed to their width (total number of parameters). With all variants having an advantage in both performance and model size, the strong improvement of our design in model efficiency is proven.

The effectiveness of mirror-sharing combined with U-ECCT is further proven by the result of VU-ECCT models in Figs. 5(b) and 5(d), where the mirror-sharing U-ECCT significantly outperforms both its non-sharing variant and the baseline ECCT. Nevertheless, the VU-ECCT without mirror-sharing demonstrates its improvement over baseline ECCT with a noticeable gain in performance.

Table IV presents the calculated size of all tested neural network models based on the chosen embedding dimension of $d_{\text{model}} = 128$. Among the non-sharing models, only Type I and Type II models allow the use of fewer parameters. In the case of mirror-sharing, model size reduction is enabled on all proposed models. Coupled with the advantage in performance, the U-ECCT variants are much more efficient in utilizing

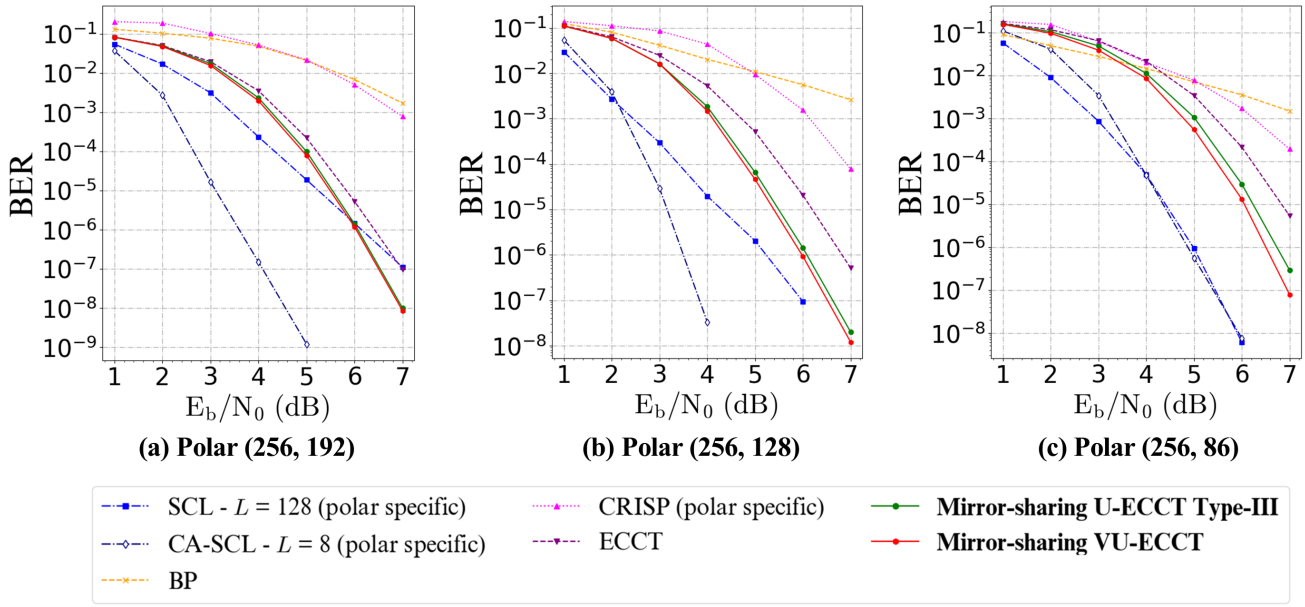


Fig. 6. BER performance results for the baseline methods and U-ECCT models on polar codes with a code-length of 256 at different code rates. (a) BER for Polar (256, 192) with code rate $R = \frac{3}{4}$. (b) BER for Polar (256, 128) with code rate $R = \frac{1}{2}$. (c) BER for Polar (256, 86) with code rate $R = \frac{1}{3}$.

the parameters for multiplicative noise estimation. The results further prove the efficiency as well as superiority of the U-ECCT design compared to the baseline ECCT.

