Neural-net Decoding of Quantum LDPC Codes with Straight-through estimators

Xin Xiao, Nithin Raveendran, and Bane Vasić
University of Arizona
Email: {7xinxiao7,nithin,vasic}@email.arizona.edu

***Abstract*—Quantum low-density parity check (QLDPC) codes with asymptotically nonzero rate are promising candidates for fault-tolerant quantum computation. Belief propagation based iterative decoding algorithms, a primary choice for classical LDPC codes, perform poorly for QLDPC codes due to numerous cycles in the associated Tanner graphs. Belief propagation algorithm and its variants were also found to be inadequate in dealing with a unique quantum feature called error degeneracy. Neural network based iterative decoders are promising to address these limitations. In this paper, we propose a general framework for error correction in QLDPC codes using neural networks (NN). The neural network performs the syndrome matching algorithm over a depolarizing channel with noiseless error syndrome measurements. We train our NN to minimize the bit error rate, which is an accurate metric to measure the performance of iterative decoders. Our NN uses straight through estimator (STE) technique to tackle the zero-gradient problem of the objective function and outperforms conventional min-sum algorithm upto an order of magnitude of logical error rate.**

I. Introduction

Deep Neural Network (DNN) is an actively pursued approach in both classical and quantum applications as it has been shown to handle well the scenarios of unknown and nonlinear channels (see [1]–[3] and references therein). In the known channel case, the efforts have mainly focused on learning decoding algorithms for error-correction codes (ECC). Recently, several research groups have shown that DNNs can be used to efficiently learn various decoding algorithms, including *Belief Propagation* (BP) decoding [4]–[9]. This trend is not just limited to the classical decoding scenario. In the quest towards achieving fault tolerant quantum computation, topological codes and their decoding algorithms also employ DNNs [8], [10], [11] to improve decoding thresholds.

Quantum low-density parity check (QLDPC) codes are a promising candidate for both quantum computing and quantum optical communications, with a history of success in classical LDPC codes in admitting low-complexity decoding and near-capacity performance. As pointed out by Gottesman [12] and Kovalev and Pryadko [13], QLDPC codes are the only known class of quantum error correction (QEC) codes that permit fault-tolerant error correction with asymptotically nonzero rate. QLDPC codes [14] based on the stabilizer formalism [15] rely on classical decoding algorithms with the syndrome measurements. Ref. [16] gives a historical account of progress of these decoding algorithms. Decoding QLDPC codes is a more challenging problem than decoding topological codes. Due to the topology of Tanner graphs of finite-length QLDPC codes and the symplectic inner product/commutativity constraint [14] among the stabilizer generators, the application of traditional BP for QEC codes in general, and for QLDPC codes in particular has some fundamental limitations. While the syndrome-based BP algorithm [16] attempts to find the most likely error, an optimal decoding algorithm should find the most likely error coset based on the sum of the probabilities of all degenerate errors. Poulin and Chung [17] investigated heuristic methods to break the symmetric input channel values to improve decoding performance. "Random perturbation" method is shown to work well in such scenarios [17], [18]. This also gives intuition that the decoder message update rule handles the above mentioned limitations better when it varies over iterations. In a recent work, [19], Poulin's group used DNN-based decoder with a different loss function to tackle error degeneracy issue in QLDPC decoding scenario.

One key property of DNNs ([4]–[9]) is that the weight matrices and activation functions over hidden layers are constrained to preserve the symmetry conditions of message-passing update rules. This allows the training to be performed on a single codeword and its noise realizations rather than on the entire code space, thus opening up the possibility of using DNN on long codes. In our preliminary work [20], we have shown that DNNs can improve the decoding convergence speed (number of iterations to achieve a desired frame error rate) of conventional *Min-Sum* (MS) decoding algorithm. Unlike well established theories on *Density Evolution* (DE) [21] and *Trapping Set* (TS) [22], which are used to guide decoder design for the waterfall and error-floor regions, respectively, there is no existing theory for speeding up decoding convergence. The multi-layer structure of DNNs can naturally learn time-varying update rules that can provably outperform fixed update rules, thereby, opening new decoding design possibilities.

