Convolutional Recurrent Neural Network-based Channel Equalization: An Experimental Study

Yang Li*, Minhua Chen*, Yang Yang†, Ming-Tuo Zhou†, Chengxiang Wang‡
*School of Information Science and Technology, ShanghaiTech University
Email: {liyang, chenmh}@shanghaitech.edu.cn
†Key Lab of Wireless Sensor Network and Communication,
Shanghai Institute of Microsystem and Information Technology,
Shanghai Research Center for Wireless Communications, Chinese Academy of Sciences
Email: {yang.yang, mingtuo.zhou}@wico.sh
‡Institute of Sensors, Signals and Systems, School of Engineering and Physical Sciences,
Heriot-Watt University, Edinburgh EH14 4AS, UK
Email: Cheng-Xiang.Wang@hw.ac.uk

***Abstract*—In this paper, we revisit the idea of using deep neural network for channel equalization to account for nonlinear channel distortions as well as temporal variations of radio signals. Our insight is leveraging the the shift-invariant properties of the convolutional neural network (CNN) to learn matched filters analogous to the tap weights of conventional equalizer. Then we feed the learned filters into a subsequent recurrent neural network (RNN) with long-short-term-memory (LSTM) cells for temporal modeling of the channel. We train our proposed CNN-RNN (CRNN) equalizer based on real testbed collected data and enlarge the generalization ability of the learned network model as much as we can to adapt to different channel conditions. Experimental results show that the SER performance for our designated single-input single-output (SISO) system which utilises quadrature phase shift keying (QPSK) modulation scheme with the proposed CRNN-based channel equalizer outperforms that of other equalizers by average 2 to 5 dB at low signal-to-noise ratio (SNR).**

I. Introduction

In digital wireless communication systems, binary symbols are transmitted via a dispersive channel causing the symbols to spread in time and produce ISI [1]. The presence of ISI hinders the efficient use of frequency bandwidth and performance improvement. Basically, the channel can be represented by a complex-valued finite impulse response (FIR) filter and the channel output is the linear combinations of the taps weights and is corrupted by noise. The problem of equalization is to reconstruct the transmitted sequences and counteract the effects of ISI and noise based on the channel observations. Typically, the transmission channel can be affected by both linear and nonlinear distortion. Conventional linear equalization algorithms such as least mean squares (LMS) [2] algorithm, recursive least squares (RLS) [3] algorithm, are not adequate to account for nonlinear equalization tasks due to the presence of nonlinear devices.

Recently artificial neural networks (NN) have attracted much attention on channel equalization due to its capability of nonlinear mapping between input and output spaces [4]. Besides, equalization can be considered as a classification problem in which the NN approach is well justified. In [5]-[7], Patra et al. have shown that NN-based nonlinear equalizers can provide better system performance in terms of bit error rate (BER) than conventional linear equalizer for signals under either pulse amplitude modulation (PAM) or quadrature amplitude modulation (QAM). As a consequence, a variety of NN-based equalizers with different structures are applied in the channel equalization task. In [8], Adali and Sonmez formulate the adaptive channel equalization as a conditional probability distribution learning problem in an information-theoretic approach. Then they parametrize the conditional probability density function of the transmitted signal given the received signal by a sigmoidal perceptron and learn by a stochastic estimator. In [9], Chang et al. introduce an adaptive decision feedback equalizer using multilayer perceptron (MLP) structure accounting for a satellite radio channel. Moreover, a modified back-propagation algorithm with better convergence properties is derived on the basis of delta-bar-delta rule. Since the radio signals are complex-valued, Huang et al. propose a complex-valued multilayer neural network based on the Kalman filter for channel equalization in digital communication systems [10].

