Improved Tone Reservation Method Based on Deep Learning for PAPR Reduction in OFDM System

Lanping Li*, Chintha Tellambura† and Xiaohu Tang*
* Laboratory of Information Coding and Transmission, Southwest Jiaotong University,
Chengdu 611756, China, E-mail: lanping@my.swjtu.edu.cn, xhutang@swjtu.edu.cn
† Department of Electrical and Computer Engineering,University of Alberta,
Edmonton T6G 2V4, Canada, E-mail:ct4@ualberta.ca

***Abstract*—This paper utilizes deep learning (DL) in tone reservation (TR) to reduce the peak-to-average power ratio (PAPR) of orthogonal frequency division multiplexing (OFDM). We propose TR based on DL (DL-TR) algorithm by considering each iteration of the classical TR algorithm as a layer of a deep neural network (DNN), and then train the network offline to obtain the clipping ratio of each layer which can minimize the loss function that is the sum of PAPR and the increased transmit power. Compared with the conventional TR method, the simulation results show that the proposed DL-TR provides a better PAPR reduction and bit-error-rate (BER) performance.**

***Index Terms*—deep learning (DL), orthogonal frequency division multiplexing (OFDM), peak-to-average power ratio (PAPR), tone reservation (TR).**

I. Introduction

Orthogonal frequency division multiplexing (OFDM) is widely used in many wireless communication system such as 802.11ac for its potential to increase spectral efficiency and robustness against multipath fading. However, when multiple sub-carriers align with the same phase, high peak-to-average power ratios (PAPRs) are generated, which leads to signal distortion, out-of-band radiation , and so on.

To deal with the PAPR problem, many methods have been proposed [1], [2], for examples, clipping (CL), clipping-filtering, block coding, selective mapping (SLM), partial transmit sequence (PTS), tone reservation (TR) and others. Among them, TR by Tellado and Cioffi [3], [4] has two significant advantages: (a) PAPR is reduced without increasing bit-error-rate (BER) or other signal distortions and (b) no side information is exchanged between transmitter and receiver. This method reserves a small number of sub-carriers to generate a time-domain peak-canceling signal to nullify high peaks in the OFDM signal. To find the optimal peak-canceling signal, a sub-optimal gradient algorithm with iterative updates is developed [3]. In few iterations, the TR method suppresses peak values that exceeds predefined target clipping ratio. So the overall PAPR suppression depends on the target clipping ratio. However, this TR algorithm also has two drawbacks: (a) the difficulty for knowing optimal target clipping ratio a-prior and (b) the excessive growth of the total transmit power due to the power allocated to the peak-reduction tones.

Recently, [5] proposed PRNet which is a novel PAPR reduction scheme based on the autoencoder of deep learning. But too many training parameters of deep learning require more training data and training time. To overcome it, many method based on model-driven [6] in deep learning have recently emerged to obtain faster convergence in physics layer of communication, for instance, the improved belief propagation (BP) decoding algorithm using deep learning for BCH codes [7] and polar codes [8], learned iterative shrinkage/thresholding algorithm (LISTA) [9], learned approximate message passing (LAMP) [10].

Inspired by the results of model-driven in deep learning, in this paper, we will unroll the TR algorithm to form a deep neural network (DNN) in which target clipping ratio as network parameters can be learned to minimize the loss function on training examples. In particular, we optimize PAPR and peak-canceling signal simultaneously. Once trained, the network outputs the transmit time domain signal with low PAPR and there is a negligible increase of transmit power. With simulations, we demonstrate that the proposed DL-TR provides better PAPR reduction and improved BER than the traditional TR method given the same number of iterations. The proposed DL-TR also can get a good balance between PAPR and BER with few training parameters and few training samples compared with PRNet.

Notation: $[\cdot]^T$ denotes the transpose of a vector or matrix; Boldface lowercase letters denote column vectors. $\|\cdot\|$ denotes the mean square norm of a vector; Pr and $E[\cdot]$ denote probability and expectation.

II. System Model

A. OFDM System and PAPR

Each OFDM symbol block consisting of N independent, modulated symbols can be represented as $\boldsymbol{X} = [X_0, X_1, \cdots, X_{N-1}]^T$, where X_k is the modulated symbol of the k-th sub-carrier in the frequency domain. After the N-point inverse fast Fourier transpose (IFFT) of $\boldsymbol{X}$, the discrete-time OFDM signal $\boldsymbol{x} = [x_0, x_1, \cdots, x_{N-1}]^T$ can be written as

$$x_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_k e^{\frac{j2\pi nk}{N}}, \quad n = 0, 1, \cdots, N-1 \qquad (1)$$

which can also be expressed in matrix form $\boldsymbol{x} = \boldsymbol{Q}\boldsymbol{X}$, where $\boldsymbol{Q}$ is the IFFT matrix with the (n,k)th element $q_{n,k} = \frac{1}{\sqrt{N}} e^{\frac{j2\pi nk}{N}}$.