In this paper, we propose to use neural networks (NNs) to optimize syndrome-based iterative decoders for quantum LDPC codes over a depolarizing channel. By assigning and training weights and biases over edges in Tanner graph appropriately, the NN has capability to compensate short cycles and reduce degenerate errors. We use bit error rate (BER) as the objective function, since it is a more common and accurate metric to measure the performance of iterative decoders. However, the BER causes a critical issue that its gradient vanishes almost everywhere, making backward propagation

inapplicable. To solve this challenge, we rely on very recent results on training quantized NNs which is of great interest on its own [23]–[26]. We focus on a technique named *straight-through estimators* (STE), for the reason that it handles the zero derivatives in the backward propagation. The simulation results show that the NN-based syndrome iterative decoding achieves lower logical error rate of one order of magnitude.

The rest of the paper is organized as follows. Section II gives the necessary background of Quantum LDPC codes and NNs. Section III presents the NN framework for syndrome decoding followed by the training in Section IV. Section V demonstrates the simulation results. We conclude with future directions in Section VI.

II. Preliminaries

A. Quantum LDPC codes using Stabilizer Formalism

Let us denote by $\mathcal{P}_n = i^l\{I, X, Y, Z\}^{\otimes n}$, $0 \leq l \leq 3$, the n-qubit Pauli group, where $\otimes n$ is the n-fold tensor product, X, Y and Z are the Pauli matrices, I is the 2×2 identity matrix, and i^l is the phase factor. Let $S =< S_1, S_2, \ldots, S_m >$, $-1 \notin S$, be an Abelian subgroup of $\mathcal{P}_n$ with generators S_i, $1 \leq i \leq m$. A (n, k) quantum stabilizer code [27] is defined as a 2^k-dimensional subspace $\mathcal{C}$ of the Hilbert space $(\mathbb{C}^2)^{\otimes n}$ that is a common $+1$ eigenspace of S:

$$\mathcal{C} = \{|\psi\rangle, \quad \text{s.t. } S_i |\psi\rangle = |\psi\rangle, \forall i\}. \tag{1}$$

Every element of stabilizer generators of S is mapped to a binary tuple as follows: $\mathrm{I} \rightarrow (0,0)$, $\mathrm{X} \rightarrow (1,0)$, $\mathrm{Z} \rightarrow (0,1)$, $\mathrm{Y} \rightarrow (1,1)$. This mapping leads to the binary representation $\mathbf{H}$ of the stabilizer matrix of dimension $m \times 2n$ given by

$$\mathbf{H} = \left[H_{\mathrm{X}} \mid H_{\mathrm{Z}}\right], \tag{2}$$

where H_{X} and H_{Z} represent binary matrices for bit flip and phase flip operators, respectively. Similar to the commutativity relation defined for stabilizer generators in Pauli representation, the generators commute with respect to the *symplectic inner product* in binary representation [28].

We focus on a widely studied model called quantum depolarizing channel (memoryless Pauli channel), characterized by the depolarizing probability p wherein the error on a qubit is independent of the error on other qubits. An error on a qubit is called a Pauli error, denoted by E, which is in the set $\{I, X, Y, Z\}$. In particular, $\Pr(E = X) = \Pr(E = Y) = \Pr(E = Z) = p/3, \Pr(E = I) = 1 - p$. A Pauli error vector on the n qubits can be expressed as a binary error vector of length $2n$ with the mapping from Pauli elements to binary tuples as follows: $\mathrm{I} \rightarrow (0,0), \mathrm{X} \rightarrow (1,0), \mathrm{Z} \rightarrow (0,1), \mathrm{Y} \rightarrow (1,1)$. An error vector $\mathbf{e}$ gives the syndrome measurement as $\mathbf{s} = \mathbf{H}\mathbf{e}^{\mathrm{T}}$.