Radio signals are time series data, however, none of the above researches takes temporal variations of data into consideration. In this paper, we propose a CRNN-based channel equalizer which addresses the problem of temporal variations of data as well as nonlinear channel distortions. Our insight is leveraging the the shift-invariant properties of the convolutional neural network (CNN) [11] to learn matched filters analogous to the tap weights of conventional equalizer. Then we feed the learned filters into a subsequent recurrent neural network (RNN) [12] with long-short-term-memory (LSTM) [13] cells for temporal modeling and then classify the received symbols. In addition, we validate the proposed CRNN algorithm with datasets collected by our 8×8 parallel multiple-input-multiple-output (MIMO) channel sounder working on 3.5GHz. To the best of our knowledge, this is the first work that applies CRNN algorithm in a channel equalization task. The key contributions of this paper are summarized as follows.

978-1-74052-390-5/17/$31.00 ©2017 IEEE

- We propose an analytical formulation of channel equalization as a conditional probability distribution learning problem, which can be solved by a neural network model. The formulation is inspired by the illuminating work in [8].
- Based on the formulation, a CRNN-based channel equalizer is proposed to cope with the problem of temporal variations of data as well as nonlinear channel distortions.

We compared the SER performance of CRNN with other approaches such as conventional method like RLS, multilayer perceptron (MLP) [14]. Extensive experimental results show the SER performance for QPSK symbols of CRNN-based equalizer outperforms the other two equalizers by average 2 to 5 dB at low SNR.

The rest of this paper is organized as follows. Section II gives the formulation of channel equalization as neural network model. In section III, we propose CRNN and illustrate its architecture. Then we show how we collect the training and testing data for our proposed CRNN as well as visualize the collected data in IV. After that, we train and test our proposed CRNN together with the other two equalizers, RLS and MLP on the same training and testing dataset in V. Finally, the conclusion is drawn in section VI.

II. Problem formulation

In this section, we seek to show how an adaptive channel equalization problem can be formulated as a conditional probability distribution learning problem, which can be solved by a neural network.

Consider the channel equalization problem shown in Fig.1. We assume $\mathcal{S}$ is the symbol alphabet for transmitting, and the size of $\mathcal{S}$ is denoted as K. The probability that the transmitted symbol $x^{(i)} = k$ from the symbol alphabet $\mathcal{S}$ is to be determined from a training sequence, given the finite past of the received signal

$$y^{(i)} = [y^{(i)}, y^{(i-1)}, \ldots, y^{(i-N+1)}]^T, \tag{1}$$

where N is the number of channel taps. Thus the equalization problem is equivalent to learn the conditional probability distribution, which is defined as

$$p_{X|Y=y} = P(X = x|Y = y), \tag{2}$$

and make the classification decision by the maximum conditional probability. Then the NN as a function parametrizes the conditional probability distribution in the following manner:

$$f_{\mathbf{w}} : \mathbf{R}^N \to \mathbf{R}^K \in [1, 2, \ldots, K], \tag{3}$$

which is defined as

$$\begin{aligned} f_{\mathbf{w}}(y^{(i)}) &\equiv P(X|Y, \mathbf{w}) \\ &= g(\mathbf{w}^T y^{(i)}), \end{aligned} \tag{4}$$

where the $N \times K$ parameter matrix $\mathbf{w} = (w_1, w_2, \ldots, w_K)^T$ controls the probability distribution, and $g(\cdot)$ is defined as

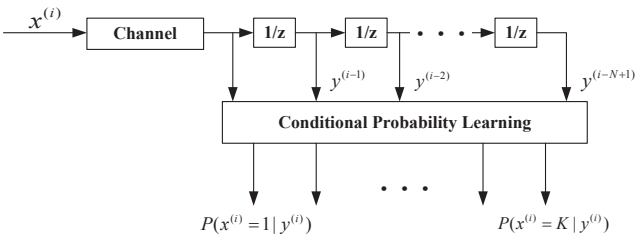

Fig. 1. Illustration of Adaptive Channel Equalization by Conditional Probability Learning.