The PAPR of $\boldsymbol{x}$ is defined as the ratio of the maximal instantaneous power to the average power [11]:

$$\text{PAPR}(\boldsymbol{x}) = \frac{\max\limits_{0\le n\le N-1} |x_n|^2}{E[\|\boldsymbol{x}\|^2]}. \tag{2}$$

The complementary cumulative distribution function (CCDF) is widely used to measure performance of PAPR reduction, which gives the probability that the PAPR is higher than given value PAPR_0 and is defined as

$$\text{CCDF} = \Pr(\text{PAPR} > \text{PAPR}_0). \tag{3}$$

B. Tone Reservation Technique

The TR method reserves several peak reduction tones (PRT) to generate a time domain peak-canceling vector $\boldsymbol{c} = [c_0, c_1, \cdots, c_{N-1}]^T$. Define $\boldsymbol{C} = [C_0, C_1, \cdots, C_{N-1}]^T$ as its counterpart. To avoid signal distortion, data symbol vector $\boldsymbol{X}$ and the peak reduction vector $\boldsymbol{C}$ are disjoint frequency domain, i.e.,

$$X_k + C_k = \begin{cases} X_k, & k \in \mathcal{R}^c \\ C_k, & k \in \mathcal{R}, \end{cases} \tag{4}$$

where $\mathcal{R} = \{r_0, r_1, \cdots, r_{N_t-1}\}$ represents the index set of the reserved sub-carriers, $\mathcal{R}^c$ is the complementary set of $\mathcal{R}$ in $\{0, 1, \cdots, N-1\}$ and $N_t < N$ is the size of PRT set.

The IFFT of $\boldsymbol{C}$ yields the time domain peak-canceling vector $\boldsymbol{c}$, and it is added to original transmit signal $\boldsymbol{x}$ in order to reduce the PAPR. Thus, the transmit time signal becomes

$$\boldsymbol{a} = \boldsymbol{x} + \boldsymbol{c} = \boldsymbol{Q}(\boldsymbol{X} + \boldsymbol{C}). \tag{5}$$

To facilitate the TR method, the PAPR of the symbol vector $\boldsymbol{a} = [a_0, a_1, \cdots, a_{N-1}]^T$ is redefined as [12]

$$\text{PAPR}(\boldsymbol{a}) = \frac{\max\limits_{0\le n\le N-1} |x_n + c_n|^2}{E[\|\boldsymbol{x}\|^2]}. \tag{6}$$

As shown in (6), $\boldsymbol{c}$ needs to been chosen to minimize $\text{PAPR}(\boldsymbol{a})$. So the optimal peak cancelling signal is given by

$$\boldsymbol{c}^{op} = \arg\min_{\boldsymbol{c}} \max_{0\le n\le N-1} |x_n + c_n|^2. \tag{7}$$

To solve (7), Tellado and Cioffi [3], [4] proposed a simple gradient algorithm to iteratively update the vector $\boldsymbol{c}$ as follows:

$$\boldsymbol{c}^{(i+1)} = \boldsymbol{c}^{(i)} - \alpha_i \boldsymbol{p}[((n - n_i))_N], \tag{8}$$

where $\alpha_i = a_{n_i}^{(i)} - Ae^{j\theta_{n_i}^{(i)}}$, threshold $A = \sqrt{\gamma E[\|\boldsymbol{x}\|^2]}$ is relevant to the clipping ratio γ, $\theta_{n_i}^{(i)}$ is the phase of $a_{n_i}^{(i)} = x_{n_i} + c_{n_i}^{(i)}$, $\boldsymbol{p} = [p_0, p_1, \cdots, p_{N-1}]^T$ is called the time domain kernel and is a constant vector, $\boldsymbol{p}[((n - n_i))_N]$ is a circular shift of $\boldsymbol{p}$ to the right by a value n_i, and n_i is the position of the peak amplitude of $\boldsymbol{a}$ in the ith iteration, which can be calculated by

$$n_i = \arg\max_n |x_n + c_n^{(i)}|. \tag{9}$$

The kernel vector $\boldsymbol{p}$ is computed by

$$\boldsymbol{p} = \frac{\sqrt{N}}{N_t}\boldsymbol{Q}\boldsymbol{P}, \tag{10}$$

where $\frac{\sqrt{N}}{N_t}$ makes sure $p_0 = 1$, and $\boldsymbol{P} = [P_0, P_1, \cdots, P_{N-1}]^T$ is the frequency domain kernel defined by

$$P_k = \begin{cases} 0, & k \in \mathcal{R}^c \\ 1, & k \in \mathcal{R}. \end{cases} \tag{11}$$

