FAID Diversity via Neural Networks

Xin Xiao*, Nithin Raveendran*, Bane Vasić*, Shu Lin†, and Ravi Tandon*

* The University of Arizona, Tucson, AZ, 85721. Email: {7xinxiao7, nithin, vasic, tandonr}@email.arizona.edu

†University of California, Davis, CA, 95616. Email: shulin@ucdavis.edu

***Abstract*—*Decoder diversity* is a powerful error correction framework in which a collection of decoders collaboratively correct a set of error patterns otherwise uncorrectable by any individual decoder. In this paper, we propose a new approach to design the decoder diversity of finite alphabet iterative decoders (FAIDs) for *Low-Density Parity Check* (LDPC) codes over the binary symmetric channel (BSC), for the purpose of lowering the error floor while guaranteeing the waterfall performance. The proposed decoder diversity is achieved by training a recurrent quantized neural network (RQNN) to learn/design FAIDs. We demonstrated for the first time that a machine-learned decoder can surpass in performance a man-made decoder of the same complexity. As RQNNs can model a broad class of FAIDs, they are capable of learning an arbitrary FAID. To provide sufficient knowledge of the error floor to the RQNN, the training sets are constructed by sampling from the set of most problematic error patterns - trapping sets. In contrast to the existing methods that use the cross-entropy function as the loss function, we introduce a frame-error-rate (FER) based loss function to train the RQNN with the objective of correcting specific error patterns rather than reducing the bit error rate (BER). The examples and simulation results show that the RQNN-aided decoder diversity increases the error correction capability of LDPC codes and lowers the error floor.**

***Index Terms*—Decoder diversity, Error floor, LDPC codes, Quantized neural network**

I. Introduction

Deep neural networks (DNNs) have gained intensive popularity in communication, signal processing, and data storage communities in the past five years due to their great potential of solving problems relevant to optimization, function approximation, and others. One popular idea, known as *model-driven neural networks* (NNs), is to combine the model knowledge (or the prototype algorithms) and the NN in conjunction with optimization techniques of NNs to improve the model. In the context of iterative decoding of error correction codes, the *deep unfolding* framework [1] is particularly attractive as it naturally unfolds the decoding iterations over the Tanner graph into a deep neural network. The activation functions are defined by the prototype decoding rules that mimic the message-passing process [2]. One merit of this framework is that the weight matrices and activation functions over hidden layers are constrained to preserve the message update function symmetry conditions, thus making it possible to train the NN on a single codeword and its noisy realizations, rather than on the entire code space.

It has been shown that deep unfolding efficiently optimizes various iterative decoding algorithms such as *Belief Propagation* (BP) and improves the decoding convergence [3]–[6]. One common feature of most existing model-driven NNs is that the training sets are constructed by *randomly* sampling channel output sequences at low signal-to-noise-ratios (SNRs), when applied on the additive white Gaussian noise (AWGN) channel [3]–[5], or at high crossover probabilities in the case of the binary symmetric channel (BSC) [6]. Consequently, when considering the *Low-Density Parity Check* (LDPC) codes, this means that the model-driven NNs are optimized for the waterfall region in the curve of decoding performance, where the probability of error starts to drop drastically. However, at low crossover probabilities, low-weight error patterns occur more frequently, and are more likely to cause decoding failure. This requires very long training time to get sufficient statistics on the error floor (EF), making the NN framework impractical for applications that require very small frame error rate (FERs).

The model-driven NNs are agnostic to these problematic uncorrectable error patterns that dominate the error floor, and to make learning efficient, it is thereby crucial to sample from a much smaller set of the most "harmful" error patterns. In addition, to optimize the error floor, our NN must use a new loss function based on FER.

It is well known that the error floors of LDPC codes are caused by specific sub-graphs of the Tanner graph, known as *Trapping sets* (TS) [7], which prevent an iterative decoder from converging to a codeword. When an LDPC code is transmitted over a BSC and is decoded with a specific decoder $\mathcal{D}$, the slope of its error floor is characterized by its guaranteed error correction capability t [8], defined as the largest weight of all error patterns correctable by the decoder $\mathcal{D}$, i.e., $\mathcal{D}$ can correct all error patterns with weight up to t. Generally speaking, the guaranteed error correction capability of a *single* iterative decoder is far lower than that of maximum likelihood decoder (MLD). To increase the guaranteed error correction capability of a given LDPC code, Declercq *et al.* [9] proposed an ensemble of finite alphabet iterative decoders (FAIDs), known as *decoder diversity*, where each FAID can correct different error patterns. The FAIDs, if designed properly, are known to be capable of surpassing the floating-point BP algorithms [10]. In [9], these decoders are selected by going over all error patterns in predefined trapping sets, such that their

