Joint Message Passing and Auto-Encoder for Deep Learning

Yiqun Ge [1], Wuxian Shi [1] , Jian Wang [2], Rong Li [2] and Wen Tong [1]

[1] *Wireless Technology Laboratory, Huawei Technologies Co., Ltd., Ottawa K0A3M0, Canada*
[2] *Wireless Technology Laboratory, Huawei Technologies Co., Ltd., Hangzhou 310051, China*

Abstract

Autoencoders (AE) are emerging artificial neural networks that learn efficient embedding of unlabeled data and have been considered in the design of end-to-end transceivers. However, AE-based end-to-end transceivers face with a major challenge of poor generalization ability due to the communication channel dynamics. In this paper, a message-passing algorithm (MPA) layer is incorporated into an AE to simultaneously enable coarse learning in training phase and adaptive reasoning in inference phase. Theoretical analysis is also conducted to demonstrate the effectiveness of the MPA layer, recommending that the proposed model is applicable in more general systems.

Keywords

Autoencoder, end-to-end communication, transceiver, MPA, back-propagation.

1. Introduction

Machine Learning (ML) is envisioned as a promising means to enable an intelligent six-generalization (6G) network. It has attracted extensive interests in both academia and industry. As a popular ML model, Autoencoder (AE) model learns some hidden but efficient data representations by combining the two neural networks that fits well with the classic transceivers in wireless communications, one neural network for an encoder and another for a decoder.

AE-based end-to-end (E2E) transceiver aims to extract the most essential information minimum for a specific goal. It naturally requires a joint design of learning algorithms and communication techniques. Specifically, an encoder neural network learns to act as a transmitter, while a decoder one learns as a receiver. By iterative data sensing and model training, AE-based E2E transceiver actually integrates the three factors, data source distribution, goal orientation, and radio channel, into one framework. In recent years, there is much research interest on this topic [1, 2, 3, 4, 5].

Despite of AE-based end-to-end transceiver's better performance than classical ones, the framework still suffer from the following three challenges:

- **Inaccurate Gradient Transmission**: Training an AE E2E transceiver needs a channel model that is mathematically differentiable to support the backward propagation (BP) of gradients from the receiver side to the transmitter side. Nevertheless, a realistic channel model must include some non-linear components such as digital/analog pre-distortion and other un-differentiable stages like up/down sampling. Therefore, the channel model used in AE E2E transceiver training tends to be oversimplified to support inaccurate gradient transmission.
- **Excessive and Dynamic Channel Distortion**: Essentially, learning on a hidden layer, or an intermediate layer, is to react or adapt itself to the posterior probability of its input signal. In an AE-based E2E transceiver framework, the first layer of the receiver is such an intermediate layer that its input signal is varied with the dynamic channel distortion.

[1] AI6G'22: First International Workshop on Artificial Intelligence in beyond 5G and 6G Wireless Networks, July 21, 2022, Padua, Italy
EMAIL: {yiqun.ge, wuxian.shi, wangjian23, lirongone.li, tongwen}@huawei.com

Furthermore, the channel variation will penetrate forward to the entire receiver neural network part. In case of a fast varying channel, a well-trained receiver might as well become obsolete quickly and receiving performance degrades soon.

- **Loss of Important Connectivity**: According to [6], some outputs of a DNN-based classifier relies on some shortcuts (localities) inside the entire deep neural network. Some parts of the neural network is more important than others. "Shortcuts" are designated to the important part. If these shortcuts were perturbed, the classification performance would degrade significantly. However, the path loss and thermal random noise in communication channels may affect the critical shortcuts in some probability, largely undermining the AE-based E2E transceiver's performance.

These three issues result in the poor generalization ability of the AE-based transceiver over the dynamic and time varying wireless environment. Some prior works have developed some solutions. In [7], a two-phase training strategy is proposed, where the AE-based transceiver is first trained through a stochastic channel model offline, and fine-tuned when it is used in the real channel. To obtain a differentiable channel, [8, 9] proposed to approximate the unknown real channel through generative adversarial networks (GANs). With a trained GAN model connecting the encoder and decoder, both forward inference and backward propagation can be conducted. Since the obstacle caused by the unknown channel is that the back-propagated gradients are not easy to get at the encoder side, methods for gradient estimation are introduced in [10, 11].

