ENABLING SMART FACTORY WITH DEEP RESIDUAL-AIDED GENERATIVE ADVERSARIAL NETWORK: PERFORMANCE ANALYSIS END-TO-END LEARNING OF MACHINE-TO-MACHINE

CONG-DANH HUYNH[1], THANH-KHIET BUI[1], JIRI HAJNYS[2]

[1]Thu Dau Mot University, Thu Dau Mot City, Binh Duong Province, Vietnam

[2]Department of Machining, Assembly and Engineering Technology, Faculty of Mechanical Engineering, VSB—Technical University of Ostrava, 17. listopadu 2172/15, 708 00 Ostrava, Czech Republic

DOI: 10.17973/MMSJ.2023_06_2023031

e-mail to corresponding author: khietbt@tdmu.edu.vn

Improving Machine-to-Machine (M2M) communication is essential for the development of Smart Factory as data can be exchanged and processed more efficiently. Herein this study, we employ the Deep Learning (DL) concepts aimed at improving end-to-end performance (E2E) M2M communication systems. Training the physical layers requires the explicit channel information to be fully known, which can be solved with generative adversarial network (GAN). Nonetheless, due to its deep neural network (DNN) structure, the GAN scheme is subjected to gradient vanishing and over-fitting, two major obstacles that can hinder the training process and limit the performance of the model. As a result, the system is significantly downgraded. To address these issues, we study a method known as Residual-Aided generative adversarial network (RA-GAN) learning scheme, in which the two problems are dealt with respectively by introducing a better propagation mechanism and a regularizer to the loss function. Herein this paper, the system model is described and the two problems are derived analytically. We also analyze the optimal learning scheme (where the channel-agnostic) and a Rayleigh-based learning scheme for comparison study. Through analyzing the block error rate (BLER), we can demonstrate that the RA-GAN approach achieves performance comparable to the optimal scheme, and significantly outperforms the conventional GAN method.

KEYWORDS

Smart factory, machine learning, GAN, machine-to-machine communication

1 INTRODUCTION

Smart Factory is concept where state-of-the-art communication technologies and conventional machinery are fused together. This is realized on the basis of Machine-to-Machine (M2M) communication, where the machines in a factory are interconnected to exchange data, monitor themselves to accomplish their tasks without human intervention. One important aspect of M2M is the need to continuously improve the communication quality and efficiency. This has been facilitated by the significant throughput improvement with 5G; the ability to collect, analyze, and learn from the data as the system operates with Machine Learning (ML) or Deep Learning (DL), etc. [Sun 2018]. The ML or DL approach is well known for its ability to execute the feature engineering process itself. Thus, it has been widely applied in wireless communication, natural language processing, and computer vision (both spoken and written form) [Zhu 2017], [Wu 2017] to perform tasks that are impossible for a robust algorithm. Owing to its data-driven structure, computational complexity can be effectively avoided, yielding in some cases even more accurate results than what is generated by humans [Pfeifer 1989], [He 2015].

Deep Neural Network (DNN) is the most popular structure that is deployed in applications related to DL. Structure-wise, a DNN is made of multiple layers that are stacked together. Each layer contains neurons, each of which is interconnected to all the neurons in the layers right before and after it (fully connected structure). In a deep neural network, there exist three types of layers: the input layer, which has neurons equal to the number of input features; one or more interconnected hidden layers, which perform the computations; and the output layer, which has neurons equal to the number of categories or other outputs that the DNN is expected to provide.