The work is funded in part by the NSF under grants CIF-1855879, CCF 2106189, CCSS-2027844 and CCSS-2052751. Bane Vasić has disclosed an outside interest in Codelucida to the University of Arizona. Conflicts of interest resulting from this interest are being managed by The University of Arizona in accordance with its policies.

combination can correct all the error patterns associated with the predefined trapping sets. Note that this FAID selection is essentially a brute force approach that checks all error patterns for all FAID candidates, and it makes a-priori assumptions on problematic trapping sets, which might not be "harmful" trapping sets for specific Tanner graphs [11].

In this paper, we propose a model-driven NN scheme to design the decoder diversity of FAIDs for regular LDPC codes over BSC, with the objective of performing well in both waterfall and error floor regions. Unlike the brute force approach in [9], our framework is dynamically driven by error patterns to design different FAIDs. The scheme begins with the design of an initial FAID with a good decoding threshold to guarantee waterfall performance. The rest of the FAIDs are designed via a recurrent quantized neural network (RQNN) in order to reduce the error floor. This RQNN models the universal family of FAIDs, thus it is capable of learning any arbitrary FAID. To collect sufficient knowledge of the trapping sets, we need to construct a training set consisting of the most problematic error patterns the initial FAID fails on. For this, we rely on the sub-graph expansion-contraction [11]. The advantage is that the sub-graph expansion-contraction method obtains the training set of harmful error patterns for any FAID without making any a-priori assumptions about which graph topologies are harmful. In addition, the method is computationally efficient compared to Monte Carlo simulation and accurate in comparison with other TS enumeration techniques that do not take the decoder into account. Instead of selecting the FAIDs by checking all error patterns in predefined trapping sets as in [9], we train an RQNN on different error patterns to design FAIDs in a sequential fashion. Since our goal is to correct specific error patterns rather than reducing the bit error rate (BER), we propose the frame error rate as the loss function to train the RQNN. Consequently, the learned FAIDs are optimized in the error floor region and are expected to correct different error patterns. We consider the *quasi-cyclic* (QC) Tanner code (155, 64) as an example and the numerical results show that the RQNN-aided decoder diversity increases the guaranteed error correction capability and has a lower error floor bound.

II. Preliminaries

A. Notation

We consider a binary LDPC code $\mathcal{C}$, with a parity-check matrix $\mathbf{H}$ of size $M \times N$. The associated Tanner graph is denoted by $\mathcal{G} = (V, C, E)$, with V (respectively, C) the set of N variable (respectively, M check) nodes corresponding to the N columns (respectively, M rows) in $\mathbf{H}$, and E the set of edges. Let the i-th variable node (VN) be v_i and the j-th check node (CN) be c_j. The set of check (respectively, variable) nodes adjacent to v_i (respectively, c_j) is denoted as $\mathcal{N}(v_i)$ (respectively, $\mathcal{N}(c_j)$). The degree of a node in $\mathcal{G}$ is defined as the number of its neighbors. If all the variable nodes have the same degree d_v, $\mathcal{C}$ is said to have regular column weight d_v, and if all the check nodes have the same degree d_c, $\mathcal{C}$ is said to have regular row weight d_c. In the following, we mainly consider the regular (d_v, d_c) LDPC codes. Assume that the crossover probability of BSC is α, the codeword transmitted over BSC is $\mathbf{x} = (x_1, x_2, \cdots, x_N) \in \mathcal{C}$, and the received channel output vector is $\mathbf{y} = (y_1, y_2, \cdots, y_N) \in \mathrm{GF}(2)^N$. Let $\mathbf{e} = (e_1, e_2, \cdots, e_N)$ be the *error pattern* introduced by the BSC, then $\mathbf{y} = \mathbf{x} \oplus \mathbf{e}$, where $\oplus$ is the component-wise XOR operator. The weight of an error pattern $\mathbf{e}$ denoted by $w(\mathbf{e})$ is defined as the total number of nonzero components.