Although the aforementioned works can overcome the three issues to some extent, all of them demand high energy consumption, large controlling overhead, and none of them can meet the real-time latency requirement in future wireless communications. These issues motivated us to incorporate a MPA layer into an AE. Its existence can simultaneously solve the out-of-distribution (OOD) and outlier problems usually in the inference stage and reduce the communication and computation costs.

2. Autoencoder-based Transceiver Design with MPA

In this section, we first propose a new AE-based transceiver by inserting an MPA layer with the transmitter, i.e., MPA-AE. Then, we introduce the detailed training process, including the forward sub-iteration and backward sub-iteration.

2.1. Autoencoder-based transceiver with MPA

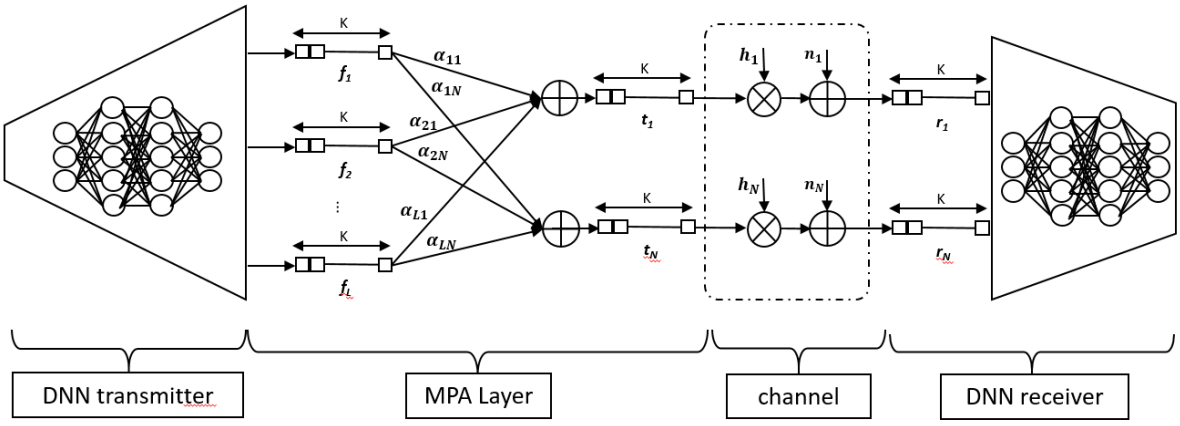

Figure 1: AE-based transceiver with an MPA layer

As illustrated in Fig. 1, we consider an AE-based transceiver that consists of a DNN-based transmitter and a DNN-based receiver. The transmitter and the receiver are connected by wireless channels. To adapt to the dynamic channel conditions, we insert an MPA layer between the transmitter and the communication channel. Without loss of generality, we assume that the channel state information is available at the transmitter, which can be realized by concurrent channel feedback or uplink/downlink channel reciprocity.

The MPA layer is responsible for conducting a linear dimension reduction transformation, whose coefficients are fine-tuned using an iterative algorithm with two sub-iterations. The first one is the forward sub-iteration that passes messages from the DNN transmitter to the DNN receiver through the channel. The other is the backward sub-iteration that passes message from channel layer to the output layer of the DNN transmitter. To better describe the working mechanism of the introduced transceiver, we first provide the key parameters throughout the paper, as summarized in Table I.

Table 1
System Parameters

Parameters	Meaning
L	Dimension of the transmitter's output
N	Dimension of the communication channel measurement
$\boldsymbol{h}_k$	The k-th channel measurement
$\boldsymbol{n}_k$	The n-th additive noise measurement
$\boldsymbol{f}_i$	The i-th feature of the transmitter's output
$\boldsymbol{t}_k$	The k-th feature vector of the MPA layer's output
$\boldsymbol{r}_k$	The k-th received signal
F	Input feature matrix $[\boldsymbol{f}_1, \dots, \boldsymbol{f}_L]$
H	Channel vector $[\boldsymbol{h}_1, \dots, \boldsymbol{h}_N]$
N	Noise vector $[\boldsymbol{n}_1, \dots, \boldsymbol{n}_N]$
R	Received signal vector$[\boldsymbol{r}_1, \dots, \boldsymbol{r}_N]$
T	Output feature matrix $[\boldsymbol{t}_1, \dots, \boldsymbol{t}_N]$

Based on the notations, we elaborate the detailed training process in the sequel.