For a (n, k) quantum LDPC code $\mathcal{C}$, let $\mathbf{H}$ be the parity check matrix corresponding to its stabilizer generators, and $\mathcal{G} = (V, C, F)$ be its Tanner graph, where V (respectively, C) is the set of variable (respectively, check) nodes, and F is the set of edges. $\mathbf{H}$ has the form of Eq. (2), thus if the number of columns in $\mathbf{H}$ is N, the number of parity check equations is m, and the number of edges in $\mathcal{G}$ is I, then $|V| = N = 2n$, $|C| = m$, and $|F| = I$. Denote the i-th variable node as v_i, j-th check node as c_j, and the edge connecting v_i and c_j as (v_i, c_j) which is indexed by some integer (t), $1 \leq i \leq N, 1 \leq j \leq m, 1 \leq t \leq I$. The syndrome is denoted by $\mathbf{s} = (s_1, s_2, ..., s_m)$. Suppose that the message update rules of variable nodes and check nodes which the NN aims to optimize are defined as $\nu_{v_i \rightarrow c_j} = \Phi(\lambda_i, \mathbf{p}_i)$ and $\mu_{c_j \rightarrow v_i} = \Psi(s_j, \mathbf{q}_j)$, respectively, where λ_i is the likelihood message of v_i, and $\mathbf{p}_i$ ($\mathbf{q}_j$) is the incoming message to a variable node v_i (check node c_j). $\tilde{\lambda}_i = \Upsilon(\lambda_i, \mathbf{p}_i)$ denotes the likelihood message approximation for variable node v_i, which will be optimized by NN as well.

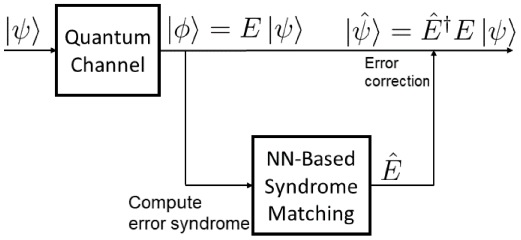

Fig. 1. Block diagram of a NN-based syndrome matching.

B. Neural networks

The NN consists of one input layer, K hidden layers and one output layer. Let $\mathbf{r}_0$ ($\mathbf{r}_{K+1}$) be the value of input (output) layer, and $\mathbf{r}_k, 1 \leq k \leq K$ be the value of the k-th hidden layer. In particular, $\mathbf{r}_k = (r_{k,1}, r_{k,2}, ...r_{k,J_k})^T$, where J_k is the number of neurons in k-th layer, and $r_{k,t}$ is the output value of the t-th neuron in k-th layer, $0 \leq k \leq K+1, 1 \leq t \leq J_k$. The $(k-1)$-th layer and k-th layer are connected by a trainable neuron weight matrix $\mathbf{W}^{(k)}_{J_k \times J_{k-1}}$, and the bias vector in the k-th layer is denoted by $\mathbf{b}^{(k)}$.

III. NN framework

A. NN-based syndrome matching

We provide the basic framework of decoding stabilizer codes as illustrated in Fig. 1, performing syndrome matching and stabilizer error recovery. Quantum state $|\psi\rangle$ decodes to erroneous quantum state $|\phi\rangle = E|\psi\rangle$. A stabilizer syndrome $\mathbf{s}$ computed by concatenating the eigenvalues of $|\phi\rangle$ for the stabilizer generators is fed into the NN based syndrome matching algorithm to find the error estimate $\hat{E}$. We retrieve the quantum state after applying the recovery operator.

B. NN structure

The proposed NN is constructed as an "unwrapped" Tanner graph, with a set of activations defined by syndrome-based iterative decoding. More specifically, there are two types of activations defined by Φ and Ψ. In general, the message update

rules Φ and Ψ can be provided by any conventional syndrome-based iterative decoder such as sum-product decoder, bit-flipping decoder, etc. In this work, we take Φ and Ψ from MS decoder. The input layer is set to be all-one vector: $\mathbf{r}_0 = \mathbf{1}$, i.e., the syndrome matching is initialized with all-zero codeword. Every two hidden layers correspond to one iteration, with odd (respectively, even) hidden layer representing variable (respectively, check) nodes message update. Each neuron in hidden layers stands for an edge. The value of the k-th hidden layer ($k > 1$) is computed as follows:

$$\mathbf{r}_k = \begin{cases} \Phi(\mathbf{b}^{(k)}\mathbf{1}, \mathbf{W}^{(k)}\mathbf{r}_{k-1}), & \text{if } k \text{ is odd,} \\ \Psi(\mathbf{s}, \mathbf{r}_{k-1}), & \text{if } k \text{ is even} \end{cases} \tag{3}$$

where $\mathbf{s}$ is the stabilizer syndrome mentioned in part A, and for any $(t) = (v_i, c_j)$, if k is odd,

$$r_{k,t} = b_t^{(k)} + \sum_{(t')=(v_i,c_{j'}),j'\neq j} w_{t,t'}^{(k)} r_{k-1,t'},$$

if k is even,

$$r_{k,t} = (-1)^{s_j} \prod_{(t')=(v_{i'},c_j),i'\neq i} \operatorname{sgn}(r_{k-1,t'}) \cdot \min_{(t')=(v_{i'},c_j),i'\neq i} |r_{k-1,t'}|$$

In particular, the first hidden layer is the initialization of syndrome matching decoding, and its value is calculated by

$$\mathbf{r}_1 = \mathbf{W}^{(1)}\mathbf{1}, \tag{4}$$

where $r_{1,t} = w_{t,i}^{(1)}, \forall (t) = (v_i, c_j)$. We force all nonzero entries in $\mathbf{W}^{(k)}$ to share the same value, i.e., $\mathbf{W}^{(k)}(i,j) = w^{(k)}$ if $\mathbf{W}^{(k)}(i,j)$ is a nonzero entry in $\mathbf{W}^{(k)}$. After each k-th layer where k is even (corresponding to check node update), for each training sample, we check whether the current syndrome of its estimated codeword matches the stabilizer syndrome $\mathbf{s}$ or not. If so, this training sample will skip the rest layers and its estimated likelihood message at the current k-th layer will be directly used to calculate the objective function, namely,

$$\mathbf{r}_{K+1} = \Upsilon(\mathbf{1}, w^{(k+1)}\mathbf{r}_k), \tag{5}$$

if $\frac{(\mathbf{1}-\operatorname{sgn}((\Upsilon(\mathbf{1},w^{(k+1)}\mathbf{r}_k))^T))}{2} \cdot \mathbf{H}^T = \mathbf{s}$. Noted that in each iteration, the same weight is used for both variable node update and likelihood message estimation. When training is completed, the NN-based syndrome iterative decoder has the following time-varying rules: $\nu_{v_i\to c_j}^{(\ell)} = \Phi(b_i^{(2\ell+1)}, w^{(2\ell+1)}\mathbf{p}_i), \mu_{c_j\to v_i}^{(\ell)} = \Psi(s_j, \mathbf{q}_j), \tilde{\lambda}_i^{(\ell)} = \Upsilon(1, w^{(2\ell+1)}\mathbf{p}_i)$.

IV. Training with NN

Since the channel is output-symmetric, and the NN's activations preserve symmetry conditions, we can assume that the all-zero codeword is transmitted, i.e., $\mathbf{x} = \mathbf{0}$. With the symmetry conditions on the weight matrices, it is sufficient to use a database composed of the noisy realizations $\mathbf{y} = (y_1, y_2, \ldots, y_N)$. The syndrome $\mathbf{s}$ of $\mathbf{y}$ is fed into the NN as parameters used in the activation over even hidden layers. The weight assigned over each edge is initialized by 1. All the biases are initialized by 1. Let $\mathbf{u} = \mathbf{r}_{K+1}$ be the value of the output layer. Then, $J_0 = J_{K+1} = N = 2n$. Denote the decoded codeword by $\hat{\mathbf{y}}$, which is computed by $\hat{\mathbf{y}} = (\mathbf{1} - \operatorname{sgn}(\mathbf{u}))/2$.