the softmax activation function in NN terminology. A suitable function is

$$g(\mathbf{x}) = \frac{e^{\mathbf{x}}}{\sum_{j=1}^{K} e^{x_j}} \in [0, 1], \tag{5}$$

where $\mathbf{x} = (w_1^T y(n), \ldots, w_K^T y(n))^T$ and $g(x)$ is a differentiable nonlinearity such that $g'(x) > 0$. Therefore the conditional probability distribution of the K-class transmitted symbols is given by

$$\begin{aligned} p_{X|Y=y^{(i)}} &= \begin{bmatrix} P(x^{(i)} = 1|y^{(i)}; w) \\ P(x^{(i)} = 2|y^{(i)}; w) \\ \vdots \\ P(x^{(i)} = K|y^{(i)}; w) \end{bmatrix} \\ &= \frac{1}{\sum_{j=1}^{K} e^{w_j^T y^{(i)}}} \begin{bmatrix} e^{w_1^T y^{(i)}} \\ e^{w_2^T y^{(i)}} \\ \vdots \\ e^{w_K^T y^{(i)}} \end{bmatrix}. \end{aligned} \tag{6}$$

Note that the term $\frac{1}{\sum_{j=1}^{K} e^{w_j^T y(n)}}$ normalizes the distribution, thus the K conditional probabilities sum to 1.

In this sense, we introduce the cross-entropy loss function which measures the error of prediction that a given set of parameters $\mathbf{w}$ can result in. The function is given by

$$E(w) = -\frac{1}{m} \left[\sum_{i=1}^{m} \sum_{j=1}^{K} \mathbf{1} \left\{ x^{(i)} = j \right\} \log \frac{e^{w_j^T y^{(i)}}}{\sum_{l=1}^{K} e^{w_l^T y^{(i)}}} \right], \tag{7}$$

where m is the total number of the training data $(y^{(i)}, x^{(i)})$ and $\mathbf{1}\{\cdot\}$ is the indicator function, which means if the hypothesis made in the curly braces is correct, it returns 1, otherwise it returns 0. In an instance of the training data, $x^{(i)}$ is defined as the label of $y^{(i)}$, which means the ground truth class of the symbol alphabet for $y^{(i)}$.

Now that we can train a NN with an algorithm automatically tuning the parameters $\mathbf{w}$ to minimize the above error function.

III. The Proposed CRNN-based Equalizer

The proposed CRNN is a pipelined neural network, which the first part is a CNN and the subsequent part is a RNN. The theory of neural networks is comprehensively described

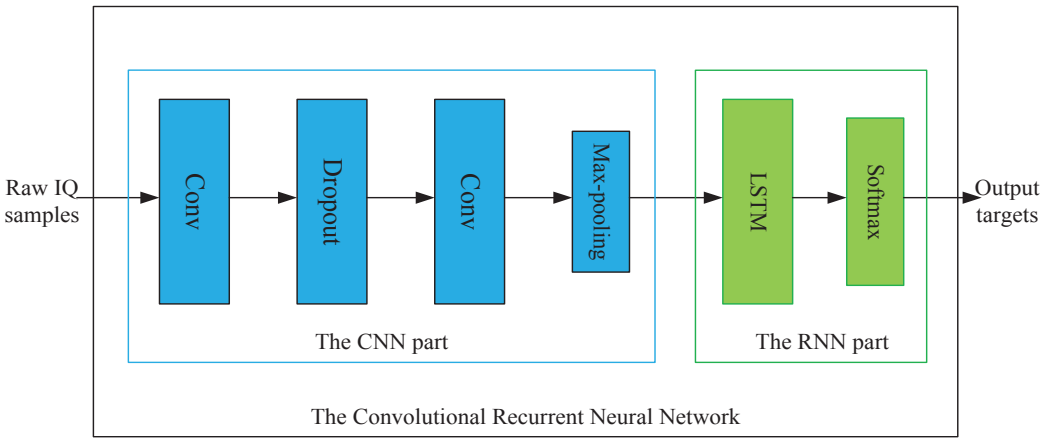

Fig. 2. Architecture of the proposed CRNN.

in [15], thus we will not discuss too much details of neural networks due to the space limit.