2.2. Forward Sub-iteration with Support Vector Machine

Support vector machine (SVM) is a supervised machine learning model used for data classification, regression, and outlier detection. In general, an SVM model is composed of a non-linear dimension extension function $\varphi(\cdot)$, a linear combination function $f(\mathbf{x}) = \mathbf{w} \cdot \varphi(\mathbf{x}) + \mathbf{b}$, and a binary classification function sign$(\cdot)$, where $\mathbf{x}$ is the input data, $\mathbf{w}$ is the weight coefficient vector and $\mathbf{b}$ is the bias vector. The objective of SVM is to divide the data samples into classes to find a maximum marginal hyper-plane.

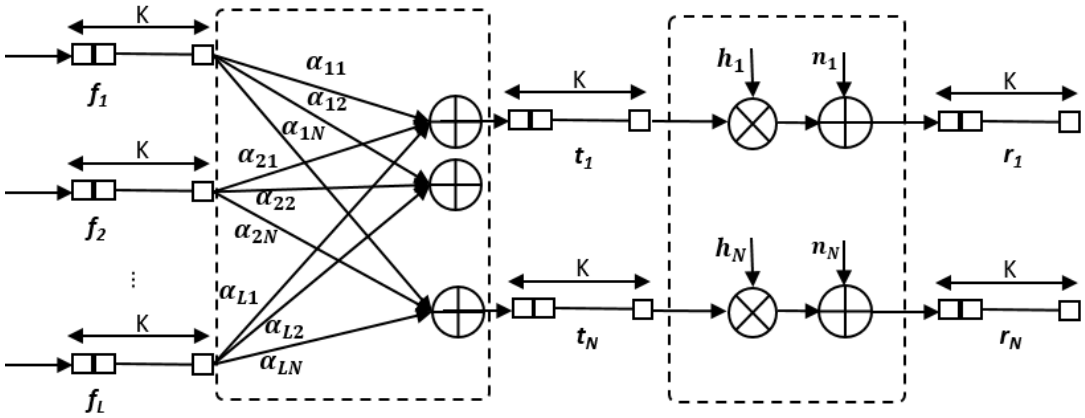

Figure 2: MPA layer with SVM

Taking advantage of the dimension transformation of SVM, we use it to transform the dimension of the transmitter's output, L, to the dimension of the communication channel measurement, N. Fig. 2 shows the detailed forward iteration with SVM. Specifically, the input of the MPA-layer is an L-

dimensional feature matrix $\mathbf{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \dots, \boldsymbol{f}_L]$, where $\boldsymbol{f}_i$ is the *i*-th input feature vector with *K*-dimension. The output of the MPA-layer is an *N*-dimensional feature matrix $\mathbf{T} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{t}_N]$ where $\boldsymbol{t}_i$ is the *i*-th output feature vector with *K*-dimension. When the output feature vectors are transmitted via communication channels, the received signal is given by

$$\boldsymbol{r}_i = \sum_{l=1}^{L} \alpha_{l,i} \cdot \boldsymbol{f}_l \cdot \boldsymbol{h}_i + \boldsymbol{n}_i,\ i = 1, \dots, N,$$

where $\alpha_{l,i}$ is the coefficient of the connection between neuron *l* and neuron *i*.

Based on the above description, we can conclude that the forward sub-iteration is to keep fine-tuning the hyper-plane of the SVM model in both training and inference phases for given transmitter's feature matrix **F**, channel state information **H,** noise vector **N,** and received signal **R**.

Note that the MPA layer is mathematically differentiable. Once the coefficients $\alpha_{l,i}$ are fixed, it can pass the BP gradients from the receiver side to the transmitter side during the training stage.

2.3. Backward Sub-iteration with Attention-DNN

As we discussed earlier, the MPA layer needs to be trained by a standalone mode rather than a connection mode with back-propagation from the receiver. In this regards, we consider to use an attention-DNN in the backward sub-iteration.