To aid in understanding, let's examine a DNN that follows the typical architecture of a Neural Network (NN), which is composed of an input layer, multiple hidden layers, and an output layer. A connection between two neurons is called a link assigned with a weight value that forms the basis for the learning of the DNN. The weight initialization must be proceeded with caution because the gradient errors are exaggerated as they transfer through the layers, which can cause the gradients to vanish or explode in values. Among the most used initialization methods, there are random approaches (weight matrices are randomly generated using a uniform or the Gaussian distributed function), or the Xavier initialization method (weights are randomly generated and scaled by a scaling function) [Glorot 2010]. The DNN is run with back propagation (BP) algorithm [Rumelhart 1986]. When the output is obtained in the output layer, it is scored and the loss value is calculated. The value as well as the type of loss function are chosen and fine-tuned with regard to the application and experience of the designers. In the next step, backpropagation is performed. During this step, the gradients are obtained by taking the derivative of the loss function with respect to the weights and biases. This process is conducted in reverse, starting from the output layer and moving through the hidden layers to the input layer. Once this is finished, the weights and biases of the model are modified. This cycle is repeated until the desired loss function is minimized.

Although we can apply DL to optimize the physical layers of the communication network, the improvement is not meaningful [O'Shea 2017]. Indeed, optimization of end-to-end (E2E) communication as a complete system is more promising. For example, DL can be used to deal with cases in which the channel models are unknown or the mobility conditions are extreme. This is possible because the E2E learning communication networks, we deploy the transmitter and receiver as DNNs. Unlike the complex conventional algorithm [Yang 2015], the DL-facilitated E2E learning system can solve the modulation and other issues related to signal processing using simple mathematic operations between DNN layers [Dai 2020]. In addition, we can utilize DL to design transmit signals facilitating straightforward algorithms to detect symbols for channel and system models. To accomplish this task, we must be able to obtain the gradient of the instantaneous channel transfer function (the channel model) [Ye 2018], whose identification is

hindered in most cases in practice. More often than not, a channel consists of a transceiver, which is non-differentiable, thus, preventing the gradient-based algorithm to execute its task. To address the problems with unknown channels in E2E learning, various ML techniques have been investigated in [Dorne 2018], [Raj 2018], [Aoudia 2019], [Ye 2018], [Aoudia 2018], [Bhattacharya 2021],[Ye 2020], [Dorner 2020] to reduce the burden on the hardware requirements to obtain the full channel information. Specifically, the authors in [Dorne 2018] proposed a training model with two phases. During the first phase, to mimic the channel condition in practice, a stochastic channel model was utilized for system training. The second step of the process involves using a supervised learning approach to adjust the receiver and compensate for any discrepancies between the stochastic and actual channels. It is noteworthy that during the second phase, the transmitter cannot be adjusted, which could downgrade the system performance.

Having acknowledged the pros and cons of the two-phase E2E learning scheme, several articles have attempted to refine it by alternative training the transmitter and receiver for better system performance. In general, these improved schemes can be classified into two types in which the receiver is aided or the channel is imitated. In the receiver-aided scheme, feedback from the receiver will be sent to the transmitter to assist the training process [Raj 2018], [Aoudia 2019]. Regarding the channel imitation scheme, the real received signal is imitated, that is, approximately generated to be closest to the real signal using a generative adversarial network (GAN) as in [Ye 2018]. A typical GAN is constructed of one generator and one discriminator, both of which are under the form of multi-layer DNNs. The system works so that the generator will base on how the real received signal is distributed to generate corresponding synthetic received signal to train the source. At the same time, the generator will be trained by the discriminator to match the real signal with the synthetic signal it produces. This process enables us to tackle the challenge of the unknown channel, making it possible to calculate the source (transmitter) gradient with ease. Moreover, studies have shown that GAN can be utilized in the case of arbitrary channels and it can the transceiver's hardware could be simplified [Ye 2018], [Aoudia 2018], [Ye 2020], [Dorner 2020].

Nevertheless, GAN training in most cases is unstable owing to the issues with mode collapse, network convergence, gradient vanishing, and overfitting [Creswell 2018], [Arjovsky 2017]. Moreover, GAN does not perform well when it is tasked with capturing a particular distribution [Wu 2017]. Thus, the authors in [Gulrajani 2017] proposed two GAN variants namely the Wasserstein GAN with gradient penalty (WGAN-GP) to reinforce the training and conditional GAN (C-GAN) to capture better the channel distribution. However, a major difficulty with this approach is with the iteration numbers since the generated error in most cases is small, i.e., the optimal number of iterations remains unknown. This problem has been countered by implementing regularization strategies such as early stopping [Srivastava 2014] and dropout [Prechelt 1998].

