Neural Network Decoders for Permutation Codes Correcting Different Errors

Yeow Meng Chee and Hui Zhang

Industrial Systems Engineering and Management, National University of Singapore, Singapore
Emails: ymchee@nus.edu.sg, isezhui@nus.edu.sg

Abstract—Permutation codes were extensively studied in order to correct different types of errors for the applications on power line communication and rank modulation for flash memory. In this paper, we introduce the neural network decoders for permutation codes to correct these errors with one-shot decoding, which treat the decoding as n classification tasks for non-binary symbols for a code of length n. These are actually the first general decoders introduced to deal with any error type for these two applications. The performance of the decoders is evaluated by simulations with different error models.

I. INTRODUCTION

A. The Background

The permutation code is a subset of the symmetric group, and was extensively studied because of potential applications on power line communication (PLC) and rank modulation (RM) for flash memory. Typically, the errors occurring in these applications include, but not limit to, insertion, deletion, substitution, adjacent transposition, and translocation.

The application of permutation code on PLC channel is in M-ary FSK modulation scheme where symbols are modulated as sinusoidal waves with M different frequencies [6], [10], [14], [22], [37], [39]. The noises prone to occur include additive background noise, impulse noise and permanent frequency disturbance, which can be considered as substitution errors on the permutation matrix of the corresponding codeword. In order to measure the PLC channel model with synchronization issue, it is natural to consider the mixture of these three types of errors and insertion/deletion errors. Nevertheless, the decoding of these mixed errors is a difficult task to achieve with classical decoding algorithms.

For PLC channel without synchronization issue, decoding algorithms for permutation code were provided in [22] and [6], [37] for codes obtained by composition of cosets of permutation groups and distance preserving mappings respectively. For PLC channel with synchronization issue, algorithms for correcting mixture of insertion/deletion/substitution errors for permutation codes were explored in [8], [9], [19], [20], [35]. Their work mainly focused on some special cases, and no efficient general decoder is known yet. In their model, the output of the channel which suffers from the mixed errors is a variation of the original permutation instead of the corresponding permutation matrix, which is not necessary in the PLC channel model for regaining synchronization we consider in this paper.

In RM scheme for flash memory, information is stored in the form of rankings of cell charges [23], [24]. The translocation error, which is an extension of another well-studied error, the adjacent transposition error [1], [5], [7], [24], [31], is caused by moving the rankings of one cell below a certain number of closest ranked cells. Interleaved codes correcting translocation errors equipped with certain Ulam distance were constructed in [13]. However no efficient decoder for the general family of interleaved codes was known yet.

B. The Neural Network Decoders

Recently, a lot of research has been done on decoding with neural networks, we call them *neural network decoders*, see for example [2]–[4], [11], [17], [25], [27]–[30], [32]–[34], [41]. Neural network decoders build upon supervised learning algorithms such as *multi-layer perceptron* (MLP), were proposed to decode codes as a classification task [18]. For binary codes, a single classification is replaced with n binary classifications in [17], [30], where n is the length of code, and they showed that the bit error rate approaches the maximum a posteriori criterion decoding algorithms.

Decoding permutation codes with non-binary symbols may induce more complexity compared to binary case, as we can see in next section that a codeword in a permutation code over n symbols is transformed into square n length bits for PLC application, which implies more decoding complexity even for codes of short length. Fortunately, in the applications of permutation codes, the length of codes is usually not quite large. For example, it was indicated in [16] that, increasing the length of code utilized implies more critical constraints in terms of program and sensing accuracy.

For linear codes, such as BCH codes, polar codes, Reed Solomon codes, deep learning succeeded on improving belief propagation decoding algorithms for large code length, see [2]–[4], [27]–[29], [33], [34], [41].

C. Our Contributions and Organization

In this paper, we introduce low-latency one-shot neural network decoders for permutation codes for the applications of PLC and RM for flash memory. We use MLPs as our decoders, and let the output execute n classification tasks as in [17], [30], but for non-binary symbols, each corresponding to one coordinate of the codeword. Actually, these are the first general decoders introduced to deal with any error type for these two

applications. Experiments show that the decoders can achieve good block error rate when the code has short length and small size, however the decoding ability decays when the code length and size increase.

The organization of the paper is as follows. In Section II, we formally state the error models of these applications, and also the permutation codes that we use in the paper. In Section III, we present the settings of the neural network decoders for permutation codes. In Section IV, we show the performance by simulation, and Section V concludes the paper.

II. THE PROBLEM STATEMENT

In this section, we formally introduce the error models of the two applications, and also provide some permutation codes that we use in simulation.