C. BER performance of U-ECCT variants

By testing the proposed designs on a variety of coding cases, it has been verified that all mirror-sharing U-shaped variants provide considerable improvements over the baseline ECCT. In Fig. 6, the BER performance results tested on different code rates for Polar codes with a code length of 256 are plotted for all considered methods. Except for SCL and CA-SCL, it can be observed that both U-ECCT and VU-ECCT outperform other methods, specifically within the E_b/N_0 range of 4dB to 7dB. However, for Polar (256,192) in Fig, 6(a), the proposed models begin to match at $E_b/N_0 = 6$dB and outperform non-CRC SCL decoding at $E_b/N_0 = 7$dB. It is worth noting that with SCL decoding, a list size of $L = 32$ is sufficient in achieving near-MAP performance. Nevertheless, compared to CRC-aided SCL, there still is a significant difference. However, compared to a SCL-inspired model like CRISP, which achieved near-SCL performance at short code lengths, our model shows a significant gain in decoding performance. Furthermore, the VU-ECCT model demonstrates its advantage for low code rates, offering significant performance gains compared to both the U-ECCT and the baseline ECCT [15]. The advantage of our model over the state-of-the-art learning-based model is further confirmed by comparing the execution time and complexity with baseline models in section V.D.

In Fig. 7, BER results are shown for ECCT and variants of U-ECCT on larger code-length. The proposed models also exhibit better performance than BP, MS, and other learning-based models at larger code lengths. For polar codes with $n = 512$ and $n = 1024$, the average performance gains of the U-ECCT models to the baseline ECCT are 0.6dB in E_b/N_0 between $BER = 10^{-5}$ and $BER = 10^{-8}$ in Fig. 7(a), which is 0.2dB higher compared to the average performance gains at the smaller code length of $n = 256$ in Fig. 6(a). For the three cases of LDPC codes in Figs. 7(d), 7(e), and 7(f), the average performance gains of our models over ECCT are 0.6dB for LDPC (512, 256) and 0.5dB for LDPC (256, 128). Even at lower E_b/N_0 values of 3dB, where the noise level is higher, the U-ECCT models still achieve a 0.4dB gain for LDPC (512, 256) and a 0.3dB gain for LDPC (256, 128). Furthermore, our model outperforms MS and SNNMS-LR-Q, both of which already surpass the baseline ECCT. It can also be observed that for cases with $R \geq \frac{1}{2}$ in Figs. 6(a), 6(b), 7(a), 7(d), 7(e), and 7(f), the VU-ECCT model does not enable any significant improvements to the existing U-ECCT design. However, for low rate codes of Polar (256, 86) in Fig. 6(c), Polar (1024, 342) in Fig. 7(b), and Polar (512, 171) in Fig. 7(c), VU-ECCT model demonstrates considerably better performance than the non-variational model, showcasing the combined advantage of the U-shaped model and the VAE-like structure. While the performance differences between SCL-based methods and the other methods become more pronounce with large code lengths, it is important to note that ECCT-based models are forced to be trained on very limited number of samples due to current hardware constraints and are possible to achieve better performance given larger batch size for training.

Table V presents the performance results of baseline ECCT and our proposed mirror-sharing U-ECCT variants for a variety of code structures at various code-length and coding rates. As shown in Table V, improvements compared to BP, MS, and learning-based models can be observed over all code-length cases. However, larger gains are observed with lengths from 256 to 1024). Evidently, based on the obtained results, the mirror-sharing U-ECCT variants outperform the baseline ECCT in almost all cases, some less than others. For short-