Indeed, GAN has proven itself to be beneficial for wireless networks and has been utilized for many interesting applications in recent years [Bin 2021], [Hu 2021], [Balevi 2021], [Shi 2021], [Leng 2022]. Specifically, a novel GAN-based model namely GAN-CDG was proposed to gather compressive data, which can achieve a compression ratio of up to 16 while ensuring highly accurate recovery, more than 30dB [Jiang 2011]. In wireless networks that employ a training approach, GAN helps to boost the channel estimation without the need to transmit lengthy training sequences [Hu 2021] and can even outperform conventional estimators that are used for digital receivers [Balevi 2021]. In the same paper, a lookup-table-based strategy was investigated so as to mitigate the overfitting problem when training the GAN model. From another perspective, authors in [Shi 2021] used GAN to simulate proofing attacks on single-input and single-output (SISO) as well as multiple-input and multiple-output (MIMO), proving that this approach is a potential threat to the modern wireless networks. Finally, the paper [Leng 2022] combines the GAN and 5G to realize cell load estimation using the in-air information obtained from the terminals. This combination significantly improves network throughput, load estimation accuracy, and reduces network delay.

However, it is worth mentioning that GAN system is subjected to its chronic problems namely the gradient vanishing and overfitting [Jiang 2011], owing to the DNN-structured generator and discriminator. Moreover, due to the fact that the channel model in the DL application is usually unknown, conventional BP algorithm cannot calculate the gradients. In an attempt to solve these problems, we investigate in this we conduct a detailed investigation of a scheme known as residual aided GAN (RA-GAN), as follows:

- The proposed scheme employs a residual neural network (Resnet) to reconstruct the generator. Specifically, a link between the generator's input and output is designed, which results in an extra gradient to diminish the problem with gradient vanishing.
- The addition of a regularizer to the loss function was implemented to limit the GAN scheme's representation capacity. By this, we are able to overcome the overfitting problem.
- Full knowledge of complex channel model is irrelevant because the E2E network can perform well in terms of BLER in Additive White Gaussian Noise (AWGN) and Rayleigh fading channel solely by using DL to replicate the real received signals.
- From the numerical simulation, we can demonstrate that the RA-GAN produces a better synthetic received signal than the GAN scheme. The improvement is shown in the block error rate (BLER) calculated for the theoretical model and for a chosen dataset.

2 SYSTEM MODEL

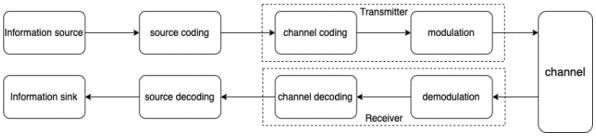

FIGURE 1. The diagram of E2E learning Machine-to-Machine communication system

Fig. 1 illustrates the diagram of the E2E learning M2M communication network considered in this study. Technique-wise, each of the modules has reached its mature state. However, as they are designed for different objectives, finding the optimal setup for a system where those modules are assembled is almost an impossible task with traditional optimization approaches. This is why researchers have focused on implementing DNN in modern communication systems to produce the so-called E2E learning M2M communication network. However, a drawback of this E2E scheme is that we must be able to monitor the channel transfer function or instantaneous channel state information (CSI). This is because

the DNN backpropagation process is blocked when the system does not know the channel, causing failures in E2E learning. Indeed, the channel transfer function can be arbitrarily chosen, but any choice would potentially introduce bias into the learning system. On the other hand, CSI is not always available and easy to record because of the noises from the environment [Glorot 2010]. As a solution to this, we can use conditional GAN to learn the channel output distribution [Yang 2015]. Conditional GAN is trained iteratively and we can minimize the E2E loss in a supervised manner. For conventional wireless communication, the system consists of several modules e.g., channel, channel encoder, channel decoder, modulator, demodulator, source encoder, source decoder, etc. However, for an E2E learning communication system, it is sufficient to include three main components, i.e., transmitter $\boldsymbol{T}$ and receiver $\boldsymbol{R}$ in both ends, and the intermediate channel. $\boldsymbol{T}$ and $\boldsymbol{R}$ are structured as DNNs with respectively two weights Θ_{T} and Θ_{R} that are trainable [O'Shea 2017]. The applied DL architecture can be observed in Fig. 2.