A. The Channel Model for PLC Channel

In an M-ary FSK modulation scheme for power line communication (PLC), symbols are modulated as sinusoidal waves with n different frequencies. In order to handle the noise, it was proposed in [14] that n detected envelopes can be used in the decoding process. The i-th envelop detector for a frequency f_i, $i \in [1, n] \triangleq \{1, \ldots, n\}$ is followed by a threshold T_i. For values above the threshold, it outputs a one, otherwise, a zero. Hence, we have n outputs per transmitted symbol. A transmitted codeword of length n thus leads to n^2 binary outputs, which are placed in a binary $n \times n$ matrix.

Example 1. Suppose a codeword $\pi = (1, 2, 4, 3)$ is sent through the channel. Let t_j represent the j-th time interval, for $j \in [1, 4]$. If the codeword is received correctly, the output of the demodulator would be

$$
\begin{array}{ccccc}
f_1 & 1 & 0 & 0 & 0 \\
f_2 & 0 & 1 & 0 & 0 \\
f_3 & 0 & 0 & 0 & 1 \\
f_4 & 0 & 0 & 1 & 0 \\
 & t_1 & t_2 & t_3 & t_4
\end{array}
$$

Three types of noises are prone to occur in the output matrix [14], [39], namely, *additive background noise*: a one becomes a zero, or vice versa (with probability p_{bg}); *impulse noise*: a complete column is received as ones (with probability p_{im}); *permanent frequency disturbance*: a complete row is received as ones (with probability p_{pfd}).

In this work, we also consider the PLC channel with synchronization issue, and employ the model in [12] (see Fig. 1). Suppose some symbols enter the queue to be transmitted over the channel. At each channel use, one of the three events occur: *(i)* with probability p_i, a random symbol is inserted and transmitted through the channel; *(ii)* with probability p_d, the next queued symbol is deleted; *(iii)* with probability $1 - p_i - p_d$, the next queued symbol is transmitted through the channel, while also suffering from the above three types of noises from the channel. Here, the insertion/deletion errors correspond to a whole column inserted/missing in the output matrix.

As in [12], we assume $p_i = p_d$, and assume a maximum insertion length $\ell_{\max}$ for each time interval, and thus generate
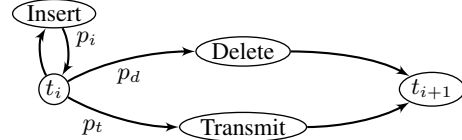

Fig. 1. The channel model for PLC. For each channel use, the "insert" state inserts a random symbol into the channel with probability p_i. With probability p_d, the next queued symbol is deleted, and with probability $p_t = 1 - p_i - p_d$, the next queued symbol is transmitted.

an output matrix of dimension $n \times (n + \ell_{\max} * (n + 1))$ by filling zero columns at the end. In order to reduce storage waste with zero columns, we only take the first $c_{\max}$ columns in each matrix for a preset $c_{\max}$ in decoding phase.

Example 2. Take $\ell_{\max} = 1$ and $c_{\max} = n + 3$. Assume the codeword $(1, 2, 4, 3)$ is sent through the channel, and the output matrix is

$$
\begin{array}{cccccccc}
f_1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
f_2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
f_4 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

We can consider there exists a deletion when sending "2", a permanent frequency disturbance at frequency f_2, and an insertion at the last time slot.

B. The Error Model for RM Scheme

In rank modulation (RM) for flash memory, the rank of a cell reflects the relative position of its own charge level, and the ranking of the n cells induces a permutation (see Fig. 2). The data may be vulnerable to noises caused by potential cell over-injection, charge leakage, and read/write disturbance. The translocation errors were defined to characterize the noises [13], [23], [24]. For a permutation $\pi = (x_1, \ldots, x_n)$, a *translocation* for distinct $i, j \in [1, n]$ is obtained by moving x_i to the j-th position and shift elements between them, including x_j. Thus if $i < j$, the permutation π after translocation i, j is

$$
\cdots, x_{i-1}, x_{i+1}, \cdots, x_j, x_i, x_{j+1}, \cdots
$$

and if $i > j$, it is

$$
\cdots, x_{j-1}, x_i, x_j, \cdots, x_{i-1}, x_{i+1}, \cdots.
$$

In order to characterize all the noises, we set our model as the analysis in [13]. We assume that each cell charge suffers from a small Gaussian noise $n_1 \sim \mathcal{N}(0, \sigma_1^2)$ which may due to read/write disturbance and a low nearly uniform charge leakage rate, and with a small probability p, it also suffers from a possible large Gaussian noise $n_2 \sim \mathcal{N}(0, \sigma_2^2)$, for some $\sigma_2 > \sigma_1$, which may caused by serious cell over-injection and high charge leakage rate. That means, suppose the charge of the i-th cell of the memory is c_i, and the charge retrieved is $c_i + n_1 + n_2$. Apparently, the ranking of the charge levels of all the cells may be reordered.