After I iterations, the peak-reduced OFDM time signal is

$$\boldsymbol{a}^{(I+1)} = \boldsymbol{x} + \boldsymbol{c}^{(I+1)} = \boldsymbol{x} - \sum_{i=1}^{I} \alpha_i \boldsymbol{p}[((n - n_i))_N]. \tag{12}$$

We can see the PAPR performance depends on the choice of clipping ratio γ. In general, the choice of optimal γ is difficult. We also see that there is not subject to a power constraint $\|\boldsymbol{c}\|^2$ in (7), so the power increased by $\boldsymbol{c}$ can be unbounded. However, in practice, such increase is highly problematic.

III. TR Based on Deep learning

In this section, we represent the original TR algorithm as a DNN, which is called DL-TR. Particularly, we aim to optimize the parameter γ by minimizing the loss function containing both the PAPR and power of peak-canceling signal $\boldsymbol{c}$ over a training set. This can be done offline and then we use the resulting γ to TR method in online deployment. We also analyze the complexity of DL-TR in term of number of trainable parameters.

A. Structure of DL-TR

In the DL-TR network, there are $I + 2$ layers containing input layer, I hidden layers and output layer. The input layer are fed into time domain signal $\boldsymbol{x}$ and the initial value $\boldsymbol{c}^{(1)} = \boldsymbol{0}$.

About the hidden layers, I layers can be constructed by unfolding TR method from (12) with I iterations and each layer has the same structure. The structure of hidden layers of the DL-TR network is illustrated in Fig. 1.

For the i-th hidden layer of the DL-TR, the input are the $\boldsymbol{c}^{(i)}$ from the $(i-1)$-th hidden layer and time domain signal $\boldsymbol{x}$, both of which are an N dimensional complex vector. The output is $\boldsymbol{c}^{(i+1)} = \boldsymbol{c}^{(i)} - \alpha_i \boldsymbol{p}[((n - n_i))_N]$, where

$$\begin{cases} \alpha_i = \eta(\boldsymbol{a}^{(i)}; A^{(i)}) e^{j\theta_{n_i}^{(i)}} \\ \eta(\boldsymbol{a}^{(i)}; A^{(i)}) = \max(\max\limits_{0\le n\le N-1} |a_n^{(i)}| - A^{(i)}, 0) \\ \boldsymbol{a}^{(i)} = \boldsymbol{x} + \boldsymbol{c}^{(i)} \\ A^{(i)} = \sqrt{\gamma_i E[\|\boldsymbol{x}\|^2]} \end{cases}, \tag{13}$$

$\boldsymbol{p}$ and n_i are defined in (10) and (9) respectively. α_i is the residual signal after peak cancellation which is controlled by peak ratio γ_i, $\boldsymbol{p}[((n - n_i))_N]$ reduces the peak at location n_i but maybe increase the value of other locations. To overcome this, $\boldsymbol{\theta} = \{\gamma_i\}$ are learnable variables that are optimized in the training process.

The output layer of DL-TR network is to obtain the final peak-reduced transmit signal $\boldsymbol{a} = \boldsymbol{a}^{(I+1)} = \boldsymbol{x} + \boldsymbol{c}^{(I+1)}$ and final c time domain peak-canceling vector $\boldsymbol{c} = \boldsymbol{c}^{(I+1)}$.

The difference between the TR algorithm and DL-TR is the learnable variables $\boldsymbol{\theta} = \{\gamma_i\}$, which play important roles in the network. When $\gamma_i = \gamma$, $i = 1, \cdots, I$, the DL-TR is simplified to TR.