During operation, $\boldsymbol{T}$ receives an input s and maps it to a one-hot vector 1_k. The one-hot vector is with K-dimension and its components are taken from set K. Notably, the only k-th element is assigned with value 1, while the remainders K-dimension, K -1 elements are 0. $\boldsymbol{T}$ then plays a role of a function $f_{\Theta_T}: \mathrm{K} \mapsto C^n$ to map the 1_{k} to the signal $\mathrm{s} \in \mathrm{C}^{\mathrm{n}}$ before sending it via n channels. Meanwhile, $\boldsymbol{R}$ would act as a function $\mathrm{f}_{\Theta_{\mathrm{R}}}: \mathrm{C}^{\mathrm{n}} \mapsto \left\{ \boldsymbol{\delta} \in \mathrm{R}_{+}^{\mathrm{K}} \mid \sum_{\mathrm{j}=1}^{\mathrm{K}} \boldsymbol{\delta}_{\mathrm{j}} = 1 \right\}$ to represent the signal $\mathrm{y} \in \mathrm{C}^{\mathrm{n}}$, it received to a probability distribution $\delta \in \mathrm{R}_{+}^{\mathrm{K}}$.

Following that, we receive a final choice $\hat{\boldsymbol{s}}$ with respect to one-hot vector $\mathbf{1}_{\hat{k}}$, where $\hat{k}$ is maximum in the probability vector $\boldsymbol{\delta}$. Generally, $\boldsymbol{T}$'s hardware would constrain the transmitted signal $\boldsymbol{s}$ in terms of power with $|\boldsymbol{s}|^2 = 1$. From the received signal in the AWGN channel, we can calculate the signal received at each time slot as $\mathrm{y} = \tau \boldsymbol{s} + \nu, where$ the channel coefficients considered in the block fading channels are assumed to vary freely from a time slot to the other. Given that there is no generality loss, we can examine the slow fading channel, marked as conditional probability $\boldsymbol{\delta}_{\tau}(\boldsymbol{y}|\boldsymbol{s})$, where we have the channel remains the same throughout the n time slots, $\tau = \tau_j, j \in 1,2,..,n$. The Gaussian noise is denoted as $\nu \in C^n$.

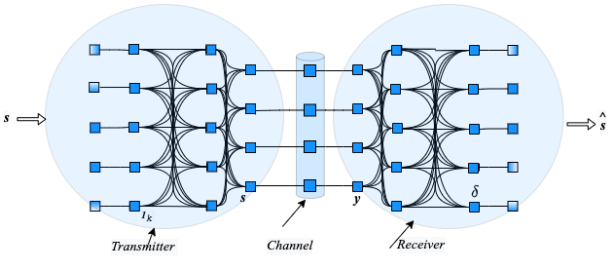

FIGURE 2. DL applied on the architecture of E2E learning communication machine-to-machine system.