Example 3. Assume $n = 9$, $\sigma_1 = 0.2$, $p = 0.001$, $\sigma_2 = 1.0$. In Fig. 2, we suppose the original charges
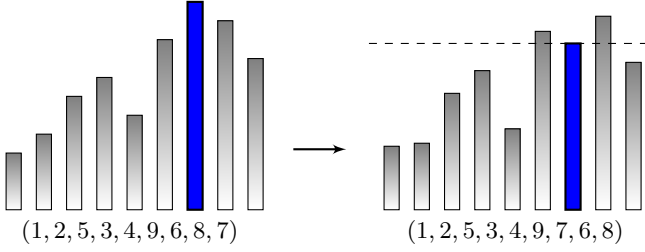
Fig. 2. The two figures show the original charge ranking and retrieved charge ranking suffering from Gaussian noises respectively.

of the cells are $[1.5, 2.0, 3.0, 3.5, 2.5, 4.5, 5.5, 5.0, 4.0]$[1], which corresponds to the permutation $(1, 2, 5, 3, 4, 9, 6, 8, 7)$, and the charges retrieved suffering from the Gaussian noises are $[1.68, 1.76, 3.08, 3.68, 2.14, 4.72, 4.40, 5.12, 3.90]$ corresponding to permutation $(1, 2, 5, 3, 4, 9, 7, 6, 8)$.

C. Permutation Codes and Decoding Algorithms

A permutation code is a subset of the symmetric group $\mathbb{S}_n$, which consists of all permutations of n symbols $1, 2, \ldots, n$.

The following Tenengolts' single insertion/deletion correcting code is well known (see [26], [38]). For any $\pi = (x_1, \ldots, x_n) \in \mathbb{S}_n$, define the vector $\alpha(\pi)$ as

$$\alpha(\pi)_i = \begin{cases} 1, & \text{if } x_{i+1} \geq x_i. \\ 0, & \text{otherwise.} \end{cases}$$

Define the code as the set

$$\mathcal{C}_n = \{\pi \in \mathbb{S}_n : \sum_{i=1}^{n-1} i\alpha(\pi)_i \equiv 0 \pmod{n}\}.$$

However, this code may not be good at correcting substitution errors because of its low Hamming distance. We take the code $\mathcal{C}_n^e$, which consists of only even permutations in $\mathcal{C}_n$ and has minimum Hamming distance at least three.

Permutation code with Hamming distance exactly characterizes PLC channel model without synchronization issue (see Appendix). Decoding algorithms were only provided in [22] and [6], [37] for permutation codes obtained by composition of cosets of permutation groups and distance preserving mappings respectively in literature as far as we know. For comparison with the neural network decoders in this paper, we also provide a minimum distance (MD) decoding in Appendix. For PLC channel with synchronization issue, algorithms for correcting mixture of insertion/deletion/substitution errors for permutation codes were explored in [8], [9], [19], [20], [35]. However their work mainly focused on some special cases. In particular, decoders of $\mathcal{C}_n$ for the case a single error per codeword were provided in [8], [9], and no general classical decoder is known yet.

For RM scheme, we take the interleaved code of size $((n/3)!/2)^3$ from [13, Proposition 15], denoted as $\mathcal{C}_n^{\mathrm{IL}}$, which

[1]Note that we take the charge voltage level arrangement similar as in [16] for simulation in this paper.
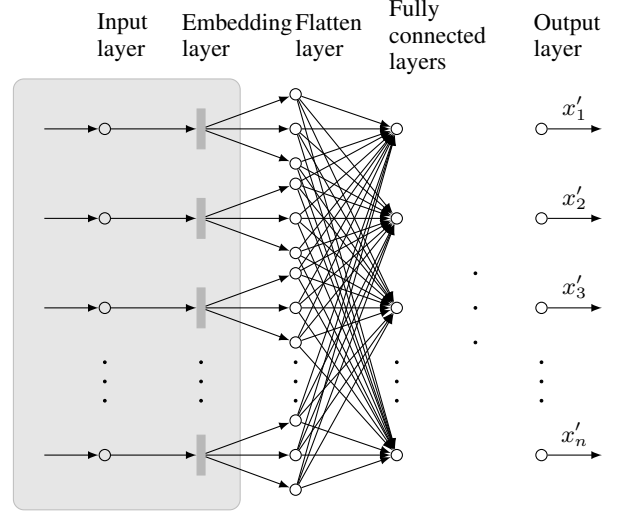

Fig. 3. The Multi-layer Perceptron Decoder

was proved to correct a single translocation error, obtained by interleaving even permutations with $n/3$ symbols. Extensions into t-translocation error correcting codes were also provided in [13]. However no efficient decoder for the general family of interleaved codes was known.