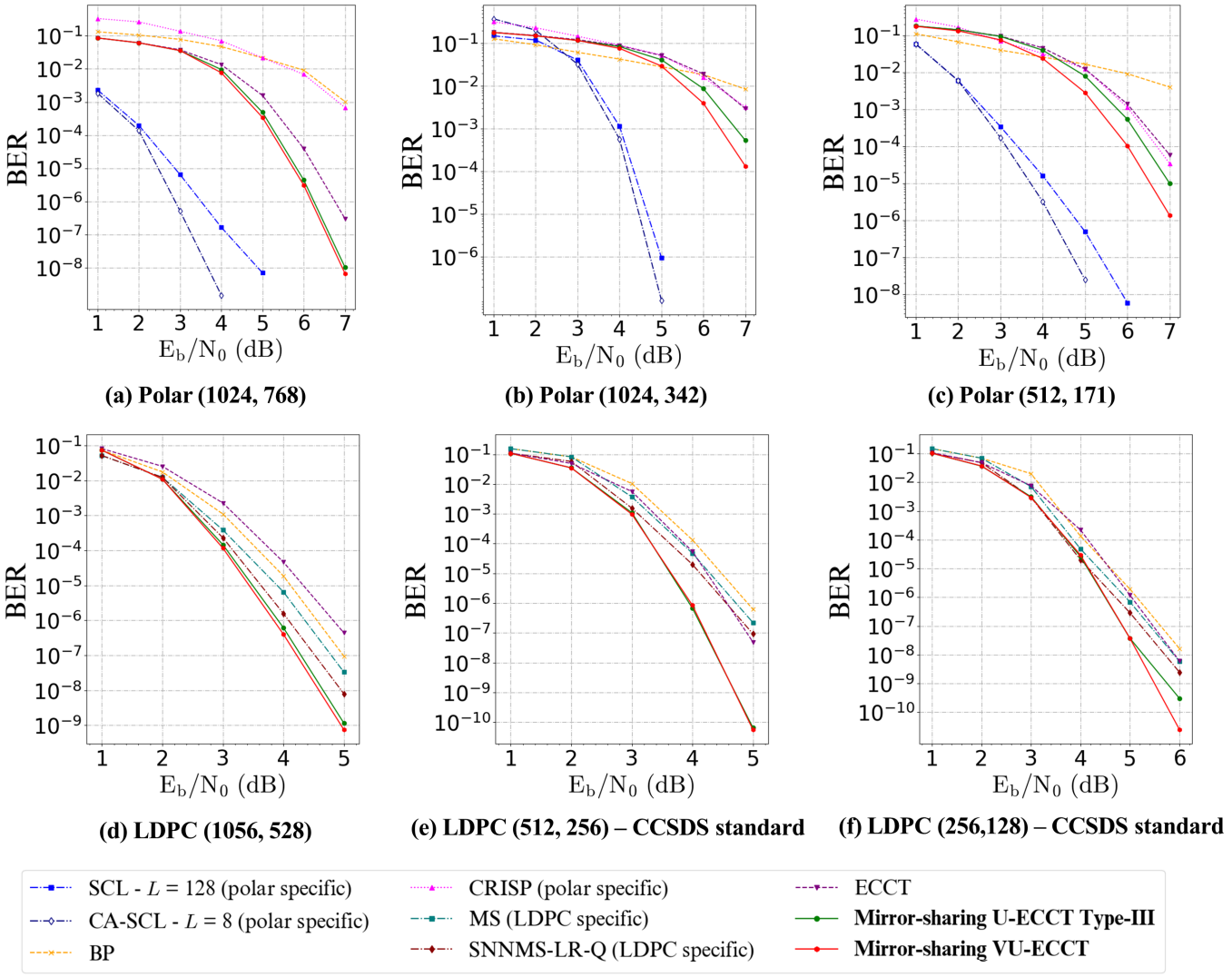


Fig. 7. BER performance results for baseline methods and U-ECCT variants on large code length. (a) Polar (1024,768). (b) Polar (512,384). (c) Polar (512,171). (d) LDPC (1056,528). (e) LDPC (512,256). (f) LDPC (256,128).

length polar codes, the proposed models outperform non-CA SCL decoding and closely performs at the level of CA-SCL, showing the potential of our models to compete with state-of-the-art polar decoding when trained with an adequate number of samples.