Fig.2 depicts our proposed CRNN, which is a 5-layer neural network. The inputs are windowed raw received symbol sequences which every symbol is constructed by the In-Phase and Quadrature (IQ) parts. We assume the window size to be N thus the input is a $2 \times N$ 2-Dimension (2D) vector which is suitable for 2D convolution operation. The first four layers form the CNN part, where the two $Conv$ layers are used to learn matched filters, the dropout [16] layer is used to avoid overfitting the training data, and the $Max-pooling$ layer is used to downsample the output of the previous layer.

The $Conv$ layers consist of a rectangular grid of neurons, where each neuron takes inputs from a rectangular section of the previous layer. The weights for this rectangular grid are the same for each neuron in the $Conv$ layer, moreover, they specify the convoltion filter. The size of a $Conv$ layer is determined by both the size and the number of 2D convolution filters that the $Conv$ layer uses, which is denoted as $Width \times Height \times Number$. After the first $Conv$ layer, a dropout layer is concatenated to control the number of neurons of the previous layer pass to the next layer. The $Max-pooling$ layer takes small rectangular blocks from the previous $Conv$ layer and downsamples it to extract the maximum from that block.

The intuition of applying CNN as the first part of the whole NN-based channel equalizer is that many recovery processes in radio communications systems can be thought of in terms of invariance to linear mixing, rotation, time shifting, scaling, and convolution through random filters (with well characterized probabilistic envelopes and coherence times) [17]. This is analogous to similar learning invariance which is significantly addressed in vision domain learning where matched filters for specific items or features in the image may undergo scaling, shifting, rotation, occlusion, lighting variation, and other forms of noise. Thus we leverage the shift-invariant properties of the convolutional neural network to learn matched filters reducing temporal variations that recover the transmitted signals.

We feed the learned filters into a subsequent layer for temporal modeling by utilizing LSTM cells, which specify the RNN part. The layer which consists of LSTM cells is called $LSTM$ layer. Basically, the $LSTM$ layer learns temporal dependency by memorizing the previous internal state and adding it to the current state at every single time step, and this is what recurrent means. Note weight parameters of the $LSTM$ layer are shared across the time steps. Finally, we use $Softmax$ activation function to derive the outputs for the last layer.

In this sense, CRNN is trained to solve a K-class decision problem given tremendous $(y^{(i)}, x^{(i)})$ instances known as training dataset. We omit the details of how to train the proposed CRNN and do the back-propagation of errors since they could easily be done with freely available open-source machine learning libraries, such as Tensorflow [18], Theano [19], etc.. We use Keras [20] as a convenient high-level abstraction front-end for Tensorflow. It helps to implement and train complex NN models on fast concurrent GPU architectures. For reproducible research, we have made parts of the source code of this paper available [21].

IV. Data preparation

A. Data collection tools and environments

Fig.3 shows our data collecting environment in a typical indoor office environment with our parallel 8×8 MIMO channel sounder which works on 3.5GHz. For simplicity, we use only 1×1 antenna pair known as SISO. At the transmitter, a baseband signal generator sends Pseudorandom Noise (PN) sequences with good cross-correlation characteristics. At the receiver, we use a customized Field Programmable Gate Array (FPGA) board to extract high speed data transferring to disk arrays for storage, and execute appropriate post-processing. For the measurement campaign, the length of QPSK PN sequence is 4096 and IQ chip rate is 100M/s. Each snapshot collected 100 cycles, and a total time length is $10ns \times 4096 \times$

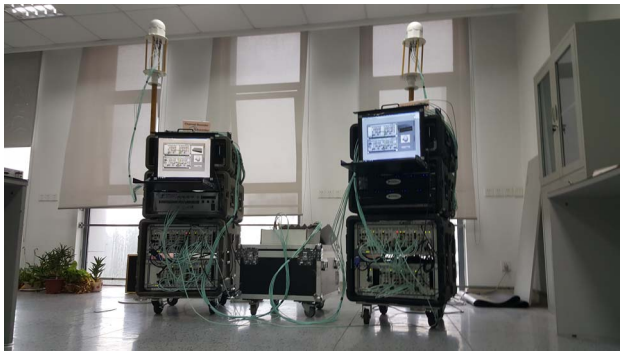

Fig. 3. Data Collecting Equipments and Environment.