III. THE NEURAL NETWORK DECODERS

In this section, we present the settings of the neural network decoders, that are employed to decode permutation codes. The readers may refer to [18] for more definitions.

A. The Setting of Neural Network Decoders

A *multi-layer perceptron* (MLP) is a class of feed-forward neural network, consisting three types of layers: *input layer*, *hidden layer*, and *output layer*. We describe explicitly the format of different layers for the two applications below.

For RM model, the input layer is just the permutation retrieved from the cells that suffers from Gaussian noises, for example the permutation $(1, 2, 5, 3, 4, 9, 7, 6, 8)$ in Example 3. We take the first hidden layer of the decoder as an *embedding layer*, namely a mapping $f : \{1, \ldots, n\} \to \mathbb{R}^*$, which maps each element of $1, \ldots, n$ into a real vector of fixed length, and it is followed by a *flatten layer* that flats the input of this layer into a one-dimension vector, and several *fully connected* (or *dense*) layers that are connected to every neuron of its preceding layer, and an output layer (see Fig. 3).

As for PLC, we just take the output matrix of the channel of dimension $n \times c_{\max}$ as given in Example 2 as the input layer, which is then followed by a flatten layer, several fully connected layers, and an output layer.

In both cases, the output layer corresponds to the codeword to be retrieved, that was fed into the error models. Actually, the decoders execute n classification tasks, where each of them corresponds to one coordinate of the codeword.

B. The Activation Functions and Output Layers

Each fully connected layer performs an *activation function* $g(WX + b)$, which is taken to be rectified linear unit (ReLU) function $g(Z)_i = \max\{0, z_i\}$ for $Z = (z_1, z_2, \dots)$, where $X = (x_1, x_2, \dots)$ represents the vector received from the neurons in the previous layer.

In the output layer, the k-th output performs a softmax function

$$g(Z_k)_i = e^{z_{k,i}} / \left(\sum_j e^{z_{k,j}} \right)$$

for $i, k \in [n]$, where $Z_k \triangleq (z_{k,1}, \dots, z_{k,n}) = W_k X + b_k$. Finally, $x'_k = \operatorname{argmax}_i g(Z_k)_i$ is taken as the predicted symbol of the k-th coordinate of the codeword (see Fig. 3).

The ReLU function works to make the network approximate nonlinear functions [21], and softmax function normalizes the output to a probability distribution in each of the n classification tasks.

C. The Loss Function

The loss function measures the difference between actual labels and the predicted output. Suppose $\pi = (x_1, \dots, x_n)$ is the codeword fed into the model. Let $\boldsymbol{y}_k$ be the one-hot vector of x_k, which is a vector of length n with all zero except a one at the x_k-th coordinate, and $\hat{\boldsymbol{y}}_k$ be the output of the k-th softmax function, then the loss is defined as the sum of cross-entropy loss of the n outputs, that is

$$Loss = - \sum_{k \in [n]} \sum_{i \in [n]} y_{k,i} \log(\hat{y}_{k,i})$$

where $y_{k,i}, \hat{y}_{k,i}$ are the i-th components of $\boldsymbol{y}_k, \hat{\boldsymbol{y}}_k$ respectively.

IV. IMPLEMENTATION AND PERFORMANCE

In this section, we show the performance of neural network decoders introduced in the previous section by simulation. We take $\ell_{\max} = 1$ and $c_{\max} = n + 3$ or n for PLC model with or without synchronization issue respectively. For RM model, we take the charge levels as $\{1.5 + 0.5i : i = [0, 8]\}$ for $n = 9$ and $\{1.5 + 0.4i : i = [0, 11]\}$ for $n = 12$.

A. The Training/Test Sets and Hyper-Parameters

The supervised learning algorithm produces an inferred function based on the labeled *training set*, and the *test set* is used for verification of the effectiveness of the training process. In the simulations, both the training and test sets are sets of data pairs composed of input and output of the neural network decoders as indicated in the previous section.