A trapping set (TS) $\mathbf{T}$ [7], [10] for an iterative decoder is a non-empty set of variable nodes in $\mathcal{G}$ that are not correct at the end of a given number of iterations. Note that $\mathbf{T}$ will depend on the decoder input as well as decoder implementation.

B. FAID

We follow the definition of FAID introduced in [10]. A b-bit FAID denoted by $\mathcal{D}_{FAID}$ is defined by a 4-tuple: $\mathcal{D}_{FAID} = (\mathcal{M}, \mathcal{Y}, \Phi, \Psi)$, where $\mathcal{M}$ is the domain of the messages passed in FAID defined as $\mathcal{M} = \{0, \pm L_1, \pm L_2 ..., \pm L_s\}$, with $L_i \in \mathbb{R}^+, 1 \leq i \leq s, s \leq 2^{b-1} - 1$ and $L_i > L_j$ if $i > j$. For a message $m \in \mathcal{M}$ associated with v_i, its sign represents an estimate of the bit value of v_i, namely $v_i = 0$ if $m > 0$, $v_i = 1$ if $m < 0$, and $v_i = y_i$ if $m = 0$, and its magnitude $|m|$ measures the reliability of this estimate. $\mathcal{Y}$ is the domain of channel outputs. For BSC, $\mathcal{Y} = \{\pm \mathrm{C}\}$ with some $\mathrm{C} \in \mathbb{R}^+$ as we use the bipolar mapping: $0 \to \mathrm{C}$ and $1 \to -\mathrm{C}$. Let $\mathbf{z} = (z_1, z_2, ...z_N)$ be the input vector to a FAID, with $z_i = (-1)^{y_i}\mathrm{C}$, $1 \leq i \leq N$. The functions Φ and Ψ describe the message update rules of variable nodes and check nodes, respectively. For a check node c_j with degree d_c, its updating rule is given by

$$\Psi(\mathbf{m}_j) = \prod_{m \in \mathbf{m}_j} \mathrm{sgn}(m) \cdot \min_{m \in \mathbf{m}_j} (|m|), \tag{1}$$

where $\mathrm{sgn}(\cdot)$ is the sign function and $\mathbf{m}_j$ is the set of extrinsic incoming messages to c_j, with $|\mathbf{m}_j| = d_c - 1$ and $\mathbf{m}_j \in \mathcal{M}^{d_c - 1}$. For a variable node v_i with degree d_v, its updating rule is given by

$$\Phi(z_i, \mathbf{n}_i) = Q\left(\sum_{m \in \mathbf{n}_i} m + \omega_i z_i\right), \tag{2}$$

where $\mathbf{n}_i$ is the set of extrinsic incoming messages to v_i, with $|\mathbf{n}_i| = d_v - 1$ and $\mathbf{n}_i \in \mathcal{M}^{d_v - 1}$. The function $Q(\cdot)$ is the quantizer defined by $\mathcal{M}$ and a threshold set $\mathcal{T} = \{T_1, ..., T_s, T_{s+1} = \infty\}$, with $T_i \in \mathbb{R}^+, 1 \leq i \leq s$ and $T_i > T_j$ for any $i > j$:

$$Q(x) = \begin{cases} \mathrm{sgn}(x) L_i & \text{if} \quad T_i \leq |x| < T_{i+1} \\ 0 & \text{if} \quad |x| < T_1 \end{cases}. \tag{3}$$

The coefficient ω_i is a real number. If ω_i is a constant for all possible $\mathbf{n}_i$, Φ is the quantization of a linear function, and its associated FAID is called *linear* FAID, otherwise, its associated FAID is called *nonlinear* FAID. At the end of each iteration, the estimate of bit associated with each variable node v_i is made by the sign of the sum of all incoming messages and channel value z_i, i.e., zero if the sum is positive, one if the sum is negative, and y_i if the sum is zero. This sum represents the estimate of bit-likelihoods and we denote it by Υ.