D. Complexity analysis

In addition to performance comparison, the quality of the methods is evaluated in terms of complexity. Since the considered methods are diverse, to ensure a fair analysis across all methods, multiply-accumulative (MAC) operation, also known as multiply-add (MAD) operation, is chosen as our complexity metric. The MAC metric account for the number of multiplication and addition operations required by a given computational model. For neural network models, the basic unit of computation is a dense layer with its MAC operation defined as

$$\mathcal{MAC}_{\text{dense}} = l \times d_{\text{input}} \times d_{\text{output}}. \tag{14}$$

In this work, The quantification of MACs for all considered learning-based models is based solely on the weight matrices, omitting the biases for simplicity of calculation. Given the length of the input sequence $\tilde{\mathbf{y}}$ as $2n-k$ and the embedding dimension as d_{model}, considering all operating steps, such as the parallel linear projections of the self-attention input to Q, K, and V vectors, the softmax dot-product attention, and the inverse projection at the output, the MAC of the MHSA is defined by

$$\begin{aligned}\mathcal{MAC}_{\text{MHSA}} &= (2n-k)^2 d_{\text{model}} + 4(2n-k)d_{\text{model}}^2 \\ &= (2n-k)^2 d_{\text{model}} + 4(2n-k)d_{\text{model}}^2.\end{aligned} \tag{15}$$

Subsequently, the MAC of FFN module can be quantified as

$$\begin{aligned}\mathcal{MAC}_{\text{FFN}} &= (2n-k) \times d_{\text{model}} \times 4d_{\text{model}} \\ &= 4(2n-k)d_{\text{model}}^2.\end{aligned} \tag{16}$$

From (15) and (16), the MAC for ECCT and our proposed models can be derived as

$$\mathcal{MAC}_{\text{Transformers}} = (2n-k)^2 d_{\text{model}} + 8(2n-k)d_{\text{model}}^2. \tag{17}$$

TABLE V
ADDITIONAL PERFORMANCE RESULTS OF ECCT [15] AND OUR PROPOSED U-ECCT VARIANTS WITH MIRROR-SHARING. THE RESULTS FOR U-ECCT ARE TESTED ON THE TYPE III MODEL. BOTH U-ECCT VARIANTS ARE CONFIGURED WITH MIRROR-SHARING. TESTED CODE STRUCTURES INCLUDE BCH CODES, LDPC CODES, AND POLAR CODES. THE RESULTS ARE PRESENTED IN NEGATIVE NATURAL LOGARITHM OF BER FOR E_b/N_0 CASES FROM 4dB TO 6dB (LARGER VALUES ARE BETTER).