In particular, for each point in Fig. 4–Fig. 6, we take a test set of size 10^6, where each original codeword π fed into the error models is randomly chosen from the whole codebook. In order to better visualize the decoding ability for different channel requirement, we train a model for each curve in the figures. For each curve, we let the training set has size δM where M is the size of code, and δ is the number of time that each codeword is fed into the error models in generating the training set pairs. For example, when $n = 6$, for the curve with

code	size	δ	dense layers	#parameters
$\mathcal{C}_6^e$	56	10^6	128	$44,708$ ($42,404$)
$\mathcal{C}_7^e$	360	10^5	128	$48,433$ ($45,745$)
$\mathcal{C}_8^e$	2544	10^4	128	$52,672$ ($49,600$)
$\mathcal{C}_8^e$	2544	10^4	256	$170,816$ ($164,672$)
$\mathcal{C}_9^{\mathrm{IL}}$	27	10^6	64	$18,914$
$\mathcal{C}_{12}^{\mathrm{IL}}$	1728	10^4	64	$27,104$

TABLE I
THE PARAMETERS OF MLP DECODERS.

$p_i = p_d = 0$ and $p_{im} = p_{pfd} = 0.001$, we take the size of training set as $10^6 M$, where each $10^5 M$ of them corresponds to errors with the same p_{im}, p_{pfd}, p_i, p_d, and one p_{bg}.

We let the MLP decoders all have three fully connected layers of the same size, and the embedding size is n if necessary. In order to relief over-fitting, there is a dropout layer with rate 0.1 before the output layer. The number δ, size of dense layers and the total number of trainable parameters are listed in Table I. Notice that, in the first four rows of the last column, the values denote the trainable parameters for PLC model with (or without) synchronization issue respectively.

We implement with TensorFlow library. In the training process, we take batch size as 200, and use Adam algorithm for optimization with a default learning rate 0.001. In order to fully utilize the data in training set, we choose the number of epochs larger than one in the way that the decoders achieve a higher BLER on the test set, which indicates the rounds that entire training set is passed in training process.

B. The Performance and Analysis

The performance of our schemes is measured by the block error rate (BLER) decoding on test set.

In Fig. 4, we consider the PLC channel without synchronization issue. We take the test sets with $p_{im} = p_{pfd} = 0.001$, and the background noises in the set $\{0.005i : i \in [1, 10]\}$. The MD decoder is implemented for comparison. We can see that when $n = 6$, the MLP decoder approaches the MD decoder, however when n increases, the gap also increases. For $n = 8$, an MLP decoder with dense layer size 256 performs better compared to the one with size 128, and more trainable parameters lead to better decoding ability in this case.

In Fig. 5, we consider the PLC channel with synchronization issue. We take the test sets with $p_{im} = p_{pfd} = 0.001$, and the background noises in the set $\{0.001 + 0.003i : i \in [0, 9]\}$, for each $p_i = p_d \in \{0.001, 0.005\}$. We can see that when $n = 6$, the BLER can reach below 10^{-4} for $p_i = p_d = p_{bg} = 0.001$. However, when n increases, the decoding ability seems decay faster than in Fig. 4. For $n = 8$, the decoder with dense layer size 256 also performs better in this case.

In Fig. 6, we consider the MLP decoders for RM scheme for $n \in \{9, 12\}$. We take the test sets with σ_1 in the set $\{0.05i : i \in [1, 10]\}$. When $p = 0$, that is the scheme only suffers from the small disturbance, both the codes can reach zero BLER when σ_1 is small. When σ_2 is twice the gap of two adjacent charge levels, the BLER for $n = 9$ reaches below
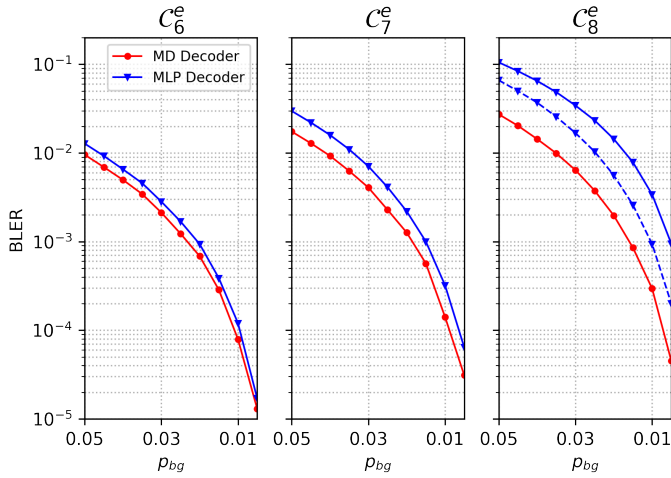
Fig. 4. The Performance of Decoders for PLC Channel without Synchronization Issue: The solid line and dashed line are for MLP decoders with hidden layers size 128 and 256 respectively. Here, $p_{im} = p_{pfd} = 0.001$.
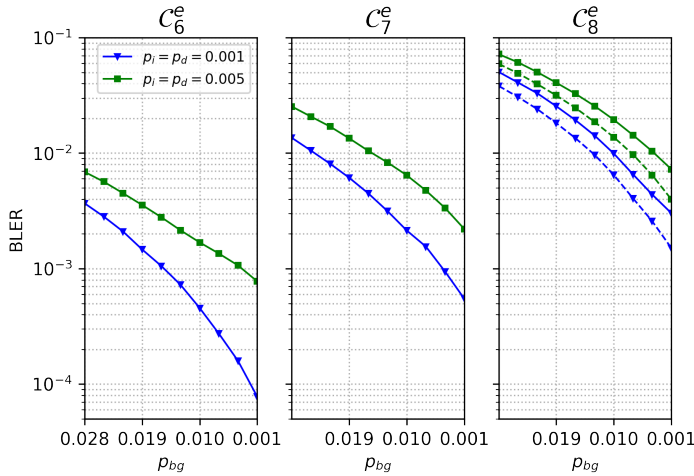