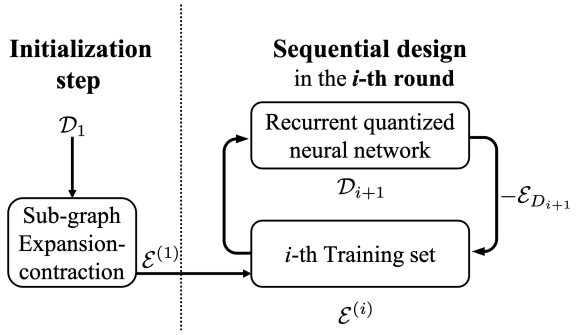

Fig. 1. Block diagram of FAID diversity via a recurrent quantized neural network. The design process consists of two steps, the Initialization step shown on the left, and the Sequential design step shown on the right. In the Initialization, the sub-graph expansion-contraction is applied for $\mathcal{D}_1$ to find the set of the most problematic error patterns $\mathcal{E}^{(1)}$, and $\mathcal{E}^{(1)}$ is set to be the first training set. In the i-th round of Sequential design, the RQNN is trained with the training set $\mathcal{E}^{(i)}$ to design the FAID $\mathcal{D}_{i+1}$. Once the training is completed, $\mathcal{E}_{\mathcal{D}_{i+1}}$, the subset of $\mathcal{E}^{(i)}$ correctable by $\mathcal{D}_{i+1}$, is excluded from $\mathcal{E}^{(i)}$.

C. Decoder diversity

In this work, we follow the definition of decoder diversity in [9]. The b-bit decoder diversity $\mathcal{D}$ is a set consisting of $N_{\mathcal{D}}$ b-bit FAIDs, which can be defined as

$$\mathcal{D} = \{\mathcal{D}_i | i = 1, \cdots, N_{\mathcal{D}}\}, \tag{4}$$

where each b-bit FAID is given by $\mathcal{D}_i = (\mathcal{M}, \mathcal{Y}, \Phi_i, \Psi)$, with Φ_i the VN updating rule of $\mathcal{D}_i$. Given an LDPC code $\mathcal{C}$ and a set $\mathcal{E}_t$ consisting of all error patterns with weight no greater than t, the objective is to design $\mathcal{D}$ with the smallest cardinality $N_{\mathcal{D}}$ such that the $N_{\mathcal{D}}$ FAIDs can collectively correct all the error patterns in $\mathcal{E}_t$. The selected FAIDs can be used in either a sequential or a parallel fashion, depending on the memory and throughput constraints. In next section, we propose a greedy framework to design $\mathcal{D}$ via an RQNN, which might not have the smallest cardinality $N_{\mathcal{D}}$ but still be capable of correcting a great number of error patterns in $\mathcal{E}_t$.

III. Decoder Diversity by RQNNs

For simplicity, we call the decoder diversity of FAIDs as *FAID diversity*.

A. A sequential framework

Basically, the proposed design of $\mathcal{D}$ consists of two main steps, namely, the Initialization and Sequential design, as shown in Fig. 1. In the Initialization step, we begin with the first FAID $\mathcal{D}_1$ that has a good decoding threshold for the purpose of good performance in waterfall region. $\mathcal{D}_1$ can be designed by various methods, such as the Density Evolution [12] and quantized neural networks [5], [6]. We then use the sub-graph expansion-contraction [11] (as will be introduced in Section III-B) to determine the most problematic error patterns that cannot be corrected by $\mathcal{D}_1$. Denote the set consisting of most problematic error patterns as $\mathcal{E}^{(1)}$ and the guaranteed correction capability of $\mathcal{D}_1$ as t, then for any error pattern $\mathbf{e} \in \mathcal{E}^{(1)}$, $w(\mathbf{e}) = t + 1$. In particular, $\mathcal{E}^{(1)}$ will be used as the initial training set in the next step. In the sequential design, we use a recurrent quantized neural network to construct the rest of the FAIDs with the objective of correcting as many error patterns in $\mathcal{E}^{(1)}$ as possible. We train the RQNN in multiple rounds, with one FAID per round. In each round, the RQNN is trained over a training set consisting of all the error patterns that cannot be corrected by any FAID in the most recently updated set $\mathcal{D}$. Once the offline training of the RQNN is completed, the learned FAID corresponding to the current RQNN is added to $\mathcal{D}$, and the error patterns that can be corrected by this FAID are excluded from the training set in the current round.