A. BER-based objective function

The output $\mathbf{u}$ consists of the estimate of likelihood messages of training samples. In most related works of using neural networks to optimize channel decoders, the binary cross entropy (BCE) function is widely applied as objective function, since it is a measurement of "soft" bit error rate, and it is differentiable everywhere. However, the BCE function is an approximation of BER. Training NNs to minimize BCE cannot guarantee to minimize BER. To see this, consider the following BCE function for each sample:

$$\Delta(\mathbf{u}, \mathbf{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(1-\sigma(u_i)) + (1-y_i)\log(\sigma(u_i)),$$

where the $\sigma(\cdot)$ is the sigmoid function defined as $\sigma(x) = (1+\mathrm{e}^{-x})^{-1}$. If the i-th bit is decoded correctly (equals to y_i) by the NN, then there exists a positive value $0 < \epsilon_i < 1/2$ such that the i-th term in the summation of $\Delta(\mathbf{u}, \mathbf{y})$ equals to $\log(\frac{1}{2} + \epsilon_i)$; otherwise there exists a positive value $0 < \epsilon_i < 1/2$ such that the i-th term in the summation of $\Delta(\mathbf{u}, \mathbf{y})$ equals to $\log(\frac{1}{2} - \epsilon_i)$. Then $\Delta(\mathbf{u}, \mathbf{y})$ can be expressed as: $\Delta(\mathbf{u}, \mathbf{y}) = \Delta_c + \Delta_e$, with $\Delta_c = \frac{1}{N}\sum_{j:y_j=\hat{y}_j} \log(1/2+\epsilon_j)^{-1}$ and $\Delta_e = \frac{1}{N}\sum_{j:y_j\neq\hat{y}_j} \log(1/2-\epsilon_j)^{-1}$. Noted that Δ_c and Δ_e are the costs contributed by the bits decoded by NN correctly and wrongly, respectively. Each term in the summation of Δ_c is within $[0, 1]$, and each term in the summation of Δ_e is larger than 1.

Consider two samples $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$, whose output values of NN, denoted by $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(2)}$, have the following conditions:

1) $\Delta(\mathbf{u}^{(1)}, \mathbf{y}^{(1)}) = \Delta_c^{(1)} + \Delta_e^{(1)}, \Delta(\mathbf{u}^{(2)}, \mathbf{y}^{(2)}) = \Delta_c^{(2)} + \Delta_e^{(2)}$;
2) There are d terms in $\Delta_e^{(1)}$ and $d-1$ terms in $\Delta_e^{(2)}$;
3) $|\Delta_c^{(1)} - \Delta_c^{(2)}| < \frac{1}{N}$; and
4) $d \max_{j:y_j^{(1)}\neq\hat{y}_j^{(1)}} \log(1/2-\epsilon_j^{(1)})^{-1} + 1 \leq (d-1) \min_{j:y_j^{(2)}\neq\hat{y}_j^{(2)}} \log(1/2-\epsilon_j^{(2)})^{-1}$.

Condition (2) implies that the first and second samples have d and $d-1$ bits in error after NN decoding, respectively, and $\Delta_c^{(1)} < 1 - \frac{d}{N}, \Delta_c^{(2)} < 1 - \frac{d-1}{N}$. Condition (3) and (4) together indicate that the first sample has smaller objective function, i.e., $\Delta(\mathbf{u}^{(1)}, \mathbf{y}^{(1)}) < \Delta(\mathbf{u}^{(2)}, \mathbf{y}^{(2)})$. In fact,

$$\begin{aligned} \Delta(\mathbf{u}^{(1)}, \mathbf{y}^{(1)}) &= \Delta_c^{(1)} + \Delta_e^{(1)} < \Delta_c^{(2)} + \frac{1}{N} + \Delta_e^{(1)} \\ &\leq \Delta_c^{(2)} + \frac{1}{N} + \frac{d}{N} \max_{j:y_j\neq\hat{y}_j^{(1)}} \log(1/2-\epsilon_j^{(1)})^{-1} \\ &\leq \Delta_c^{(2)} + \frac{d-1}{N} \min_{j:y_j\neq\hat{y}_j^{(2)}} \log(1/2-\varepsilon_j^{(2)})^{-1} \\ &\leq \Delta_c^{(2)} + \Delta_e^{(2)} = \Delta(\mathbf{u}^{(2)}, \mathbf{y}^{(2)}) \end{aligned}$$

Therefore, a smaller BCE loss cannot guarantee a smaller BER.