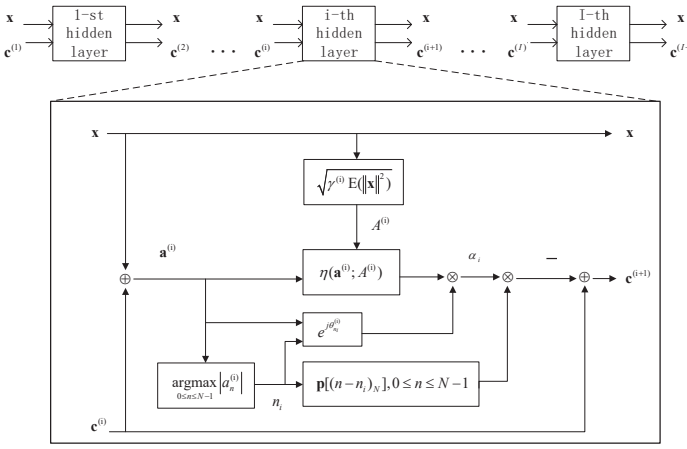

Fig. 1: The structure of the DL-TR network

B. Training of DL-TR

Parameters $\{\gamma_i\}$ in DL-TR network can be optimized to reduce PAPR quickly while avoiding the excessive power of peak-canceling vector $\boldsymbol{c}$. We initialize $\gamma_i = \gamma$ since γ is clipping ratio in TR.

To train DL-TR network, we need generate training samples $\boldsymbol{x}_s$, $s = 1, \cdots, N_s$, where $\boldsymbol{x}_s$ is the transmitted time domain signal. To generate s-th training sample $\boldsymbol{x}_s$, we randomly generate $(N - N_t)\log_2(M)$ bits and obtain M-QAM modulated symbols. After allocating modulated symbols to $(N - N_t)$ subcarriers, we get OFDM symbol $\boldsymbol{X}_s$ in the frequency domain. $\boldsymbol{X}_s$ is then converted to OFDM time domain signal via IFFT, i.e., $\boldsymbol{x}_s = \boldsymbol{Q}\boldsymbol{X}_s$.

For each sample, upon getting $\boldsymbol{x}_s$, with initialization value $\boldsymbol{c}^{(1)} = \boldsymbol{0}$, we apply the DL-TR network under current parameters $\boldsymbol{\theta} = \{\gamma_i\}$ to obtain $\boldsymbol{a}_s$ and $\boldsymbol{c}_s$, which are peak-reduced transmit signal and peak-canceling signal of $\boldsymbol{x}_s$.

The goal of the training process is to minimize the loss function which reduces PAPR of the transmit OFDM signal while avoids the excessive power of peak-canceling vector. We set loss function of s-th sample:

$$l(\boldsymbol{a}_s, \boldsymbol{c}_s; \boldsymbol{\theta}) = \text{PAPR}(\boldsymbol{a}_s) + \lambda \|\boldsymbol{c}_s\|^2, \tag{14}$$

where $\boldsymbol{\theta} = \{\gamma_i\}$ are parameters to be learned and λ is the weighting factor.

The loss function often decomposes as a sum over training examples of per-example loss function, so large training sets lead expensive computation. We will compute loss function over a mini-batch of examples $(\boldsymbol{a}_1, \boldsymbol{c}_1, \cdots, \boldsymbol{a}_D, \boldsymbol{c}_D)$ which are the desired output of the DL-TR network for the transmitted time domain signal $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_D)$ from the training samples. In our experiment settings, the loss function is

$$L(\boldsymbol{a}_1, \boldsymbol{c}_1, \cdots, \boldsymbol{a}_D, \boldsymbol{c}_D; \boldsymbol{\theta}) = \max_{1 \le d \le D} \text{PAPR}(\boldsymbol{a}_d) + \lambda \frac{1}{D} \sum_{d=1}^{D} \|\boldsymbol{c}_d\|^2, \tag{15}$$

where D is mini-batch size. The first term in loss function is the PAPR to be reduced and the second term constraints the average power of peak-canceling vectors.

One popular algorithm to update $\boldsymbol{\theta} = \{\gamma_i\}$ is Adam [13] based on stochastic gradient descent (SGD). It needs one mini-batch samples to update $\boldsymbol{\theta} = \{\gamma_i\}$ one time. The $t+1$-th updating parameter can be compute by

$$\begin{cases} \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\hat{\boldsymbol{m}}}{\sqrt{\hat{\boldsymbol{n}}} + \varepsilon} \quad \text{(element-wise operations)} \\ \hat{\boldsymbol{m}} \leftarrow \frac{\boldsymbol{m}}{1-\mu^t} \\ \hat{\boldsymbol{n}} \leftarrow \frac{\boldsymbol{n}}{1-\nu^t} \\ \boldsymbol{m} \leftarrow \mu \boldsymbol{m} + (1-\mu)\boldsymbol{g} \\ \boldsymbol{n} \leftarrow \nu \boldsymbol{n} + (1-\nu)\boldsymbol{g} \bigodot \boldsymbol{g} \quad (\bigodot \text{multiply by element-wise}) \\ \boldsymbol{g} \leftarrow \nabla L(\boldsymbol{a}_1, \boldsymbol{c}_1, \cdots, \boldsymbol{a}_D, \boldsymbol{c}_D; \boldsymbol{\theta}) \end{cases}, \tag{16}$$

where ∇ is gradient operation, α is the learning rate that can be chosen by hyperparameter optimization, and appropriate default setting are $\mu = 0.9$, $\nu = 0.999$, $\varepsilon = 10^{-8}$. Adam works better than SGD because it considers first and second moments of the gradients to get adaptive learning rates.