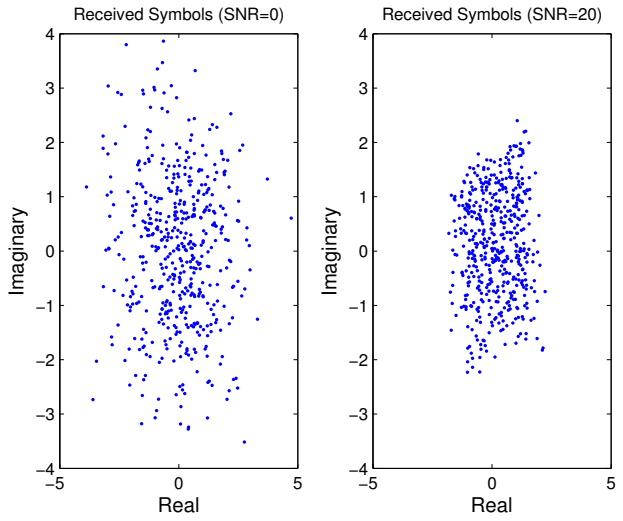

Fig. 4. The scatter diagram of the received symbols under different SNR.

$100 \times 2 = 8.192ms$. The data is stored with a Technical Data Management Streaming (TDMS) file format which can easily be processed. The data is collected uniformly distributing in SNR from 0dB to 20dB and tagged so that we can evaluate the performance of channel equalization algorithms on designated subsets. Specially, we collected 110,000 samples under each SNR, where we use 100,000 of the total samples for training and the rest for testing purpose.

B. Data visualization

Fig.4 shows two scatter diagrams of parts of the received symbols under different SNR. It is obvious that the received symbols under high SNR are more noisy that that of low SNR. Thus to achieve an acceptable SER performance, much more effort is needed to account for the low SNR case. An algorithm is deemed to be more efficient that other algorithms when achieving lower SER under same SNR. Except for our proposed CRNN-based channel equalizer, several other algorithms are also evaluated our with the collected dataset in the next section.

V. Experimental Results

In this section, we train our proposed CRNN with previously obtained data, and then we evaluate the trained CRNN with testing dataset by comparing the SER performance with other techniques. Our training environment is a DELL graphical work station running Ubuntu 16.04 with NVIDIA GeForce GTX 1080 graphical card drived by CUDA 8.0 [22].

A. Hyperparameter optimization

Recall the NN channel equalization algorithm is to tune the weight parameters $\mathbf{w}$ that result in perfect prediction for the unseen signal samples. The parameters except weight parameters of the NN, such as the number of hidden layers, the size of hidden layers and others are defined as hyperparameters. Thus hyperparameter optimization is to choose a set of hyperparameters that optimally suits the NN in both inference and generalization.

We use grid search to perform hyperparameter optimization, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space. Before that, some of the network hyperparameters should be manually determined to avoid a timeless searching. The designated hyperparameters are given in Table I. We introduce $Batch\ size$ except for other hyperparameters in the table that have been specified in previous section, which defines the size of a subset of the whole training samples we used to train the network for every iteration of the learning algorithm. The hyperparameters for optimization are listed as follows

TABLE I
CRNNE Parameters

Input layer size	2×12
Conv1 layer size	$2 \times 4 \times 64$
Conv2 layer size	$2 \times 4 \times 32$
Max-pooling size	2×2
LSTM layer size	100
Output layer size	4
Batch size	1024
SNR	15 dB

$$\begin{aligned} &Learning\,rate : \gamma \in \{1.0 \times 10^{-1}, 1.0 \times 10^{-2}, 1.0 \times 10^{-3}\} \\ &Dropout : \delta \in \{0.0, 0.2, 0.6, 0.9\}, \\ &Training\,epochs : \sigma \in \{10, 20, 30, 40, 50, 60\} \end{aligned} \tag{8}$$

where γ denotes the learning step size, δ stands for the dropout percentage of the previou layer and σ indicates how many times the whole training batches are used. Note we choose the above three hyperparameters as an example of hyperparameter optimization due to space limit, we have to use grid search for all hyperpameters in the real world. Luckily, we find that γ is easy to determine. Fig.5 shows the cross-entropy loss of CRNN with regard to the iterations under different learning rate. When $\gamma = 1.0 \times 10^{-1}$ and whatever the other two hyperparameters are, the cross-entropy loss fails to decrease,