In this work, instead of using BCE function, we consider the following mean square error objective function for each sample, which measures the hamming distance between received channel output vector $\mathbf{y}$ and the decoded codeword $\hat{\mathbf{y}}$,

$$\Gamma(\mathbf{u},\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2, \text{ where } \hat{\mathbf{y}} = \frac{\mathbf{1}-\text{sgn}(\mathbf{u})}{2}. \quad (6)$$

$\Gamma(\mathbf{u},\mathbf{y})$ is an actual and practical metric to measure the performance of iterative decoders. Minimizing $\Gamma(\mathbf{u},\mathbf{y})$ is equivalent to minimize BER. However, it has derivatives of zero almost everywhere because of the sign ($\text{sgn}(\cdot)$) function. To solve the zero gradients problem of Eq. (6), we apply straight-through estimators in the chain rule to calculate the gradients, which are introduced below.

B. Straight-through estimators

The straight-through estimator (STE) is a proxy derivative used in the quantized NN training to replace the zero derivative of quantization function in the chain rule [29]. Intuitively, STE is an estimate of the true partial gradient. In [29], it was found that the most efficient training of quantized NNs was using STEs, which was a good way to provide a non-trivial search direction.

To see how STE works for proposed NN, we take a look at the backward propagation. The partial derivative of the objective function $\Gamma(\mathbf{u},\mathbf{y})$ as defined in Eq. (6), with respect to $w^{(k)}$ is calculated using chain rule as follows

$$\begin{aligned}\frac{\partial\Gamma(\mathbf{u},\mathbf{y})}{\partial w^{(k)}} &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial(y_i-\hat{y}_i)^2}{\partial w^{(k)}} = \frac{1}{N}\sum_{i=1}^{N}(-2)(y_i-\hat{y}_i)\frac{\partial(\hat{y}_i)}{\partial w^{(k)}}\\ &= \frac{1}{N}\sum_{i=1}^{N}(-2)(y_i-\hat{y}_i)\frac{\partial(0.5-0.5\text{sgn}(u_i))}{\partial w^{(k)}}\\ &= \frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)\cdot\frac{\partial(\text{sgn}(u_i))}{\partial u_i}\cdot\frac{\partial u_i}{\partial w^{(k)}}\end{aligned} \quad (7)$$

Apparently, $\frac{\partial(\text{sgn}(u_i))}{\partial u_i}$ is zero almost everywhere, making the weight updates still. To solve this, we use a proper surrogate derivative $\frac{\partial h(u_i)}{\partial u_i}$, called STE, to replace $\frac{\partial(\text{sgn}(u_i))}{\partial u_i}$ in Eq. (7). Therefore, the partial derivative with respect to $w^{(k)}$ can be approximated by

$$f(w^{(k)}) = \frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)\cdot\frac{\partial(h(u_i))}{\partial u_i}\cdot\frac{\partial u_i}{\partial w^{(k)}} \quad (8)$$

The problem of designing good STEs has been studied extensively in [23]–[26]. We introduce the following function, whose gradient is chosen as the STE for the sign function used in Eq. (6):

$$h^{\text{sgn}}(x) = \begin{cases} x & \text{if } |x|<T \\ \text{sgn}(x)T & \text{otherwise}\end{cases}, \quad (9)$$

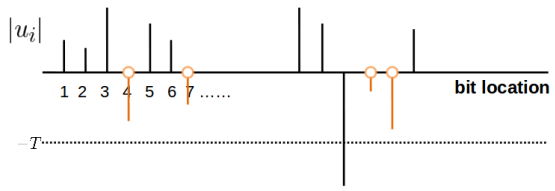

Fig. 2. The mechanism of STE in Eq. (10) assuming that $\mathbf{y}=\mathbf{0}$. The bit locations that contribute to the weight gradients are marked by orange color, e.g., the 4-th, 7-th and so on.

where T is a pre-defined threshold. The gradient of $h^{\text{sgn}}(x)$ with respect to x is given below, which is the STE we use:

$$\frac{\partial h^{\text{sgn}}(x)}{\partial x} = \begin{cases} 1 & \text{if } |x|<T \\ 0 & \text{otherwise}\end{cases}. \quad (10)$$

From Eq. (8), also as shown in Fig. 2, for each bit in a sample, if it is decoded correctly (i.e., $\hat{y}_i = y_i$), it will have no influence on the weight change. If not, and the magnitude of its likelihood message $|u_i|$ is large enough (larger than some threshold T), it will not affect the weight gradient (it is treated as "decoded correctly"). The only case which contributes to the weight change is when the bit is decoded wrongly, and $|u_i|<T$ (its likelihood message is not strong enough).