When we compute loss function and update $\boldsymbol{\theta} = \{\gamma_i\}$ using only a single example at a time, the optimization algorithm is called online training.

Algorithm 1 Training of DL-TR

1) Collecting training samples:
   Initialize: Set data set $\boldsymbol{G} = \emptyset$.
   For $s = 1, \cdots, N_s$
   a) Generate randomly $(N - N_t)\log_2(M)$ bits;
   b) After M-QAM modulation, modulated symbols maps to an OFDM symbol $\boldsymbol{X}_s$ in the frequency domain;
   c) Compute OFDM time signal $\boldsymbol{x}_s = \boldsymbol{Q}\boldsymbol{X}_s$ via IFFT;
   d) Store $\boldsymbol{x}_s$ in $\boldsymbol{G}$.
   End for
2) Updating parameters $\boldsymbol{\theta} = \{\gamma_i\}$:
   Initialize: Set $\gamma_i = \gamma$ for $i = 1, \cdots, I$, where γ is clipping ratio in TR.
   For $N_e = 1, \cdots, epochs$
   For $N_b = 1, \cdots, \frac{N_s}{D}$
   i) Get D samples $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_D)$ from N_b-th mini-batch of $\boldsymbol{G}$; Feed each sample $\boldsymbol{x}_d$ into DL-TR network in Fig. 1 under current parameters $\boldsymbol{\theta} = \{\gamma_i\}$ with $\boldsymbol{c}^{(1)} = \boldsymbol{0}$ and get output $\boldsymbol{c}_d = \boldsymbol{c}^{(I+1)}$, $\boldsymbol{a}_d = \boldsymbol{x}_d + \boldsymbol{c}^{(I+1)}$;
   ii) Compute loss function (15);
   iii) Update $\boldsymbol{\theta}$ by (16) with $t = (N_e - 1)\frac{N_s}{D} + N_b$.
   End for
   End for
   Return $\boldsymbol{\theta} = \{\gamma_i\}$

After several times of updating parameters, we get $\boldsymbol{\theta}$. In general, there are $epochs * \frac{N_s}{D}$ times to update parameters. One epoch is when the entire training dataset is passed both

forward Fig. 1, (15) and backward (16) through the network only once. $\frac{N_s}{D}$ is the number of mini-batch. The process of training DL-TR is shown in Algorithm 1.

It should be noted that we retrain the network only when one of N, $\mathcal{R}$ and modulation type is changed.

The purpose of training is to determine the optimized parameters $\{\gamma_i\}$ to use in the test stage. After the training, we collect testing data like training samples and adopt the new $\{\gamma_i\}$ in test stage.

C. Complexity

PRNet is composed of a simple encoder and decoder, each of which contains fully connected layers of L hidden layers. There are $N_iN_o + N_o$ trainable variables in each layer where it contains N_i input neurons and N_o output neurons. The numbers of neurons in input layer and output layer of encoder are NM, $2N$, respectively. Suppose there are K neurons in each hidden layer, the total number of trainable parameters of encoder is equal to $\left[NMK + K + (L-2)K^2 + (L-2)K + 2NK + 2N\right]$. The decoder has the same number of parameters as encoder. Thus, the total number of trainable parameters of PRNet is $2\left[NMK + K + (L-2)K^2 + (L-2)K + 2NK + 2N\right]$. Too many training parameters require more training data and training time.

Compared to PRNet, there are only I trainable parameters in the DL-TR network, since each layer contains a adjustable variable $\{\gamma_i\}$. Furthermore, the number of trainable variables of the DL-TR network is independent of the number of subcarriers N and modulation type, and only determined by the number of layers I. This is an advantageous feature when N is large. With only few trainable variables, the stability and speed of convergence can be improved in the training process.

IV. Simulation Results

In this section, we show the performance of the proposed DL-TR algorithm in terms of CCDF of PAPR and BER over AWGN channels. We also evaluate DL-TR with different setting parameters in the training stage.