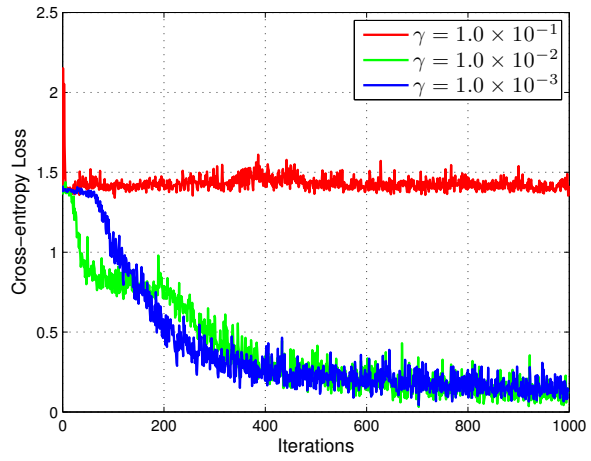

Fig. 5. The cross-entropy loss of our proposed CRNN versus three different learning rate.

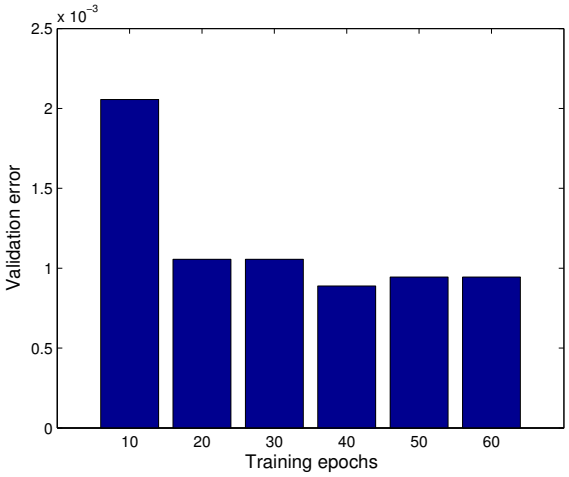

Fig. 7. Influence of the training epochs on the validation error of our proposed CRNN with learning rate $\gamma = 1.0 \times 10^{-3}$ and $\delta = 0.0$.

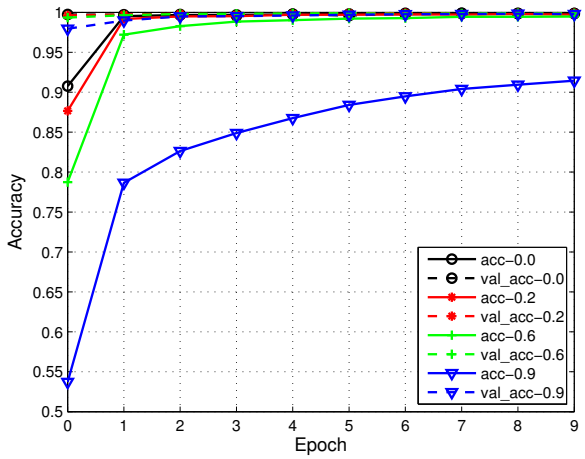

Fig. 6. Training accuracy and validation accuracy of our proposed CRNN versus different dropout values with learning rate $\gamma = 1.0 \times 10^{-3}$. The dashed lines are training accuracy and the solid lines are validation accuracy.