V. Numerical results

We built NN framework in Python3.6 and used Pytorch library for training. The NN is optimized by ADAM. The training set consists of realizations of depolarizing noise vectors assuming that all-zero codeword is transmitted. The measure of performance is the *frame-error-rate* (FER). We compared the performance of syndrome MS decoding and syndrome NN decoding with same number of iterations.

We consider a class of non-CSS codes introduced in [30], which are *quasi-cyclic* (QC) QLDPC codes. Each QC QLDPC code $\mathcal{C}_{qc}$ of this class has code length of $2dq$ and dq parity check equations, where d is a positive integer and $4d+1$ is an odd prime. $\mathbf{H}_{qc}$ of $\mathcal{C}_{qc}$ has dimension of $dq\times 4dq$, whose circulant permutation matrix has size of q. $\mathbf{H}_{qc}$ has regular column and row weights of d and $4d$, respectively. H_X and H_Z are based on multiplicative groups of order $4d$. $\mathcal{C}_{qc}$ does not have cycles of length 4.

In this experiment, we take $d=3, q=78$. $\mathbf{H}_{qc}$ has dimension of 234×936, and regular column and row weights of 3 and 12, respectively. We construct a neural network of 10 hidden layers corresponding to 5 iterations. Each hidden layer has 2808 neurons ($=936\times 3$). The input layer and output layer have 936 neurons.

The training set consists of 5000 samples at depolarizing $p=0.0008$. The number of epochs is 200, and the learning rate of ADAM is 0.01, with mini-batch size of 100. The pre-defined threshold T is set to be 20. The maximum number of iterations is set to be 5. Fig. 3 gives the FER performance of syndrome MS decoding and syndrome NN decoding with 5 iterations. It is shown that the syndrome NN decoding

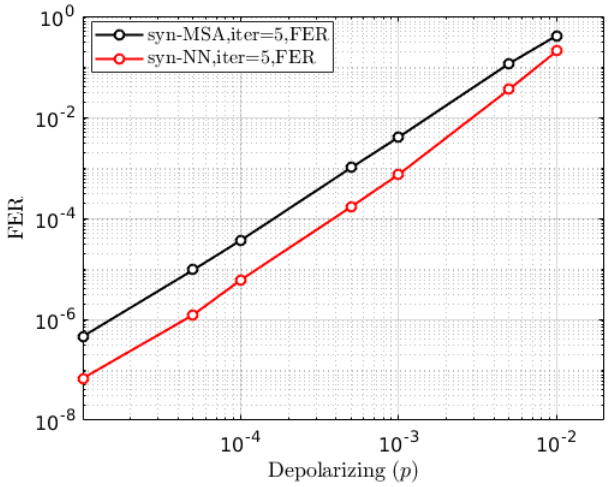

Fig. 3. FER Performance of syndrome MS decoding and syndrome NN decoding of Quantum code for five iterations.

can achieve lower logical error rate of one order of magnitude.

VI. Conclusions and Discussions

In this paper, we introduce a general framework for error correction in QLDPC codes using neural networks. We construct a neural network to perform the syndrome matching algorithm over a depolarizing channel. We propose the BER-based objective function for training. In the training, we introduce a STE to around zero derivatives of the sign function used in hard decision. Simulation results show that within 5 iterations, the syndrome NN decoding can achieve lower logical error rate of one order of magnitude at additional trivial complexity.

We remark that in this work, the STE is chosen based on the channel model. Open questions are left for future research such as how to chose STE, what property is necessary that a STE should have to guarantee convergence for different channel models.