Remarkably, henceforward, we assume that an ideal communication system is with flawless timing, carrier-phase and frequency synchronization. $\boldsymbol{R}$'s task is to reproduce the estimated message $\hat{\boldsymbol{s}}$ based on the original $\boldsymbol{s}$ message, each of which is a bit sequence of length $d = log_2(K)$. The communication rate is therefore calculated as d/n (bits/channel use). According to [O'Shea 2017], [Ye 2018], and [Jiang 2011], BLER is sufficient for the evaluation of the proposed DNN-based structure. When the system transmits different messages, the block error rate (BLER) is calculated as

$$P_e = \frac{1}{K} \sum_{\boldsymbol{s}} Pr(\hat{\boldsymbol{s}} \neq \boldsymbol{s} \mid \boldsymbol{s}). \tag{1}$$

To obtain the optimal weights $\boldsymbol{\Theta}_{\mathbf{T}^*}$ and $\boldsymbol{\Theta}_{\mathbf{R}^*}$, for $\boldsymbol{T}$ and $\boldsymbol{R}$, we have to perform the training on the DNN $\boldsymbol{T}$ and DNN $\boldsymbol{R}$ to iteratively update the weights $\Theta_{\mathbf{T}}$ and Θ_{R}. It is assumed that $\boldsymbol{R}$ knows the transmitted information. This information is generatable with the help of random seed that is in $\boldsymbol{T}$ and $\boldsymbol{R}$. Accordingly, we use a loss function to calculate the error between the $\boldsymbol{T}$'s one-hot vector $\mathbf{1}_{\mathbf{k}}$ and the probability distribution that is $\boldsymbol{\delta}$ recovered [Aoudia 2019]. The channel has the training set of $\Gamma = \{\tau^{(1)}, \tau^{(2)}, .., \tau^{(W)}$ with W being the size of the batch. As a result, the loss function can be written as

$$\mathcal{L}(\Theta_{\mathrm{T}}, \Theta_{\mathrm{R}}, \Gamma) \approx \frac{1}{\mathrm{W}} \sum_{\mathrm{j}=1}^{\mathrm{W}} \mathrm{l}\left(\boldsymbol{\delta}^{(\mathrm{j})}, 1_{\mathrm{k}^{(\mathrm{j})}}\right), \tag{2}$$

where we have the received signal $\boldsymbol{y}^{(j)}$, the training sample $\mathbf{1}_{\mathbf{k}^{(\mathrm{j})}}$, and the probability vector $\boldsymbol{\delta}^{(j)}$ at order j-th. According to [Goodfellow 2016], the distance from $\mathbf{1}_{\mathbf{k}}$ to $\boldsymbol{\delta}$ can be calculated with the help of cross-entropy (CE) as $l(\boldsymbol{\delta}, \mathbf{1}_{\mathbf{k}}) = -\sum_{j=1}^{M} (\mathbf{1}_{\mathbf{k}})_j \ln(\boldsymbol{\delta}_j) + (1 - (\mathbf{1}_{\mathbf{k}})_j) \ln(1 - \boldsymbol{\delta}_j)$.

As aforementioned in the Introduction, in the BP algorithm, it is necessary to calculate the gradient of the loss function (2) to update the weights Θ_{T} and Θ_{R}. However, it is impossible to calculate the gradient $\nabla_{\Theta_{\mathrm{T}}} \tilde{\mathcal{L}}$ with respect to Θ_{T} because the channel is unknown [Dorne 2018]. In other words, the channel-agnostic τ has blocked the BP process with

$$\nabla_{\boldsymbol{\Theta}_{\mathbf{T}}} \tilde{\mathcal{L}}(\Theta_{\mathrm{T}}) = \frac{1}{W} \sum_{j=1}^{W} \tau^{(j)} \frac{\partial\, l}{\partial \boldsymbol{f}_{\boldsymbol{\theta}_R}} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}_R}}{\partial \mathbf{y}^{(\mathrm{j})}} \mathbf{I}_{\mathbf{n}} \nabla_{\boldsymbol{\Theta}_{\mathbf{T}} \mathbf{f}_{\boldsymbol{\Theta}_{\mathbf{T}}}} \left(\mathbf{1}_{\mathbf{k}^{(\mathrm{j})}}\right). \tag{3}$$

Nonetheless, it is noteworthy that only Θ_R in (2) can be updated with the below gradient

$$\nabla_{\Theta_R} \tilde{\mathcal{L}}(\Theta_R) = \frac{1}{W} \sum_{j=1}^{W} \nabla_{\Theta_R} l\left(f_{\Theta_R}\left(y^{(j)}\right), 1_{k^{(j)}}\right), \tag{4}$$

where we have the loss function approximately derived from (3) as $\tilde{\mathcal{L}}$.