We build the DL-TR training network on the deep learning framework Tensorflow [14]. Training is conducted using Adam with the learning rate of $\alpha = 0.01$. The total training samples are 200000 and the mini-batch size is set to 100, i.e., $N_s = 2 \times 10^5$, $D = 100$. We train the model for 1 epoch. $\lambda = 0.1$ is default setting in (15) of DL-TR. It is noted that the parameters such as learning rate, mini-batch in deep learning are not arbitrarily chosen but brute-force searched.

In Fig. 2 and Fig. 3, we generate 10^5 OFDM symbol blocks randomly to test the CCDF and BER of different algorithms, respectively. An OFDM system with $N = 64$ sub-carriers and 4-QAM is adopted. The number of reserved PRTs is $N_t = 4$ with random index set. The clipping ratio of TR is $\gamma = 4$ dB. The maximum iteration number is $I = 8$ in both DL-TR and TR in all simulations. For comparison, we set $L = 5$ and $K = 2048$ in PRNet. The parameters to train PRNet are the same as DL-TR except for $\alpha = 0.0001$ and different samples

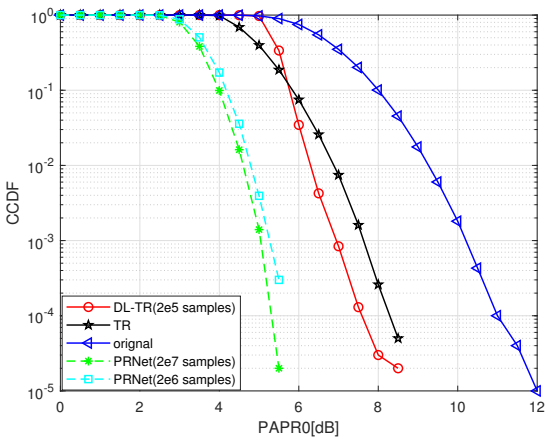

Fig. 2: PAPR of DL-TR ($\lambda = 0.1$) ,TR and PRNet with $N = 64$

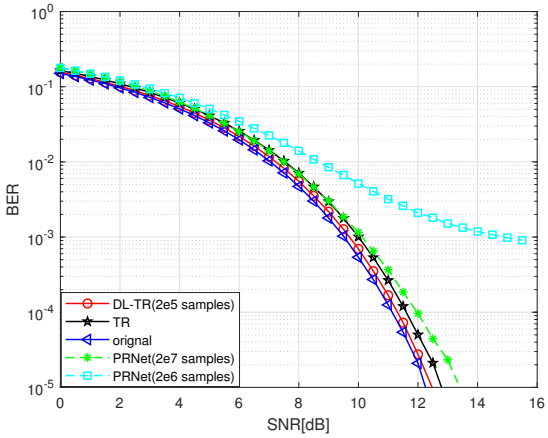

Fig. 3: BER of DL-TR ($\lambda = 0.1$) ,TR and PRNet with $N = 64$

N_s. In Fig. 2, the PRNet with 2×10^7(2e7) and 2×10^6(2e6) training samples, which is respectively 100 times and 10 times of the DL-TR network, have the best and second good CCDF. The PAPR of the DL-TR is lower than that of TR and original schemes.

TR and DL-TR require more signal power than original OFDM because of the additional PRTs. We thus normalize the transmit signals of TR and DL-TR back to the original power level, which degrades BER. The transmit signals of PRNet is also normalized to the original power. Because there are too many trainable parameters in the PRNet, it needs enough training samples to learn constellation mapping and demapping of symbols on each subcarrier in OFDM system for good BER. Fig. 3 indicates the PRNets have worst and second bad performance of BER because PRNet needs more training samples to help find more proper constellation mapping and demapping. The BER of DL-TR is closer to that of original than TR method because DL-TR has less increased power than TR.