Acknowledgment

This work is funded by the NSF under grant NSF ECCS-1500170 and NSF SaTC-1813401.

References

[1] Y. Be'ery, "eforum on BP, LP and NN/ML decoding of HDPC codes," http://listserv.tau.ac.il/archives/hdpc-codes.html.

[2] O. Simeone, "A Very Brief Introduction to Machine Learning With Applications to Communication Systems," *ArXiv e-prints*, August 2018.

[3] C. Chamberland and P. Ronagh, "Deep neural decoders for near term fault-tolerant experiments," *Quantum Science and Technology 3, 044002*, 2018.

[4] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Information Theory (ISIT), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1361–1365.

[5] E. Nachmani, E. Marciano, D. Burshtein, and Y. Beéry, "Rnn decoding of linear block codes," 2017. [Online]. Available: https://arxiv.org/abs/1702.07560

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

[7] F. Liang, C. Shen, and F. Wu, "An iterative bp-cnn architecture for channel decoding," 2017. [Online]. Available: https://arxiv.org/abs/1707.05697

[8] S. Krastanov and L. Jiang, "Deep neural network probabilistic decoder for stabilizer codes," *Nature, Scientific Reports*, vol. 7, September 2017.

[9] W. Xu, Z. Wu, Y. L. Ueng, X. You, and C. Zhang, "Improved polar decoder based on deep learning," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct 2017, pp. 1–6.

[10] S. Varsamopoulos, K. Bertels, and C. G. Almudever, "Designing neural network based decoders for surface codes," *arXiv preprint arXiv:1811.12456*, 2018.

[11] A. Davaasuren, Y. Suzuki, K. Fujii, and M. Koashi, "General framework for constructing fast and near-optimal machine-learning-based decoder of the topological stabilizer codes," *arXiv preprint arXiv:1801.04377*, 2018.

[12] D. Gottesman, "Fault-tolerant quantum computation with constant overhead," *Quantum Information and Computation*, vol. 14, no. 15–16, pp. 1338–1372, Nov. 2014.

[13] A. A. Kovalev and L. P. Pryadko, "Fault tolerance of quantum low-density parity check codes with sublinear distance scaling," *Phys. Rev. A*, vol. 87, p. 020304, Feb. 2013.

[14] D. MacKay, G. Mitchison, and P. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2315–2330, Oct. 2004.

[15] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, 1997.

[16] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.

[17] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Information and Computation*, vol. 8, no. 10, pp. 987–1000, Nov. 2008.

[18] N. Raveendran, P. J. Nadkarni, S. S. Garani, and B. Vasić, "Stochastic resonance decoding for quantum LDPC codes," in *2017 IEEE Intl. Conf. on Commun. (ICC)*, May 2017, pp. 1–6.

[19] Y.-H. Liu and D. Poulin, "Neural belief-propagation decoders for quantum error-correcting codes," *arXiv preprint arXiv:1811.07835*, 2018.

[20] B. Vasić, X. Xiao, and S. Lin, "Learning to decode ldpc codes with finite-alphabet message passing," in *2018 Information Theory and Applications Workshop (ITA)*. IEEE, 2018, pp. 1–9.

[21] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 599–618, 2001.

[22] 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.

[23] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[24] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," *arXiv preprint arXiv:1802.04680*, 2018.

[25] P. Yin, S. Zhang, J. Lyu, S. Osher, Y. Qi, and J. Xin, "Blended coarse gradient descent for full quantization of deep neural networks," *arXiv preprint arXiv:1808.05240*, 2018.

[26] P. Yin, J. Lyu, S. Zhang, S. J. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Skh4jRcKQ

[27] D. Gottesman, "Class of quantum error-correcting codes saturating the quantum Hamming bound," *Phys. Rev. A*, vol. 54, no. 3, pp. 1862–1868, Sept. 1996.

[28] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. New York, NY, USA: Cambridge University Press, 2011.

[29] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.
[30] Y. Xie and J. Yuan, "Reliable quantum ldpc codes over gf(4)," in *2016 IEEE Globecom Workshops (GC Wkshps)*, Dec 2016, pp. 1–5.