Fig. 5. The Performance of Decoders for PLC Channel with Synchronization Issue: The solid line and dashed line are for MLP decoders with hidden layers size 128 and 256 respectively. Here, $p_{im} = p_{pfd} = 0.001$.

10^{-4} when $p \in \{0.001, 0.005\}$ for small σ_1, and it also holds for $n = 12$, when $p = 0.001$. However, when taking fixed $\sigma_2 = 1$, the code with length 12 has less reliability.

In our decoders, the decoding is treated as n classification tasks. In essence, in training process when labeled data is fed into the neural network, the local features are memorized by trainable parameters, and in decoding phase, the network infers the label of data by recognizing the features. Therefore, compared to the classical decoding algorithms, the neural network decoder may provide more flexibility.

V. Conclusion

In this paper, we introduced neural network decoders for one-shot decoding of permutation codes correcting errors for PLC and RM for flash memory, which are the first general
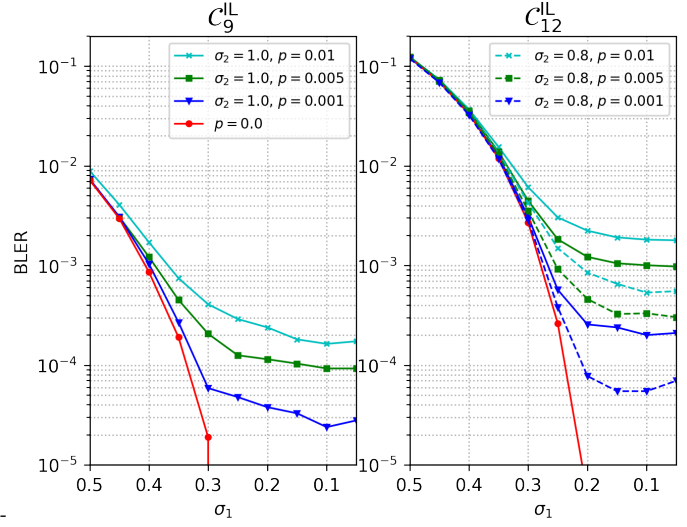

Fig. 6. The Performance of MLP Decoders for RM Scheme.

decoders introduced to correct any error types. Experiments show that the decoders perform well on correcting the errors, in particular for codes of small length and size. The decoding ability decays when the length and size of codes increase. It will be interesting to explore in future work whether it is possible to increase the decoding ability by further increasing the complexity of neural network decoders.

Appendix

The connection between permutation codes with Hamming distance and PLC channel model was implied by not given explicitly in literature. Here, we provide a proof. For a permutation $\pi = (x_1, \ldots, x_n)$, we let Y^π be the $n \times n$ matrix by assigning one in the x_i-th row in the i-th column for $i \in [1, n]$ and zero in all other positions, and let $Y^{\mathcal{C}} = \{Y^\pi : \pi \in \mathcal{C}\}$. Let $\pi, \sigma \in \mathcal{C}$, and the Hamming distances $d_H(\pi, \sigma)$ and $d_H(Y^\pi, Y^\sigma)$ are defined as the number of positions that the two components differ in.