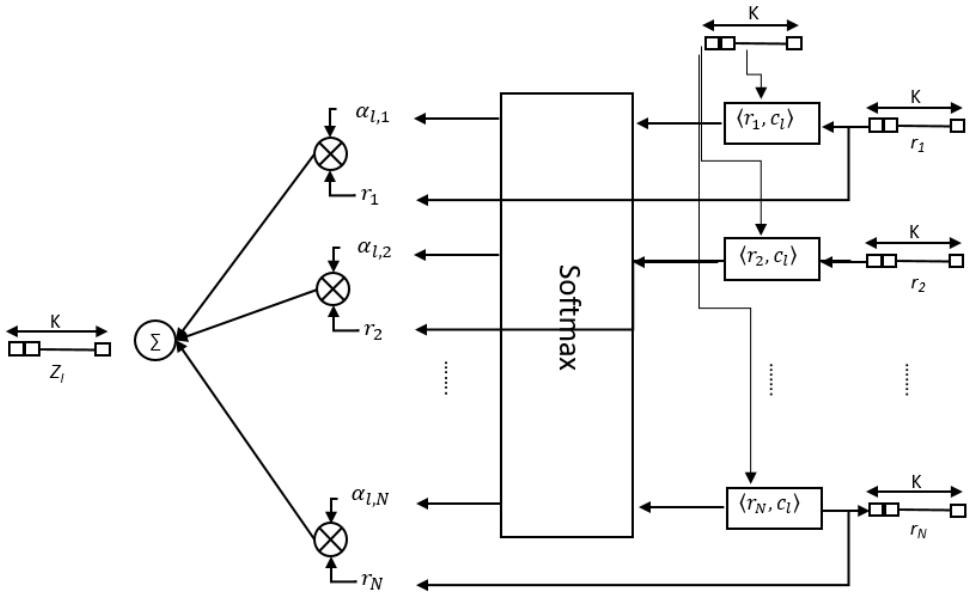

Figure 3: The structure of attention-DNN

Attention-DNN is an efficient approach that measures the similarity of two features with different dimensions. Fig. 3 depicts the structure of the attention-DNN. The input is the received signal **R**. The attention operation is conducted by computing the inner product of each $\boldsymbol{r}_i$ with an attention coefficient $\boldsymbol{c}_l$, i.e., $\langle \boldsymbol{r}_i, \boldsymbol{c}_l \rangle$. This inner product implies the similarity of the signal $\boldsymbol{r}_i$ and the attention coefficient $\boldsymbol{c}_l$, which is normalized by a softmax layer as

$$\alpha_{l,i} = \frac{e^{\langle \boldsymbol{r}_i, \boldsymbol{c}_l \rangle}}{\sum_{n=1}^{N} e^{\langle \boldsymbol{r}_n, \boldsymbol{c}_l \rangle}}, i = 1, \dots, N.$$

Then, the output of the attention-DNN is given by

$$\boldsymbol{z}_l = \sum_{i=1}^{N} \alpha_{l,i} \cdot \boldsymbol{r}_i\,, l = 1, \dots, L.$$

We shall note that the number of attentions are less than the number of received signals, i.e., $L < N$.

Attention-DNN can be employed in the MPA layer for back-propagation. Specifically, each extracted feature vector $\boldsymbol{f}_l$ can be used as an attention coefficient. Then, in the backward sub-iteration, the coefficient $\alpha_{l,i}$ can be given by

$$\alpha_{l,i} = \frac{e^{\langle \boldsymbol{r}_i, \boldsymbol{f}_l \rangle}}{\sum_{n=1}^{N} e^{\langle \boldsymbol{r}_n, \boldsymbol{f}_l \rangle}}, i = 1, \dots N, l = 1, \dots, L.$$

3. Global Tandem Learning

In this section, we propose two algorithms for the AE-based transceiver in the training phase and the inference phase.

3.1. Coarse Learning

The training of the AE-based transceiver includes two parts. One is for the MPA layer in a standalone mode and the other is for the DNNs in the transmitters and receivers by BP. The detailed procedure is summarized in Algorithm 1.