Fig. 4 explains CCDF of different algorithms under different

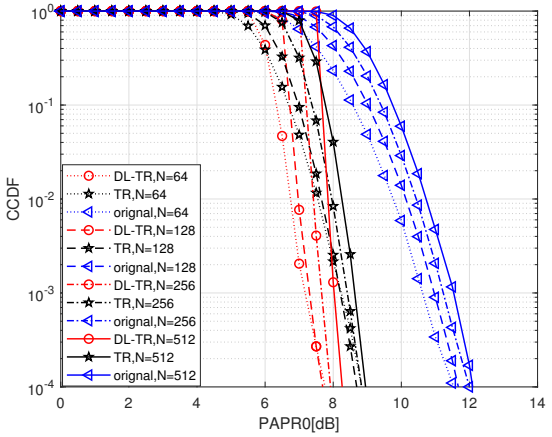

Fig. 4: PAPR of DL-TR ($\lambda = 0.1$) with different N

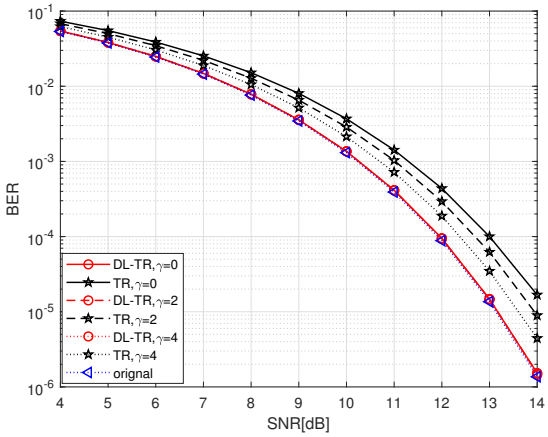

Fig. 6: BER of DL-TR ($\lambda = 0.1$) and TR with $N = 512$.

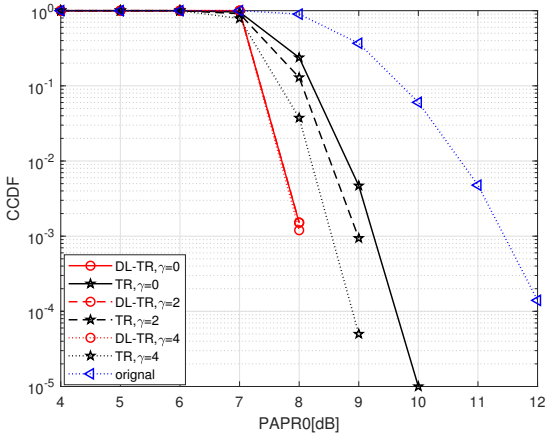

Fig. 5: PAPR of DL-TR ($\lambda = 0.1$) and TR with $N = 512$.

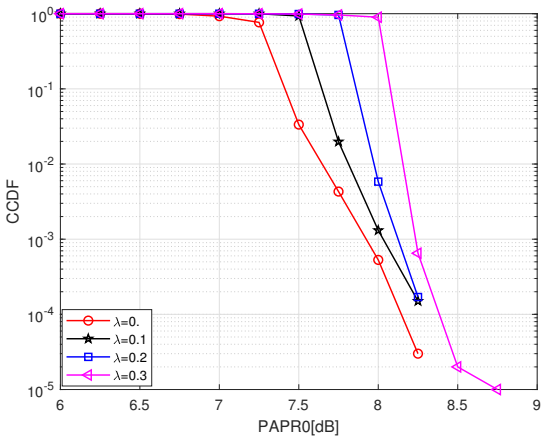

Fig. 7: PAPR reduction in DL-TR with different training λ.

sub-carrier numbers $N = 64, 128, 256, 512$ with 4-QAM. The number of reserved PRTs is $N_t = \frac{1}{16}N$ with random index set. To better estimate the PAPR, the oversampling factor $U = 4$ is employed [11]. We set $N_s = 20000$, $D = 100$, $epochs = 10$ in DL-TR network. We don't consider PRNet with large number sub-carriers due to the problem of number of required training samples. We can see that the larger the number of sub-carrier in the same algorithm, the greater the CCDF. Under the same number of subcarrier numbers, the DL-TR algorithm has the best performance, and the original algorithm has the worst performance.

For the following simulation results, we consider an OFDM system with $N = 512$ sub-carriers, $U = 4$ oversampling factor and 16-QAM. The number of reserved PRTs is $N_t = 32$ with index set in [12]. We set $N_s = 20000$, $D = 100$, $epochs = 10$ in DL-TR network.

In Fig. 5 and Fig. 6, we discuss the impact of clipping ratio γ on the PAPR reduction and BER in TR and DL-TR methods. In TR, clipping ratio is set to $\gamma = 0$ dB, $\gamma = 2$ dB, $\gamma = 4$ dB respectively which is also the initial clipping ratio in DL-TR training stage. $\boldsymbol{\theta}$ is optimized in DL-TR with $\lambda = 0.1$. Fig. 5 illustrates when $CCDF = 10^{-3}$, compared to the original OFDM PAPR, the PAPR could be reduced about by 3.5 dB for the DL-TR method with different initial γ, while the TR scheme can only achieve about 3 dB, 2.5 dB, 2.3 dB PAPR reduction with $\gamma = 0$ dB, $\gamma = 2$ dB, $\gamma = 4$ dB, respectively. It shows that DL-TR outperforms traditional TR, which is sensitive to the clipping ratio γ. DL-TR can almost achieve the same CCDF because all the layers use the optimized $\boldsymbol{\theta}$ after training. BER of DL-TR in Fig. 6 with different initial γ is more or less that of original OFDM, and DL-TR has about 0.4 dB, 0.7 dB, 1 dB gain over TR with $\gamma = 0$ dB, $\gamma = 2$ dB, $\gamma = 4$ dB, respectively.