Code types	E_b/N_0 (dB)	BP	SCL	CA-SCL	MS	SNNMS-LR-Q	CRISP	ECCT	U-ECCT	VU-ECCT
Polar (512, 384)	4	3.45	**12.19**	7.61	N/A	N/A	4.17	4.82	5.28	5.32
	5	5.24	**18.42**	11.14	N/A	N/A	5.41	7.40	8.50	8.79
	6	7.37	**22.33**	14.17	N/A	N/A	8.89	11.09	13.07	13.09
Polar (128, 43)	4	4.32	6.07	**9.76**	N/A	N/A	2.32	5.80	5.88	6.44
	5	5.47	8.43	**14.62**	N/A	N/A	4.91	7.94	8.17	8.81
	6	7.39	10.75	**19.27**	N/A	N/A	7.56	10.66	10.96	11.73
Polar (64, 48)	4	4.74	6.43	**7.41**	N/A	N/A	6.22	6.36	6.44	6.40
	5	5.94	8.56	**10.50**	N/A	N/A	8.15	8.44	**8**.56	8.51
	6	7.42	11.32	**14.41**	N/A	N/A	10.97	11.10	11.98	11.76
Polar (64, 32)	4	4.26	4.46	**8.14**	N/A	N/A	4.32	6.99	7.37	7.10
	5	5.38	6.05	**11.14**	N/A	N/A	5.75	9.44	10.27	9.91
	6	6.50	8.32	**14.25**	N/A	N/A	7.82	12.32	13.22	13.24
Polar (64, 22)	4	5.49	5.56	7.21	N/A	N/A	5.24	7.34	7.67	**8.07**
	5	6.42	7.53	10.35	N/A	N/A	6.03	9.84	10.68	**11.17**
	6	8.27	9.58	14.51	N/A	N/A	9.24	13.10	14.08	**14.57**
LDPC (576, 480)	4	2.75	N/A	N/A	2.75	8.57	N/A	7.35	**8.04**	7.89
	5	3.54	N/A	N/A	3.60	13.31	N/A	12.74	**14.61**	14.18
	6	6.62	N/A	N/A	6.94	18.37	N/A	18.55	**22.33**	21.93
LDPC (121, 80)	4	4.96	N/A	N/A	4.00	7.54	N/A	7.41	7.84	**7.82**
	5	8.50	N/A	N/A	6.45	11.85	N/A	11.51	12.32	**12.58**
	6	13.80	N/A	N/A	10.24	17.14	N/A	16.44	18.24	**18.53**
BCH (255, 163)	4	2.87	N/A	N/A	N/A	N/A	N/A	4.35	4.35	**4.35**
	5	3.29	N/A	N/A	N/A	N/A	N/A	6.19	6.29	**6.34**
	6	4.88	N/A	N/A	N/A	N/A	N/A	9.34	9.67	**9.69**
BCH (127, 64)	4	3.04	N/A	N/A	N/A	N/A	N/A	3.40	3.45	**3.47**
	5	3.86	N/A	N/A	N/A	N/A	N/A	4.59	4.67	**4.71**
	6	5.39	N/A	N/A	N/A	N/A	N/A	6.54	6.65	**6.80**
BCH (63, 51)	4	4.51	N/A	N/A	N/A	N/A	N/A	5.66	**5.80**	5.73
	5	5.82	N/A	N/A	N/A	N/A	N/A	7.89	**8.25**	8.02
	6	7.42	N/A	N/A	N/A	N/A	N/A	11.01	**11.53**	11.34

The baseline ECCT used in simulation is configured with an architecture of stacked $M = 6$ layers. Therefore, the MAC of ECCT is defined by $M \times \mathcal{MAC}_{\text{Transformers}}$. For VU-ECCT, the factor M is replaced with $\frac{M}{2}$ and the MAC of the latent approximating networks is included. For the U-ECCT Type-III model, the MAC is calculated considering the rescale factor at each level alongside the additional MAC for the rescale layers. Apart from the baseline ECCT and our proposed models, the MAC of other methods is referenced from the complexity analysis reported in the original article for each respective model. All MAC quantification, exact calculation at various code lengths, inference time per sample, and training time per epoch are reported in Table VI.

Table VI shows complexity quantification for all models considered in this work. Among the learning-based models, CRISP utilizes the highest number of MAC units, making it significantly less efficient than other models. At code length $n = 1024$, despite CRISP using fewer MAC than baseline ECCT, its actual training time is significantly due to its use of various sequential-based computations including the use of RNN cells and curriculum-guided mechanism. Furthermore, to achieve the reported results for CRISP at short code length, the original paper used a large batch size (4096 samples per batch), 1000 batches per epoch, and up to 120000 epochs. Compared to the ECCT-based models, including our proposed models, which use 128 batches under same number of batches and only 1000 epochs, CRISP is highly inefficient as it achieves only near-SCL performances at short-length polar codes. For SNNMS-LR-Q, despite its low MAC units, this model trades spatial complexity for time complexity as it employs various looping mechanisms for summation over sparse connections between variable nodes and check nodes, as well as unfolding iterations into stacking sequential neural layers. Therefore, its training time is also sub-optimal. While ECCT-based models use significantly more MAC units than classical decoders, the actual inference times of ECCT-based models are much better, which enhances the appeal toward these models, particularly for our proposed designs. For a fair comparison with the classical decoding algorithms, training and testing time for learning-based models are also obtained under CPU-only conditions. It is observed that decoding time per codeword for learning-based models are comparable to SCL decoding, surpassing BP-based approaches. Among the learning-based approaches, SNNMS-LR-Q benefits from CPU-only conditions due to avoiding the overhead of excessive data transfer between CPU and GPU. Nevertheless, ECCT-based methods still demonstrate fast decoding times and lowest training times under CPU-only conditions.