To be more specific, consider the i-th round as shown in Fig. 1, where $i \geq 1$. Let the training set used in the i-th round be $\mathcal{E}^{(i)}$, the FAID to be learned in the i-th round be $\mathcal{D}_{i+1}$, and the subset of $\mathcal{E}^{(i)}$ corrected by the learned FAID be $\mathcal{E}_{\mathcal{D}_{i+1}}$. Then, for any error pattern $\mathbf{e} \in \mathcal{E}^{(i)}$, it cannot be corrected by any of $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_i$. The RQNN is trained with $\mathcal{E}^{(i)}$ to minimize the training set error so that the cardinality of $\mathcal{E}_{\mathcal{D}_{i+1}}$ is maximum. Subsequently, $\mathcal{D}_{i+1}$ is added to $\mathcal{D}$ and the training set $\mathcal{E}^{(i+1)}$ used in the next round is derived by

$$\mathcal{E}^{(i+1)} = \mathcal{E}^{(i)} \setminus \mathcal{E}_{\mathcal{D}_{i+1}}. \tag{5}$$

This process continues until $\mathcal{E}^{(i)}$ becomes an empty set or a predefined maximum number of rounds $N_{\mathcal{D}} - 1$ has been reached. If $\mathcal{E}^{(i)}$ eventually becomes empty, the designed FAID diversity can correct all error patterns with weight up to $t + 1$ in $\mathcal{E}^{(1)}$. Otherwise, the process have completed $N_{\mathcal{D}} - 1$ rounds and the last training set $\mathcal{E}^{(N_{\mathcal{D}})}$ is a nonempty set. As a result, the designed FAID diversity can correct most of the error patterns with weight up to $t + 1$, and only $\left|\mathcal{E}^{(N_{\mathcal{D}})}\right|$ uncorrectable error patterns with weight $t + 1$.

Because of the sequential training strategy, the FAID diversity $\mathcal{D}$ uses its decoders in the same order as how its decoders are determined. Specifically, assume that the predefined maximum number of iterations of $\mathcal{D}_i$ is I_i. To decode an input vector $\mathbf{z}$, $\mathcal{D}_1$ with I_1 iterations is first applied. If $\mathcal{D}_1$ decodes $\mathbf{z}$ successfully, the decoding process terminates, otherwise it is re-initialized with $\mathbf{z}$ and switches to $\mathcal{D}_2$ with I_2 iterations. The decoding process continues decoding $\mathbf{z}$ with FAIDs in order until a FAID $\mathcal{D}_j$ corrects $\mathbf{z}$. Otherwise, all FAIDs in $\mathcal{D}$ fail on $\mathbf{z}$, in this case, the decoding process claims a decoding failure.

B. Subgraph expansion-contraction

The expansion-contraction method introduced in [11] estimates the error floor of an arbitrary iterative decoder operating on a given Tanner graph of an LDPC code in a computationally efficient way by identifying the minimal-weight uncorrectable error patterns and harmful TSs for the code and decoder. In the expansion step of the method, a list of short cycles (of length g and $g + 2$, where g is the girth of the Tanner graph) present in the Tanner graph $\mathcal{G}$ of the code is expanded to each of their sufficiently large neighborhood in $\mathcal{G}$. The expansion step for a given LDPC code only needs to be executed once and its output: $\mathcal{L}_{\text{EXP}}$, the list of expanded sub-graphs can be used for different decoders for the next step of contraction.

Each of the expanded sub-graphs in $\mathcal{L}_{\text{EXP}}$ are now *contracted* in order to identify failure inducing sets for a given

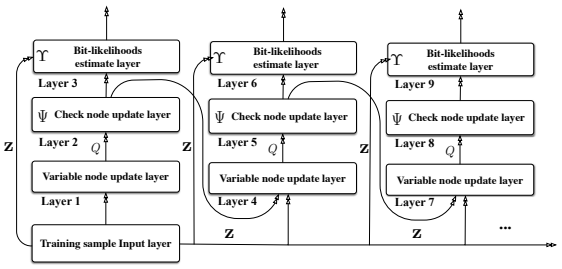

Fig. 2. Block diagram of a RQNN. Each column corresponds to one iteration, where variable nodes are first updated, followed by the quantization function Q. The quantized messages are then used to update check nodes. The output of Ψ and Input layer are fed into both Υ in current iteration and the variable node update layer in next iteration.

decoder $\mathcal{D}$. This is achieved by exhaustively decoding error patterns running $\mathcal{D}$ on these sub-graphs (not on the entire Tanner graph $\mathcal{G}$) ensuring that the messages accurately represent the actual messages operating on G. This process lists all minimum-weight failure inducing error patterns which will then be used as the training set for the RQNN.