Fig. 7 and Fig. 8 respectively present PAPR and BER of the DL-TR method with different weighting factor λ. We can observe that the smaller λ leads to better CCDF. We can also see that best PAPR reduction occurs when $\lambda = 0$, which simultaneously has the worst BER due to the lack of power constraint of $\boldsymbol{c}$. DL-TR constrained power using $\lambda = 0.1$, $\lambda = 0.2$ and $\lambda = 0.3$ have nearly the same BER.

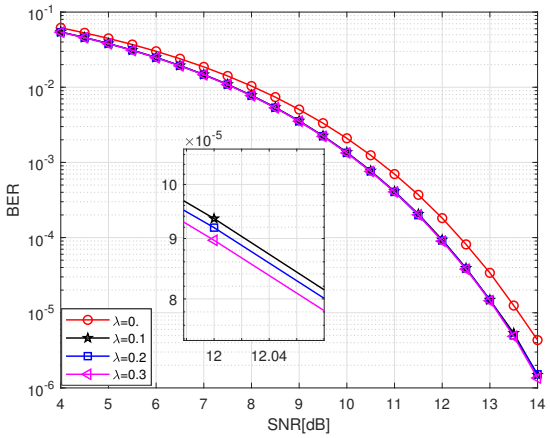

Fig. 8: BER with different λ in DL-TR training stage.

V. Conclusions

In this paper, we proposed a new deep-learning based tone reservation method to reduce the PAPR of OFDM. The DL-TR algorithm is constructed by unfolding the classical iterative TR gradient method; and thus, has the same computational complexity as conventional TR given the same number of iterations. It outperforms conventional TR in terms of both PAPR reduction and BER. We also discussed the effect of different setting parameters in the loss function.

References

[1] S. H. Han and J. H. Lee, "An overview of peak-to-average power ratio reduction techniques for multicarrier transmission," *IEEE Wireless Commun.*, vol. 12, no. 2, pp. 56–65, Apr. 2005.

[2] L. Wang and C. Tellambura, "An overview of peak-to-average power ratio reduction techniques for OFDM systems," in *IEEE Int. Symp. Signal Process. Inform. Tech. (ISSPIT)*, Aug. 2006, pp. 840–845.

[3] J. Tellado and J. M. Cioffi, "PAR reduction in multicarrier transmission systems," *ANSI document, T1E1*, vol. 4, pp. 1–14, Feb. 1998.

[4] ——, "Peak power reduction for multicarrier transmission," in *IEEE Global Commun. Conf. (GLOBECOM)*, vol. 99, 1998, pp. 5–9.

[5] M. Kim, W. Lee, and D.-H. Cho, "A novel PAPR reduction scheme for OFDM system based on deep learning," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 510–513, Mar. 2018.

[6] H. He, S. Jin, C. Wen, F. Gao, G. Ye Li, and Z. Xu, "Model-driven deep learning for physical layer communications," *IEEE Wireless Commun.*, pp. 1–7, 2019.

[7] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. Int. Symp. Inform. Theory (ISIT)*, Jun. 2017, pp. 1361–1365.

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

[9] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Mach. Learning (ICML)*. Omnipress, Jun. 2010, pp. 399–406.

[10] M. Borgerding, P. Schniter, and S. Rangan, "AMP-inspired deep networks for sparse linear inverse problems," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4293–4308, Aug. 2017.

[11] C. Tellambura, "Computation of the continuous-time par of an OFDM signal with BPSK subcarriers," *IEEE Commun. Lett.*, vol. 5, no. 5, pp. 185–187, May 2001.

[12] Y. Wang, W. Chen, and C. Tellambura, "Genetic algorithm based nearly optimal peak reduction tone set selection for adaptive amplitude clipping PAPR reduction," *IEEE Trans. Broadcast.*, vol. 58, no. 3, pp. 462–471, Sep. 2012.

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

[14] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, Nov. 2016, pp. 265–283.