VI. CONCLUSION

As model-free neural-based ECC decoding schemes are on the rise, we propose U-ECCT models to enhance the

TABLE VI
MAC COMPARISONS BETWEEN THE CONSIDERED MODELS.
NOTATIONS ARE REUSED FROM THE DEFINITION IN TABLE III. ADDITIONAL NOTATION INCLUDES: e_i: THE NUMBER OF "ONES" (EDGES) IN THE $i-th$ COLUMN OF THE PCM. EXACT QUANTIFICATION OF MAC ARE CALCULATED FOR CODE LENGTHS $n = 64, 256, 1024$ WITH CODING RATE $R = \frac{1}{2}$. TRAINING AND INFERENCE TIMES ARE OBTAINED FROM SIMULATION WITH CODE LENGTH $n = 1024$ (POLAR CODES), $n = 1056$ (LDPC CODES) AND CODE RATE $R = \frac{3}{4}$

Model	**MAC calculation**	**MAC** ($\times 10^6$)			**Inference time** (seconds / sample)	**Training time** (seconds / epoch)
		64	256	1024		
BP [45]	$2Inlog_2(n)$	0.15	0.82	4.10	CPU: 2.66	N/A
SCL [47]	$Lnlog_2(n)$	0.05	0.26	1.31	CPU: 0.108	N/A
MS [46]	$I + \sum_{i=1}^{n}(e_i(e_i-1)I+1)I$	0.85	24.12	649.42	CPU: 3.04	N/A
SNNMS-LR-Q [7]	$n(I+1)+n$ $+\sum_{i=1}^{n}(e_i(e_i-1)+1)I$	0.46	12.46	326.42	CPU: 0.057 GPU: 0.258	CPU: 10175 GPU: 45912
CRISP [48]	$n(2d(n+1)+6d^2)$	104.92	470.02	2684.35	CPU: 0.188 GPU: 0.026	CPU: 11581 GPU: 1465
ECCT [15]	$M(2n-k)^2 d_{\text{model}}$ $+8M(2n-k)d_{\text{model}}^2$	82.58	415.24	3019.89	CPU: 0.079 GPU: 0.041	CPU: 3381 GPU: 145
U-ECCT (Type-III)	$(M_{\text{L1}}+M_{\text{L2}}+M_{\text{L3}})$ $\times(\frac{11}{4}(2n-k)^2 d_{\text{model}}$ $+\frac{49}{2}(2n-k)d_{\text{model}}^2)$ $+\frac{35}{4}(2n-k)d_{\text{model}}^2$	68.81	331.87	2233.47	CPU: 0.113 GPU: 0.045	CPU: 5812 GPU: 298
VU-ECCT	$\frac{M}{2}(2n-k)^2 d_{\text{model}}$ $+\frac{8M}{2}(2n-k)d_{\text{model}}^2$	68.81	346.03	2516.58	CPU: 0.128 GPU: 0.039	CPU: 5621 GPU: 272

performance of ECCT on moderate code-length decoding. The basic U-ECCT design forms a unique interaction between the shortcut connections and the main network flow, supporting an enhanced extraction of reliable information beyond just relying on the self-attention mechanism. This enables the model to achieve both performance enhancement and model size reduction. With the proposed mirror-sharing strategy, parameter utilization efficiency has significantly increased. This method also enables a deeper yet lighter model, thus achieving better performance and size reduction compared to the baseline ECCT. With the modification in the form of VU-ECCT, the decoding performance at low code rates is significantly enhanced, providing up to 0.7dB in average performance gains on the scale of E_b/N_0.