C. A model-driven structure for general FAID

In the Sequential design, we rely on a recurrent quantized neural network to design $\mathcal{D}_i$, $i \geq 2$. The proposed RQNN is a model-driven recurrent deep neural network, which is constructed by unfolding the general FAID with a given number of iterations, as shown in [6]. The connection between consecutive layers is determined by the Tanner graph $\mathcal{G}$, and activation functions over hidden layers are defined based on Φ, Ψ, and Υ. As it has recurrent structure, the trainable parameters are shared among all the iterations. As shown in Fig. 2, three consecutive hidden layers in one column correspond to one iteration in FAID, with one layer for VN message update, one layer for CN message update, and one layer for bit-likelihood estimation. In particular, rather than introducing weights matrices between layers as in most existing model-driven NN frameworks, the RQNN has only the coefficients ω in (2) as trainable parameters. Since Φ is a symmetric function mapping from $\mathcal{Y}\times\mathcal{M}^{d_v-1}$ to $\mathcal{M}$, to design a FAID for a given quantizer $Q(\cdot)$, we only need to determine $|\mathcal{M}|^{d_v-1}$ values of ω for the case where the input value is $-\mathrm{C}$. We use $\omega[i_1,\cdots,i_{d_v-1}]$ to indicate the coefficient ω to be learned for the case that the extrinsic incoming message is $(i_1,\cdots,i_{d_v-1})$, where $i_k \in \mathcal{M}, k = 1,\cdots,d_v-1$. Moreover, we are interested in the Φ that is invariant to the ordering of the extrinsic incoming messages. This means that $\Phi(z_i,\mathbf{n}_i{}') = \Phi(z_i,\mathbf{n}_i)$, where $\mathbf{n}_i{}'$ is an arbitrary permutation of $\mathbf{n}_i$. This rotation symmetry of Φ limits the RQNN to learn the coefficients with non-decreasing arguments, namely, $\omega[i_1,\cdots,i_{d_v-1}]$ where $i_j \leq i_k, \forall j < k$, and reduces the number of trainable coefficients from $|\mathcal{M}|^{dv-1}$ to $\binom{|\mathcal{M}|+d_v-2}{d_v-1}$. We call the FAID whose Φ has rotation symmetry property as *symmetric* FAID. We denote the set of these trainable coefficients by Ω, i.e., $\Omega \triangleq \{\omega[i_1,\cdots,i_{d_v-1}]|i_k \in \mathcal{M}, i_1 \leq \cdots \leq i_{d_v-1}\}$. In [6], we proposed using RQNNs to design linear FAIDs. This approach, however, has limitations when an optimal FAID is nonlinear whose VN updating rule satisfies specific constraints. In [10], Lemma 1 provided an example of possible constraints and showed that any FAID whose Φ satisfying these constraints cannot be expressed as a quantization of a linear function. This means that the approach [6] cannot learn the optimal FAID if the optimal FAID has Φ satisfying the constraints in Lemma 1 [10]. The main reason of these limitations lies in its prototype VN updating rule, which assumes that the coefficient ω is a constant. On the contrary, the prototype VN updating rule in our proposed RQNN can express any arbitrary symmetric FAID, as shown in the following proposition.

Proposition 1: Given an arbitrary symmetric FAID, whose VN updating mapping is determined by $\binom{|\mathcal{M}|+d_v-2}{d_v-1}$ values $\{\mu[i_1,\cdots,i_{d_v-1}]|i_k \in \mathcal{M}, i_1 \leq \cdots \leq i_{d_v-1}\}$, where $\mu[i_1,\cdots,i_{d_v-1}] \in \mathcal{M}$ is the value for the case that the received input is $-\mathrm{C}$ and the extrinsic incoming message is $(i_1,\cdots,i_{d_v-1})$. Then, there exist $\binom{|\mathcal{M}|+d_v-2}{d_v-1}$ coefficients such that for any $i_k \in \mathcal{M}$, $i_1 \leq \cdots \leq i_{d_v-1}$, $\Phi(-\mathrm{C}, i_1,\cdots,i_{d_v-1}) = \mu[i_1,\cdots,i_{d_v-1}]$.
Noticed that these $\binom{|\mathcal{M}|+d_v-2}{d_v-1}$ coefficients are defined independently, the proof is straightforward as we can determine each coefficient individually by replacing each μ and $(i_1,\cdots,i_{d_v-1})$ in (2). Proposition 1 indicates that our RQNN framework, if properly initialized, can learn arbitrary FAID.