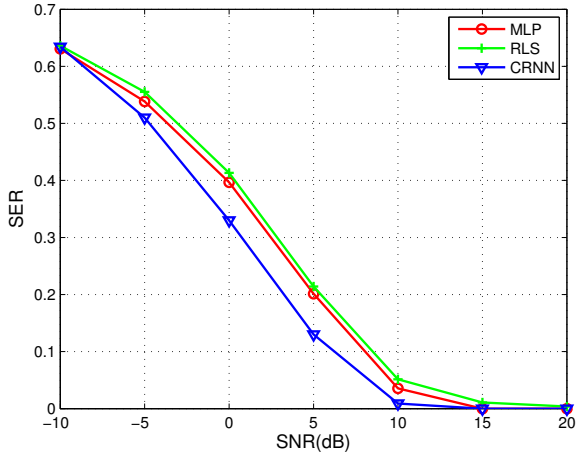

Fig. 8. SER performance of our proposed CRNN versus RLS and MLP under different SNR.

which indicates that the learning rate is too aggressive for the network to learn a proper direction that minimizes the loss. When $\gamma = 1.0 \times 10^{-2}$ or $\gamma = 1.0 \times 10^{-3}$, the cross-entropy loss decreases and converges after 1,000 iterations. It is not explict but the cross-entropy loss with $\gamma = 1.0 \times 10^{-3}$ converges to a lower value than that with $\gamma = 1.0 \times 10^{-2}$. Thus the learning rate for CRNN is determined to be 1.0×10^{-3}. We divide the training dataset into two disjoint subsets, training set and a validation set. Thus the validation accuracy evaluates the ability of generalization since the network is evaluated with samples that never appear in training set. Fig.6 shows the training accuracy and validation accuracy that the network can achieve with different dropout values. It seems there is no need to apply dropout in our network since $\delta = 0.0$ has the highest validation accuracy, which means our network does not overfit the data.

We use validation error to inversely reflect the validation accuracy since there are minor differences when applying different training epochs. Fig.7 indicates training epochs has upper bounds for accuracy gains. Thus we have to search for the least upper bound to aviod over training. Fortunately, we can use early stopping [23] to stop the training process when accuracy gains vanish. As shown in Fig.7, we choose training epochs $\sigma = 40$ with the lowest validation error.

Up to now, we have optimized the hyperparameters and finally chosen learning rate $\gamma = 1.0 \times 10^{-3}$, dropout $\delta = 0.0$ and training epochs $\sigma = 40$. Then we can use these hyperparameters to retrain our network and obtain the best model.

B. SER Performance

We use SER perfomance to evaluate CRNN-based equalizerf since it is a significant metric of channel equalizers. We also implemented a RLS-based equalizer and a MLP

equalizer in comparision with our method. We omit the details of implementing RLS-based and MLP-based equalizers due to space limit. All the three equalizers are trained with the same training dataset.

Fig.8 shows the SER performance of the three equalizers with regard to different SNR conditions from -10dB to 20dB. It is clear to see that all the three equalizers cannot perform proper symbol recovery when the SNR is below 0dB since they can only learn from noise under this circumstance. When SNR ranges from 0dB to 10dB, CRNN-based equalizer outperforms the RLS-based equalizer by average 5dB and the MLP-based equalizer by average 2dB. While SNR is above 10dB, the SER performance of the MLP-based equalizer is similar with that of CRNN-based equalizer. However, only when SNR is above 16dB can the RLS-based equalizer obtain an acceptable SER performance.

Since our training data are time series signals, convolution layers are utilized to reduce temporal variations in our proposed CRNN architecture. Moreover, we utilize an LSTM layer for temporal modeling with memory. These features augment the ability of modeling the time variant channel and thus render CRNN-based channel equalizer a better channel equalizer than other NN-based channel equlizers.