REFERENCES

[1] A. Buchberger, C. Häger, H. D. Pfister, L. Schmalen, and A. Graell i Amat, "Pruning and quantizing neural belief propagation decoders," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1957-1966, July 2021.

[2] G. Larue, L.-A. Dufrene, Q. Lampin, H. Ghauch, and G. R.-B. Othman, "Neural belief propagation auto-encoder for linear block code design," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7250-7264, Nov. 2022.

[3] E. Nachmani and L. Wolf, "Autoregressive belief propagation for decoding block codes," 2021, *arXiv preprint arXiv:2103.11780*.

[4] D. Liu, M. Bober, and J. Kittler, "Neural belief propagation for scene graph generation," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 45, no. 8, pp. 10161-10172, Aug. 2023.

[5] I. Be'Ery, N. Raviv, T. Raviv, and Y. Be'Ery, "Active deep decoding of linear codes," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 728-736, Feb. 2020.

[6] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 144-159, Feb. 2018.

[7] Q. Wang *et al.*, "Normalized min-sum neural network for LDPC decoding," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 1, pp. 70-81, Feb. 2023.

[8] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cog. Commun. Netw.*, vol. 3, no. 4, pp. 563-575, Dec. 2017.

[9] Y. Jiang *et al.*, "Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 2758-2768.

[10] Y. Jiang *et al.*, "Feedback turbo autoencoder," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2020, pp. 8559-8563.

[11] J. Clausius *et al.*, "Serial vs. parallel turbo-autoencoders and accelerated training for learned channel codes," in *Proc. 11th Inter. Symp. Topics Coding (ISTC)*, 2021, pp. 1-5.

[12] F. A. Aoudia and J. Hoydis, "Model-free training of end-to-end communication systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2503-2516, Nov. 2019.

[13] D. García *et al.*, "Model-free machine learning of wireless SISO/MIMO communications," *Comp. Commun.*, vol. 181, pp. 192-202, Jan 2022.

[14] T. Gruber, S. Cammerer, J. Hoydis, and S. t. Brink, "On deep learning-based channel decoding," in *Proc. 51st Annual Conf. Inf. Sciences Syst. (CISS)*, 2017, pp. 1-6.

[15] Y. Choukroun and L. Wolf, "Error correction code transformer," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp.38695-38705.

[16] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol.30, pp. 5998-6008, 2017.

[17] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep learning for decoding