D. Training RQNN

The coefficients Ω in the RQNN can be initialized by conventional iterative decoders or the decoders constructed by specific techniques such as Density Evolution and the selection approach in [10]. Specifically, we first derive the LUT of some well-known decoder like quantized Offset MS decoder, and take its coefficients as the initialization of the RQNN. Since the RQNN preserves the symmetry conditions, we can simply use the all-zero codeword and its noisy realizations to construct a training set. In particular, to design $\mathcal{D}_{i+1}$ in the i-th round, the RQNN is trained with $\mathcal{E}^{(i)}$. Let $\mathbf{u}$ be the values in the output layer. Since the objective of the RQNN is to correct the error patterns in $\mathcal{E}^{(i)}$ as many as possible, and we assume $\mathbf{x} = \mathbf{0}$, we propose the following FER as the loss function

$$\Gamma(\mathbf{u}) = \frac{1}{2}\left[1 - \mathrm{sgn}\left(\min_{1\leq i\leq N} u_i\right)\right]. \tag{6}$$

Note that u_i represents the estimate likelihood of the i-th bit. Since $\mathbf{x} = \mathbf{0}$, each component in $\mathbf{u}$ is expected to be positive. Therefore, a frame is decoded in error if and only if the minimum component is negative. The quantizer $Q(\cdot)$ and the sign function in (6) cause a critical issue that their gradients vanish almost everywhere, making it difficult to use classical backward propagation. Similar to [6], we leverage straight-through estimators (STEs) as surrogate gradients to tackle this issue. In particular, we use the same STEs as [6]. Motivated by the fact that applying different learning rates to Ω can help training convergence, we employ ADAM [13] with mini-batches with gradients accumulated in full precision.

IV. A Case study and numerical results

In this section, we consider the Tanner code (155, 64) as an example to demonstrate how to use an RQNN to design a

TABLE I
STATISTICS ON THE ERROR CORRECTION OF FAID DIVERSITIES

Error patterns	FAID Diversity [9]	RQNN-aided FAID Diversity
$\mathcal{E}^{(1)}$	930	0
$\mathcal{E}'$	507966	480655

3-bit FAID diversity. The Tanner code has column weight of 3 and row weight of 5, with a minimum distance of 20. The $\mathcal{M}$ and $\mathcal{T}$ are predefined to be $\mathcal{M} = \{0, \pm 1, \pm 2, \pm 3\}$ and $\mathcal{T} = \{\pm 0.5, \pm 1.5, \pm 2.5\}$, respectively. We first select the 3-bit nonlinear FAID D_0 in [10][TABLE II] as our $\mathcal{D}_1$. As shown in [10], $\mathcal{D}_1$ is guaranteed to correct all error patterns with weight up to 5. We apply the sub-graph expansion-contraction method for $\mathcal{D}_1$ with 100 iterations, and obtain $\mathcal{E}^{(1)}$ consisting of 29294 error patterns with weight of 6. We construct an RQNN with 50 iterations. The size of each mini-batch is set to 20, and the learning rate is set to 0.001. We sequentially train the RQNN in 6 rounds, with the initialization shown in the longer version of this paper [14][Table I]. The training of RQNN in 6 rounds converged within 10, 10, 20, 60, 60, 30 epochs, respectively. The trained coefficients are provided in [14][Table II], from which we derive 6 RQNN-aided FAIDs. Consequently, our RQNN Diversity consists of 7 3-bit FAIDs. Set the maximum number of iterations of these 7 FAIDs to $100, 90, 50, 40, 50, 30, 30$ accordingly. For comparison, we consider the FAID Diversity in [9] consisting of 9 3-bit FAIDs, with each FAID performing 50 iterations. Note that the FAID Diversity in [9] starts from $\mathcal{D}_1$ as well. Table I summarizes the statistics on the error correction of the FAID Diversity derived by RQNN and the FAID Diversity in [9]. The set $\mathcal{E}'$ consists of all 1147496 error patterns with weight of 7 uncorrectable by $\mathcal{D}_1$, which is obtained by applying the sub-graph expansion-contraction method for $\mathcal{D}_1$ with 100 iterations. The numbers in Table I indicate how many error patterns uncorrectable by the corresponding FAID diversity. As shown in Table I, the FAID Diversity derived via RQNNs has less uncorrectable error patterns than the FAID Diversity in [9]. In particular, the error correction capability of the FAID Diversity in [9] is 6, while the error correction capability of the FAID Diversity via RQNNs is 7. Fig. 3 shows the error floor estimation of the FER performance of single FAID $\mathcal{D}_1$, FAID Diversity in [9], and the FAID Diversity via RQNNs. The bounds of error floor is estimated by the statistics in Table I. As shown in Fig. 3, the FAID Diversity via RQNNs has the lowest error floor bound compared to the others.