Proposition 4. *A permutation code can correct any combination of e_1 background noise, e_2 impulse noise, and e_3 permanent frequency disturbance with $e_1 + e_2 + e_3 \leq d - 1$ if and only if it has minimum Hamming distance d.*

Proof: For any $\pi, \sigma \in \mathcal{C}$, since $d_H(\pi, \sigma) \geq d$, we have $d_H(Y^\pi, Y^\sigma) \geq 2d$. In the output matrix M suffering from noise, firstly we can locate the rows I and columns J that the impulse noise and permanent frequency disturbance occur, since they are all-one vectors. Then we can consider these rows and columns as erasures. Omitting the chosen rows and columns in all the matrices in $Y^{\mathcal{C}}$, the resulting set of matrices still has minimum Hamming distance at least $2d - 2e_2 - 2e_3$,

and can correct up to $\lfloor (2d - 2e_2 - 2e_3 - 1)/2 \rfloor = d - 1 - e_2 - e_3$ background noise. The reverse is obvious, since if the permutation code has minimum Hamming distance $d-1$, then some $d - 1$ background noise (or impulse noise, permanent frequency disturbance) may ruin the decoding ability of $\mathcal{C}$. ∎

We denote the matrix Y by omitting the rows in I and columns in J as $Y\backslash\{I, J\}$. For a received matrix M of the channel output, we can decode M as the codeword

$$\pi^\star = \text{argmin}_{\pi \in \mathcal{C}} d_H(Y^\pi\backslash\{I, J\}, M\backslash\{I, J\}) \qquad (1)$$

Since the decoding by (1) is time costly, in Fig. 4 we implement the minimum distance (MD) decoding by the following

$$\pi^\star = \text{argmin}_{\pi \in \mathcal{C}} d_H(Y^\pi, M) \qquad (2)$$

When the probabilities of impulse noise, and permanent frequency disturbance are low, that is, mostly only one of them occurs, the decoding with (2) that differs from decoding with (1) will be rare.

REFERENCES

[1] A. Barg and A. Mazumdar, "Codes in permutations and error correction for rank modulation," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3158–3165, Jul. 2010.

[2] 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.

[3] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep learning for decoding of linear codes – a syndrome-based approach," in *Proc. IEEE Int. Symp. Inf. Theory*, Vail, CO, USA, Jun. 2018, pp. 1595–1599.

[4] A. Buchberger, C. Häger, H. D. Pfister, L. Schmalen, and A. G. Amat, "Pruning neural belief propagation decoders," in *Proc. IEEE Int. Symp. Inf. Theory*, Los Angeles, CA, USA, Jun. 2020, pp. 338–342.

[5] S. Buzaglo and T. Etzion, "Perfect permutation codes with the Kendall's τ-metric," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, HI, USA, Jun./Jul. 2014, pp. 2391–2395.

[6] Y. M. Chee and P. Purkayastha, "Efficient decoding of permutation codes obtained from distance preserving maps," in *Proc. IEEE Int. Symp. on Inf. Theory*, Cambridge, MA, USA, Jul. 2012, pp. 641–645.

[7] Y. M. Chee and V. K. Vu, "Breakpoint analysis and permutation codes in generalized Kendall tau and Cayley metrics," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, HI, USA, Jun./Jul. 2014, pp. 2959–2963.

[8] L. Cheng, T. G. Swart, and H. C. Ferreira, "Synchronization using insertion/deletion correcting permutation codes," in *Proc. Int. Symp. on Power Line Commun. and its Applications*, Jeju Island, Korea, Apr. 2008, pp. 135–140.

[9] L. Cheng, T. G. Swart, and H. C. Ferreira, "Re-synchronization of permutation codes with Viterbi-like decoding," in *Proc. Int. Symp. on Powerline Commun. and its Applic.*, Dresden, Germany, Mar. 2009, pp. 36–40.

[10] W. Chu, C. J. Colbourn, and P. Dukes, "Constructions for permutation codes in powerline communications," *Des. Codes Cryptogr.*, vol. 32, pp. 51–64, 2004.

[11] V. Corlay, J. J. Boutros, P. Ciblat, and L. Brunel, "Neural network approaches to point lattice decoding," *IEEE Trans. Inf. Theory*, vol. 68, no. 5, pp. 2969–2989, May 2022.

[12] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions deletions and substitutions," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.

[13] F. Farnoud, V. Skachek, and O. Milenkovic, "Error-correction in flash memories via codes in the Ulam metric," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3003–3020, May 2013.

[14] H. C. Ferreira, A. J. H. Vinck, T. G. Swart, and I. de Beer, "Permutation trellis codes," *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1782–1789, Nov. 2005.

[15] F. Göloğlu, J. Lember, A.-E. Riet, and V. Skachek, "New bounds for permutation codes in Ulam metric," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2015, pp. 1726–1730.

[16] M. Grossi, M. Lanzoni, and B. Riccò, "Program schemes for multilevel flash memories," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 594–601, 2003.