Algorithm 1. The training algorithm

1. **Initialize** the coefficients of the MPA layer $\alpha_{l,i} = \frac{1}{L}$.
2. **Initialize** the batchsize b.
3. **For** step from 1: T **do**
4. In tandem stage 1
5. Sample a batch of training messages $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_b]$.
6. DNN-based transmitter computes $\mathbf{F} = [\boldsymbol{f}_1, \dots, \boldsymbol{f}_L]$ based on the training messages **X**.
7. Compute

$$\boldsymbol{t}_i = \sum_{l=1}^{L} \boldsymbol{\alpha}_{l,i} \cdot \boldsymbol{f}_l \ , i = 1, \dots N,$$

8. Send $\mathbf{T} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{t}_N]$to the DNN-based receiver via communication channels, as
9. $\boldsymbol{r}_i = \boldsymbol{t}_i \cdot \boldsymbol{h}_i + \boldsymbol{n}_i, i = 1, \dots, N.$
10. DNN-based receiver inputs the received signals $\mathbf{R} = [\boldsymbol{r}_1, \boldsymbol{r}_2, \dots, \boldsymbol{r}_N]$ into the DNN and computes the decoded message.
11. Update the transmitter and the receiver by back-propagation.
12. In tandem stage 2
13. **For** iteration from 1: M **do**

Compute $\boldsymbol{r}_i = \sum_{\boldsymbol{i}} \boldsymbol{h}_i \, \alpha_{l,i} \cdot \boldsymbol{f}_l$.

14. Compute

$$||\boldsymbol{r}|| = \sqrt{\boldsymbol{r}_1^2 + \boldsymbol{r}_2^2 + \cdots + \boldsymbol{r}_N^2}.$$

15. Update

$$\boldsymbol{\beta}_{l,i} = \left\langle \frac{\boldsymbol{r}_i}{||\boldsymbol{r}||}, \boldsymbol{f}_l \right\rangle, i = 1, \dots, N, l = 1, \dots, L.$$

16. Update the coefficients of the MPA layer by
17. $\boldsymbol{\alpha}_{l,i} = \text{softmax}(\boldsymbol{\beta}_{l,i}), i = 1, \dots, N, l = 1, \dots, L.$
18. **Endfor**
19. **Endfor**
20. **Output** $\boldsymbol{\alpha}_{l,i}, l = 1, \dots, L; i = 1, \dots, N.$

3.2. Inference Cycle Adaptation

Once the training is finished, the AE-based transceiver is used for inference. In particular, the MPA-layer can help the transceiver adapt to the channel dynamics: when the neurons on the encoder neural

network and decoder neural network are fixed, the coefficient $\alpha_{l,i}$ on the MAC layer could continue to adapt themselves in terms of the current physical channel condition. The detailed procedure is summarized in Algorithm 2.

Algorithm 2. The inference algorithm

1. **Input** : New messages **X**.
2. DNN-based transmitter computes $\mathbf{F} = [\boldsymbol{f}_1, \dots, \boldsymbol{f}_L]$ based on the new message **X**.
3. **For** iteration from 1: M **do**
   Compute
   $\boldsymbol{r}_i = \sum_{l=1}^{L} \boldsymbol{h}_i \, \alpha_{l,i} \cdot \boldsymbol{f}_l \, , , i = 1, \dots N.$
4. Compute
$$||\boldsymbol{r}|| = \sqrt{\boldsymbol{r}_1^2 + \boldsymbol{r}_2^2 + \cdots + \boldsymbol{r}_N^2}.$$
5. Update
$$\boldsymbol{\beta}_{l,i} = \left\langle \frac{r_i}{||\boldsymbol{r}||}, \boldsymbol{f}_l \right\rangle, i = 1, \dots, N, l = 1, \dots, L.$$
6. Update the coefficients of the MPA layer by $\boldsymbol{\alpha}_{l,i} = \text{softmax}(\boldsymbol{\beta}_{l,i}), i = 1, \dots, N, l = 1, \dots, L.$
7. **Endfor**
8. Compute
$$\boldsymbol{t}_i = \sum_{l=1}^{L} \boldsymbol{\alpha}_{l,i} \cdot \boldsymbol{f}_l \, , i = 1, \dots N,$$
9. Compute
   $\boldsymbol{r}_i = \boldsymbol{t}_i \cdot \boldsymbol{h}_i + \boldsymbol{n}_i, i = 1, \dots, N.$
10. DNN-based receiver inputs the received signals$\mathbf{R} = [\boldsymbol{r}_1, \dots, \boldsymbol{r}_N]$ into the DNN and computes the decoded message $\widehat{\mathbf{X}}$

11. **Output $\widehat{\mathbf{X}}$**.