of linear codes - a syndrome-based approach," in *Proc. 2018 IEEE Inter. Symp. Info. Theory (ISIT)*, Vail, CO, USA, 2018, pp. 1595-1599.
[18] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, pp. 599–618, Feb. 2001.
[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. Comput. Vision Pattern Recognit.*, pp. 770-778, 2016.
[20] F. He, T. Liu, and D. Tao, "Why resnet works? Residuals generalize," *IEEE Trans. Neural Netw. Learn. Syst.*, no. 12, vol. 31, pp. 5349-5362, 2020.
[21] W. Tong, W.Chen, W. Han, X. Li, and L. Wang, "Channel-attention-based DenseNet network for remote sensing image scene classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens."*, vol. 13, pp. 4121-4132, 2020.
[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, pp. 4700-4708, 2017.
[23] S. Qian, C. Ning, and Y. Hu, "MobileNetV3 for image classification." in *Proc. IEEE 2nd Int. Conf. Big Data, Artif Intell. Internet of Things Eng. (ICBAIE).*, 2021.
[24] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. 18th Int. Conf Med. Image Comput. Comput.-Assist. Interv.*, Oct. 2015, pp. 234-241.
[25] J. Devlin *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv preprint arXiv:1810.04805*.
[26] X. Liu, H. Lu, J. Yuan, and X. Li, "CAT: Causal audio transformer for audio classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2023, pp. 1-5.
[27] B. Tang and D. S. Matteson, "Probabilistic transformer for time series analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 23592-23608.
[28] K. Han *et al.*, "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 45, no. 1, pp. 87-110, Jan. 2023.
[29] Y. Wang, Z. Gao, D. Zheng, S. Chen, D. Gunduz, and H. V. Poor, "Transformer-empowered 6G intelligent networks: From massive MIMO processing to semantic communication," *IEEE Wireless Commun.*, vol. 30, no. 12, pp. 127-135, Dec. 2023.
[30] E. Ozfatura, Y. Shao, A. G. Perotti, B. M. Popović, and D. Gündüz, "All you need is feedback: Communication with block attention feedback codes," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 3, pp. 587-602, Sept. 2022.
[31] Y. Shao, E. Ozfatura, A. G. Perotti, B. M. Popović, and D. Gündüz, "AttentionCode: Ultra-reliable feedback codes for short-packet communications," *IEEE Trans. Commun.*, vol. 71, no. 8, pp. 4437-4452, Aug. 2023.
[32] W. Kong, X. Jiao, Y. Xu, B. Zhang, and Q. Yang, "A transformer-based contrastive semi-supervised learning framework for automatic modulation recognition," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 4, pp. 950-962, Aug. 2023.
[33] J. Cai, F. Gan, X. Cao, and W. Liu, "Signal modulation classification based on the transformer network," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 3, pp. 1348-1357, Sept. 2022.
[34] S. Yao, K. Niu, S. Wang, and J. Dai, "Semantic coding for text transmission: An iterative design," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 4, pp. 1594-1603, Dec. 2022.
[35] C. -M. Fan, T. -J. Liu, and K. -H. Liu, "SUNet: Swin transformer UNet for image denoising," in *Proc. 2022 IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, 2022, pp. 2333-2337.
[36] D. H. Thai, X. Fei, M. T. Le, A. Züfle, and K. Wessels, "Riesz-Quincunx-UNet variational autoencoder for unsupervised satellite image denoising," *IEEE Trans. Geosci. Remote Sens.*, Art no. 5404519, vol. 61, pp. 1-19, 2023.
[37] A. Lin *et al.*, "DS-TransUNet: Dual swin transformer U-Net for medical image segmentation," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1-15, 2022.
[38] M. Perslev *et al.*, "U-time: A fully convolutional network for time series segmentation applied to sleep staging." in *Proc. Adv. Neural Inf. Process. Syst.* vol. 32, 2019, pp. 4415-4426.
[39] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[40] D. Liu and G. Liu, "A transformer-based variational autoencoder for sentence generation," in *Procs 2019 Int. Joint Conf. on Neural Netw. (IJCNN)*, 2019, pp. 1-7.
[41] V. Raj and S. Kalyani, "Design of communication systems using deep learning: A variational inference perspective," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1320-1334, Dec. 2020.
[42] M. A. Alawad, M. Q. Hamdan and K. A. Hamdi, "Innovative variational autoencoder for an end-to-end communication system," *IEEE Access*, vol. 11, pp. 86834-86847, 2023.
[43] Y. M. Saidutta, A. Abdi and F. Fekri, "Joint source-channel coding over additive noise analog channels using mixture of variational autoencoders," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2000-2013, July 2021.
[44] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv preprint arXiv:1606.08415*.
[45] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* Morgan kaufmann, 1988.
[46] M.P.C. Fossorier, M. Mihaljević, H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.* , vol.47, no.5, pp.673–680, May 1999.
[47] I. Tal and A. Vardy, "List Decoding of Polar Codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213-2226, May 2015.
[48] S. A. Hebbar *et al.*. "CRISP: Curriculum based sequential neural decoders for polar code family," in *Procs. Inter. Conf. Machine Learn.*, 2023, pp. 12823-12845.
[49] M. Helmling *et al.*, "Database of channel codes and ML simulation results", www.uni-kl.de/channel-codes, 2019.
[50] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8024–8035.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv preprint arXiv:1412.6980*.
[52] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016, *arXiv preprint arXiv:1608.03983*.