[17] T. Gruber, S. Cammerer, J. Hoydis, and S. T. Brink, "On deep learning based channel decoding," in *Proc. IEEE 51st Annu. Conf. Inf. Sci. Syst.*, Baltimore, MD, USA, 2017, pp. 1–6.

[18] S. Haykin, *Neural Networks: A Comprehensive Foundation (2 ed.)*, Prentice Hall, 1998.

[19] R. Heymann and H. C. Ferreira, "Using tree structures to resynchronize permutation codes," in *Proc. Int. Symp. on Powerline Commun. and its Applic.*, Rio de Janeiro, Brazil, Mar. 2010, pp. 108–113.

[20] R. Heymann, H. C. Ferreira, and T. G. Swart, "Combined permutation codes for synchronization," in *Proc. Int. Symp. Inf. Theory Appl.*, Honolulu, Hawaii, USA, Oct. 2012, pp. 230–234.

[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks, vol. 2, no. 5, pp. 359–366, 1989.

[22] F. H. Hunt, S. Perkins, and D. H. Smith, "Decoding mixed errors and erasures in permutation codes," *Des. Codes Cryptogr.*, vol. 74, pp. 481–493, 2015.

[23] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank Modulation for Flash Memories," in *Proc. IEEE Int. Symp. on Inf. Theory*, Toronto, ON, Canada, Jul. 2008, pp. 1731–1735.

[24] A. Jiang, M. Schwartz, and J. Bruck, "Error-Correcting Codes for Rank Modulation," in *Proc. IEEE Int. Symp. on Inf. Theory*, Toronto, ON, Canada, Jul. 2008, pp. 1736–1740.

[25] C. T. Leung, R. V. Bhat, and M. Motani, "Low-Latency neural decoders for linear and non-linear block codes," *IEEE GLOBECOM*, Waikoloa, HI, USA, Dec. 2019.

[26] V. I. Levenshtein, "On perfect codes in deletion and insertion metric," *Discrete Math. Appl.*, vol. 3, no. 1, pp. 3–20, 1991.

[27] M. Lian, F. Carpi, C. Häger, and H. D. Pfister, "Learned belief-propagation decoding with simple scaling and SNR adaptation," in *Proc. IEEE Int. Symp. Inf. Theory*, Paris, France, pp. 161–165, Jul. 2019.

[28] M. Lian, C. Häger, and H. D. Pfister, "What can machine learning teach us about communications?," in *Proc. IEEE Inform. Theory Workshop*, Guangzhou, China, Nov. 2018.

[29] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. IEEE Int. Symp. on Inf. Theory*, Aachen, Germany, Jun. 2017, pp. 1361–1365.

[30] W. Lyu, Z. Zhang, C. Jiao, K. Qin, and H. Zhang, "Performance evaluation of channel decoding with deep neural networks," in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, USA, May 2018, pp. 1–6.

[31] A. Mazumdar, A. Barg, and G. Zémor, "Constructions of rank modulation codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 2, pp. 1018–1029, Feb. 2013.

[32] M. Mohammadkarimi, M. Mehrabi, M. Ardakani, and Y. Jing, "Deep learning based sphere decoding," *IEEE Trans. Wireless Commun.*, pp. 4368–4378, Sep. 2019.

[33] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annu. Allerton Conf. Commun. Control Comput.*, Monticello, IL, USA, Sep. 2016, pp. 341–346.

[34] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.

[35] T. Shongwe, T. G. Swart, H. C. Ferreira, and T. van Trung, "Good synchronization sequences for permutation codes," *IEEE Trans. Commun.*, vol. 60, no. 5, pp. 1204–1208, May 2012.

[36] D. H. Smith and R. Montemanni, "A new table of permutation codes," *Des. Codes Cryptogr.*, vol. 63, pp. 241–253, 2012.

[37] T. Swart and H. Ferreira, "Decoding distance-preserving permutation codes for power-line communications," *AFRICON*, Sep. 2007, pp. 1–7.

[38] G. M. Tenengolts, "Nonbinary codes, correcting single deletion or insertion (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 30, no. 5, pp. 766–769, Sep. 1984.

[39] A. J. H. Vinck, "Coded modulation for powerline communications," *Proc. Int. J. Elec. Commun.*, vol. 54, no. 1, pp. 45–49, 2000.

[40] G. Zeng, D. Hush, and N. Ahmed, "An application of neural net in decoding error-correcting codes," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol. 2, May 1989, pp. 782–785.

[41] W. Zhang, S. Zou, and Y. Liu, "Iterative soft decoding of Reed–Solomon codes based on deep learning," *IEEE Commun. Lett.*, vol. 24, no. 9, pp. 1991–1994, Sep. 2020.