4. Simulations

We consider the following simulation settings. The transmitter sends a block with 256 bits in each time slot through 16QAM modulation scheme without channel coding. The channel gain changes every 200 time slots by a random distortion, which follows $h_{t+1} = h_t + \Delta h_d$, where the random distortion follows $\Delta h_d \sim CN(0, 0.3)$.

The proposed MPA-AE is first pre-trained at the channel condition h_0 with a fixed SNR 10dB to learn all the neurons on the encoding and decoding neural network and the coefficients on the MAP layer. Each time that the channel varies, only the coefficients $\alpha_{l,i}$ at the MPA layer are fine-tuned, while the rest neurons are fixed.

For comparison, we simulate a pre-trained AE without the MPA layer as a baseline, whose neurons are fixed even when the channel has changed, and a retrained AE without MPA layer as another baseline, that is completely retrained whenever the channel varies.

The three frameworks use the same AE structure in which a fully connected neural network with one hidden layer of 16 neurons is used for both the encoder and the decoder and ReLU is used as the activated function for the hidden layer and Adam optimizer is used to train the AE. The only difference of MPA-AE to the rest AE is the inserted MPA layer.

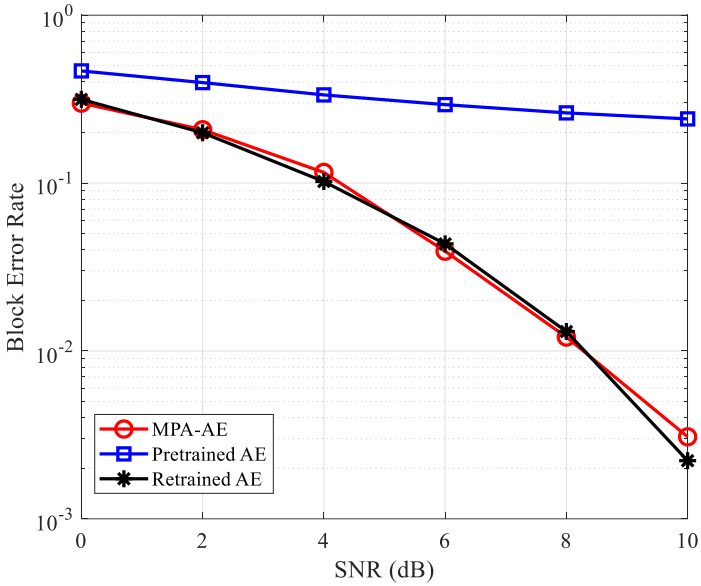

Figure 4: Performance comparison

Fig. 4 shows the block error rate performance of the proposed MPA-AE method together with the two baselines, i.e., pre-trained AE without updating during the channel changing and retrained AE. It can be seen that without updating accordingly, the pre-trained AE failed to work. The proposed MPA-AE and retrained AE show almost the same performance. However, we would like to emphasize that retrained AE took almost 14 more times for updating the whole AE than fine-tuning only the MPA layer. The retrained AE is too time-consuming to be implemented in the real application scenarios.

The improvement of the generalization is attributed to the MPA layer tuned per the dynamic channel condition during the inference stage. In classic AE architecture, all the neuron layers, both transmitter and receiver parts, are frozen. If the channel ran out of distribution in the inference stage (highly likely in wireless system), the AE-based transceiver would suffer from these outliers. The MPA layer provides some resilience against these dynamic changes.