To update Θ_{T}, a surrogate gradient must be employed with the help of GAN learning scheme [Aoudia 2019]. We present the generator $\boldsymbol{G}$ and discriminator $\boldsymbol{D}$, that are executed as DNNs in the GAN scheme. The two possess weights Θ_G and Θ_D that can be updated through training. Based on the Gaussian distribution, $\boldsymbol{G}: f_{\theta_G}: C^{2n} \mapsto C^n$ would generate synthetic received signal $\tilde{y}$ after considering the source transfer $s \in C^n$ with random noise $\chi \in C^n$. Here, the random noise χ is built-in for simplification. The task of χ is to make $\boldsymbol{G}$ produce different output for any given input s. As a result, $\boldsymbol{G}$ could generate a distribution that resembles the true received signal distribution. Meanwhile, $\boldsymbol{D}: f_{\theta_D}: C^n \mapsto (0,1)$ would be deployed to train $\boldsymbol{G}$, enabling $\boldsymbol{G}$ to imitate the real received signal. $\boldsymbol{D}$ is used to differentiate the real from the synthetic received signals.

On the one hand, when the real received signal distribution $\delta_{\tau}(y \mid s) \to 1$ passes through $\boldsymbol{D}$. On the other hand, for the synthetic received signal distribution $p_{\tilde{\tau}}(\tilde{y} \mid s) \to 0$. For the imitation of real received signal, the output from $\boldsymbol{G}$, $\tilde{y}$ has to make the output in $\boldsymbol{D}, f_{\Theta_D}(\tilde{y}) \to 1$. As previously discussed on how the GAN scheme works, we can update the weights of $\boldsymbol{D}$, θ_D with regard to the actual received signal's loss function y as

$$\tilde{\mathcal{L}}(\Theta_D) = \frac{1}{W} \sum_{j=1}^{W} l\left(\boldsymbol{f}_{\boldsymbol{\Theta}_D}\left(\boldsymbol{y}^{(j)}\right), 1\right) l\left\{\boldsymbol{f}_{\boldsymbol{\Theta}_D} + \left(\left(\tilde{\boldsymbol{y}}^{(j)}\right), 0\right)\right\}. \tag{5}$$

Accordingly, the weights of $\boldsymbol{G}$, Θ_G will be used to update the loss function of the synthetic received signal $\tilde{y}$ as

$$\tilde{\mathcal{L}}(\theta_G) = \frac{1}{W}\sum_{i=1}^{W} l\left(f_{\theta_D}\left(f_{\theta_G}(s^{(j)}, \chi^{(j)})\right), 1\right). \quad (6)$$

Subsequently, we can calculate the gradients using $\nabla_{\theta_G}\tilde{\mathcal{L}}(\theta_G)$ and $\nabla_{\theta_D}\tilde{\mathcal{L}}(\theta_D)$. The loss function (5) and (6) reduced with the use of the Adam gradient descent technique [Kingma 2015]. As it is able to train $\boldsymbol{G}$ so that it can generate the synthetic received signal based on the real received signal, we can employ gradient surrogate, which is computed to be similar to the gradient (4) as

$$\nabla_{\boldsymbol{\Theta_T}}\tilde{\mathcal{L}}(\boldsymbol{\Theta_T}) = \frac{1}{W}\sum_{j=1}^{W} \frac{\partial l}{\partial \boldsymbol{\delta}^{(j)}}\frac{\partial \boldsymbol{\delta}^{(j)}}{\partial \tilde{\boldsymbol{y}}^{(j)}}\frac{\partial \tilde{\boldsymbol{y}}^{(j)}}{\partial \boldsymbol{s}^{(j)}}\nabla_{\boldsymbol{\Theta_{Tf_{\Theta_T}}}}\left(\mathbf{1}_{\boldsymbol{k}^{(j)}}\right). \quad (7)$$