V. CONCLUSION

In this paper, we proposed a new approach to automatically design FAID diversity for LDPC codes over BSC via RQNNs. The RQNN framework models the universal family of FAIDs, thus it can learn arbitrary FAID. We applied sub-graph expansion-contraction to sample the most problematic error patterns for constructing the training set, and provided a FER loss function to train the RQNN. We designed the FAID diversity of the Tanner code (155, 64) via the proposed framework. The numerical results showed that RQNN-aided FAID diversity increases the error correction capability and has a low error floor bound.

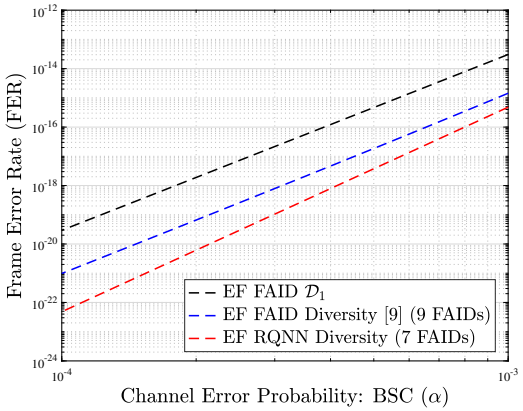

Fig. 3. Error floor estimation of $\mathcal{D}_1$, FAID Diversity [9], and RQNN-aided FAID Diversity. FAID Diversity [9] has 9 FAIDs with 450 iterations, while the RQNN-aided FAID Diversity has only 7 FAIDs with 390 iterations.

REFERENCES

[1] J. R. Hershey, R. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," *arXiv preprint arXiv:1409.2574*, 2014.

[2] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2019, pp. 266–271.

[3] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *54th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Oct. 2016, pp. 341–346.

[4] B. Vasić, X. Xiao, and S. Lin, "Learning to decode LDPC codes with finite-alphabet message passing," in *Information Theory and Applications Workshop (ITA 2018)*, San Diego, CA, Feb. 2018, pp. 1–10.

[5] X. Xiao, B. Vasić, R. Tandon, and S. Lin, "Finite alphabet iterative decoding of LDPC codes with coarsely quantized neural networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[6] X. Xiao, B. Vasić, R. Tandon, and S. Lin, "Designing finite alphabet iterative decoders of LDPC codes via recurrent quantized neural networks," *IEEE Trans. Commun.,*, vol. 68, no. 7, pp. 3963–3974, 2020.

[7] T. Richardson, "Error floors of LDPC codes," in *Proceedings of the annual Allerton conference on communication control and computing*, vol. 41, no. 3. The University; 1998, 2003, pp. 1426–1435.

[8] M. Ivkovic, S. K. Chilappagari, and B. Vasić, "Eliminating trapping sets in low-density parity-check codes by using tanner graph covers," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3763–3768, 2008.

[9] D. Declercq, B. Vasić, S. K. Planjery, and E. Li, "Finite alphabet iterative decoders—part II: Towards guaranteed error correction of LDPC codes via iterative decoder diversity," *IEEE Trans. Commun.,*, vol. 61, no. 10, pp. 4046–4057, 2013.

[10] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasić, "Finite alphabet iterative decoders—part I: Decoding beyond belief propagation on the binary symmetric channel," *IEEE Trans. Commun.,*, vol. 61, no. 10, pp. 4033–4045, 2013.

[11] N. Raveendran, D. Declercq, and B. Vasić, "A sub-graph expansion-contraction method for error floor computation," *IEEE Trans. Commun.,*, vol. 68, no. 7, pp. 3984–3995, 2020.

[12] T. T. Nguyen-Ly, V. Savin, K. Le, D. Declercq, F. Ghaffari, and O. Boncalo, "Analysis and design of cost-effective, high-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 26, no. 3, pp. 508–521, Mar. 2018.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] X. Xiao, N. Raveendran, B. Vasić, S. Lin, and R. Tandon, "FAID diversity via neural networks," *arXiv preprint arXiv:2105.04118*, 2021. [Online]. Available: https://arxiv.org/abs/2105.04118