VI. Conclusions

In this paper, we propose an analytical formulation of channel equalization as a conditional probability distribution learning problem, which can be solved by a neural network model. Then we propose a CRNN-based equalizer to account for nonlinear channel distortions as well as the problem of temporal variations of time series data. We collect the practical time series data for training, validation and testing with our 8×8 parallel channel sounder. We use grid search to find the best hyperparameters that fit our model. Then we train and test CRNN-based equalizer together with the other two RLS-based and MLP-based equalizers. Experimental results show the SER performance of our proposed CRNN-based equalizer outperforms the other two equalizers by average 2 to 5 dB at low SNR. Future investigations will be based on more sophisticated hyperparameter optimization and more realistic scenarios such as Multi-user MIMO (MU-MIMO).

VII. Acknowledgement

The authors would like to acknowledge the support from the EU H2020 ITN 5G Wireless project (Grant No. 641985), EU FP7 QUICK project (Grant No. PIRSES-GA-2013-612652), and EPSRC TOUCAN project (Grant No. EP/L020009/1).

References

[1] X. Lyu, W. Feng, R. Shi, Y. Pei and N. Ge, "Artificial neural network-based nonlinear channel equalization: A soft-output perspective," in *Proceedings of the International Conference on Telecommunications (ICT)*, pp. 243-248, 2015.

[2] C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. ASSP-32, pp. 34-41, 1984.

[3] Y. Wang,"Channel Equalization Using a Robust Recursive Least-Squares Adaptive-Filtering Algorithm," in *Proceedings of the IEEE International Conference on Computer and Information Technology (ICCIT)*, pp. 135-138, 2012.

[4] K. Burse, R. N. Yadav and S. C. Shrivastava, "Channel Equalization Using Neural Networks: A Review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 3, pp. 352-357, 2010.

[5] J. C. Patra, Wei Beng Poh, N. S. Chaudhari and A. Das, "Nonlinear channel equalization with QAM signal using Chebyshev artificial neural network," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 5, pp. 3214-3219, 2005.

[6] J. C. Patra, W. C. Chin, P. K. Meher and G. Chakraborty, "Legendre-FLANN-based nonlinear channel equalization in wireless communication system," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 1826-1831, 2008.

[7] C. Y. Lo and W. d. Weng, "Application of Neural Network Techniques on Nonlinear Channel Equalization for 16-QAM Modulation Systems," in *Proceedings of the Eighth International Conference on Intelligent Systems Design and Applications*, pp. 356-361, 2008.

[8] T. Adali and M. K. Sonmez, "Channel equalization with perceptrons: an information-theoretic approach," in*Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol.3, pp. 297-300, 1994.

[9] P. Chang and B. Wang, "Adaptive decision feedback equalization for digital satellite channels using multilayer neural networks," IEEE Journal on Selected Areas in Communications, vol. 13, no. 2, pp. 316-324, 1995.

[10] Ren-Cheng Huang and Mu-Song Chen, "Adaptive equalization using complex-valued multilayered neural network based on the extended Kalman filter," in *Proceedings of the International Conference on Signal Processing*, vol.1, pp. 519-524, 2000.

[11] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[12] H. Zhao, X. Zeng and Z. He, "Low-Complexity Nonlinear Adaptive Filter Based on a Pipelined Bilinear Recurrent Neural Network," *IEEE Transactions on Neural Networks*, vol. 22, no. 9, pp. 1494-1507, Sept. 2011.

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, Nov. 1997.

[14] R. R. de F. Attux et al., "MLP-Based Equalization and Pre-Distortion Using an Artificial Immune Network," in *Proceedings of the IEEE Workshop on Machine Learning for Signal Processing*, pp. 177-182, 2005.

[15] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol.15, pp. 19291958, 2014.

[17] Timothy J. OShea and J. Corgan. "Convolutional radio modu- lation recognition networks," *CoRR*, abs/1602.04105, 2016.

[18] https://www.tensorflow.org.

[19] https://github.com/Theano/Theano.

[20] https://github.com/fchollet/keras.

[21] https://github.com/ColdCodeCool/available-after-review.

[22] https://developer.nvidia.com/cuda-toolkit.

[23] G. Federico, M. Jones, T. Poggio. "Regularization Theory and Neural Networks Architectures," *Neural Computation*, vol. 7, no. 2, pp. 219269, 1995.