The DNN part is a key component against channel uncertainty. Different SNR represents different white noise levels. In reality, timing-varying dynamic channel is fading channel that includes path loss changes, phrase changes, multiple-path changes and so on. It is hard for a transmitter to perfectly know the current channel conditions. Both receiver's channel feedback and UL/DL reciprocity would introduce some uncertainty or bias about the current ground-true channel condition. The uncertainty includes both shifts and rotations, which are well addressed by the non-linearity of DNNs on both transmitter and receiver and the MPA layer iteration at the transmitter. In this sense, it seems indispensable to couple MPA with DNN to enable the wireless transceiver to profit from the DNN-based autoencoder.

5. Conclusions and Future Directions

In this paper, we proposed a MPA-AE structure and its corresponding algorithms to train the end-to-end transceiver in the scenarios with time-varying channels. The MPA layer inserted between the encoder and decoder of traditional AE can be fine-tuned when the under-going channel changes from the one the transceiver is trained with. Simulations show the superior performance of the proposed method.

The MPA layer could be flexibly incorporated with the AE-based transceiver in many use cases, including single user scenarios and multiuser scenarios. Specifically, in the single user scenarios, the MPA layer can be used to design source coding scheme, high-order modulation scheme, massive MIMO scheme, and pre-distortion scheme, which can well adapt to the time-varying channel conditions. In principle, the DNN layers and MPA layer of a transmitter is to distort the input distribution to match the current channel distortion distribution. The non-linear DNN layers provide a

quick and powerful non-linear distortion, while the linear MPA layer provides a quick and adaptive linear matching.

Moreover, the MPA layer can also be applied in multiuser scenarios for both uplink and downlink MIMO design and Multiple-Access design. More than one MPA-AE could share the same channel. Then, the MPA layer of each becomes autonomous coding design. These research aspects will be considered in future.

Another research direction is to use more complex channel models in the training phase. In particular, the channel models are generated by a DNN with the input of surrounding topological information. In this case, the MPA layer is still effective since the inference DNN of the channel is a non-linear function. However, the DNN-based channel model is often very large. Therefore, how to reduce the DNN model size is an interesting research topic for future investigation.

6. References

[1] Ye, H., Li, G. Y., Juang, B. H. F., & Sivanesan, K. (2018, December). Channel agnostic end-to-end learning based communication systems with conditional GAN. In 2018 IEEE Globecom Workshops (GC Wkshps) (pp. 1-5). IEEE.

[2] Aoudia, F. A., & Hoydis, J. (2018, October). End-to-end learning of communications systems without a channel model. In 2018 52nd Asilomar Conference on Signals, Systems, and Computers (pp. 298-303). IEEE.

[3] Goutay, M., Aoudia, F. A., & Hoydis, J. (2019, June). Deep reinforcement learning autoencoder with noisy feedback. In 2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT) (pp. 1-6). IEEE.

[4] Hu, B., Wang, J., Xu, C., Zhang, G., & Li, R. (2021, September). A Kalman-based Autoencoder Framework for End-to-End Communication Systems. In 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) (pp. 1-6). IEEE.

[5] Cammerer, S., Aoudia, F. A., Dörner, S., Stark, M., Hoydis, J., & Ten Brink, S. (2020). Trainable communication systems: Concepts and prototype. IEEE Transactions on Communications, 68(9), 5489-5503.

[6] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[7] Dörner, S., Cammerer, S., Hoydis, J., & Ten Brink, S. (2017). Deep learning based communication over the air. IEEE Journal of Selected Topics in Signal Processing, 12(1), 132-143.

[8] O'Shea, T. J., Roy, T., & West, N. (2019, February). Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks. In 2019 International Conference on Computing, Networking and Communications (ICNC) (pp. 681-686). IEEE.

[9] Ye, H., Liang, L., Li, G. Y., & Juang, B. H. (2020). Deep learning-based end-to-end wireless communication systems with conditional GANs as unknown channels. IEEE Transactions on Wireless Communications, 19(5), 3133-3143.

[10] Raj, V., & Kalyani, S. (2018). Backpropagating through the air: Deep learning at physical layer without channel models. IEEE Communications Letters, 22(11), 2278-2281.

[11] Aoudia, F. A., & Hoydis, J. (2019). Model-free training of end-to-end communication systems. IEEE Journal on Selected Areas in Communications, 37(11), 2503-2516.