It should be noted that training of $\boldsymbol{T}$, $\boldsymbol{R}$, $\boldsymbol{G}$, and $\boldsymbol{D}$ have different objectives, thus, each of the module is trained separately. In other words, as one module is trained, the weights of other modules are kept constant. Nevertheless, it has been acknowledged that the unstable training would vastly downgrade the GAN performance [Aoudia 2018]. Particularly, when the data is passed through multiple layers of the generator, the gradient will be vanished. Additionally, because a large number of weights are involved in the iterative training process, an overfitting issue would occur. In view of this, we investigate the RA-GAN learning scheme to address the two problems with detail discussed below.

3 RA-GAN SCHEME IN E2E LEARNING COMMUNICATION SYSTEM

Because there exists the channel-agnostic, training the transmitter is a difficult mission. We establish a surrogate gradient during the GAN scheme's training procedure in (7) to update the weights of the transmitter. Nevertheless, There is a substantial disparity between the synthetic signal distribution and the real signal distribution $\delta_{\tilde{\tau}}(\tilde{y} \mid s)$ and the real received one $\delta_{\tau}(y \mid s)$. Due to the nature of the DNNs we use in $\boldsymbol{G}$, there are two explanations for this: gradient vanishing and over-fitting problems. During operation, $\boldsymbol{T}$ receives an input s and maps it to a one-hot vector 1_k. The one-hot vector is with K-dimension and its components are taken from set K. Notably, the only k-th element is assigned with value 1, while the remainders K-dimension, K-1 elements are 0. $\boldsymbol{T}$ then plays a role of a function $f_{\Theta_T}: \mathrm{K} \mapsto C^n$ to map the 1_{k} to the signal $\mathrm{s} \in \mathrm{C^n}$ before sending it via n channels. Meanwhile, $\boldsymbol{R}$ would act as a function $\mathrm{f}_{\Theta_R}: \mathrm{C^n} \mapsto \left\{\delta \in \mathrm{R_+^K} \mid \sum_{\mathrm{j}=1}^{\mathrm{K}} \delta_{\mathrm{j}} = 1\right\}$ to represent the signal $\mathrm{y} \in \mathrm{C^n}$ it received to a probability distribution $\delta \in \mathrm{R_+^K}$. This is the inspiration for us to investigate the so-called RA-GAN scheme.

3.1 *Gradient vanishing mitigation with residual learning*

In BP algorithm, a variable is input and multiplied by the partial derivatives while passing through the layers. As the number of layers increases and the partial derivative value approaches 0, the resulted gradient will vanish. This is so-called gradient vanishing and is an obstacle to the DNN generator training. To overcome this, in the RA-GAN scheme, a connection for the in-out layers of the generator has been constructed based on the residual learning [He 2016] so that the input does not have to pass through the intermediate layers.

The way we train the RA-GAN scheme is equivalent to the way we train the conventional GAN [Aoudia 2019]. Particularly, $\boldsymbol{R}$ is under supervised learning problem to acquire the loss function. In the context of reinforcement learning, $\boldsymbol{T}$ learns how to replicate the environment with the loss function from $\boldsymbol{R}$ and collect positive feedbacks as rewards. The results of simulation have shown that this approach performs comparatively well in comparison to the E2E supervised learning DNN network where the channel model is known.

In particular, we use $\tilde{\mathrm{y}}$ and $\boldsymbol{y}$ for training $\boldsymbol{D}$ and $\tilde{\boldsymbol{y}}$ to train $\boldsymbol{G}$ of the proposed RA-GAN. Followingly, $\boldsymbol{R}$ is trained only with $\boldsymbol{y}$, while $\boldsymbol{T}$ is trained with $\tilde{\boldsymbol{y}}$. The training of $\boldsymbol{T}$ and $\boldsymbol{R}$ is monitored respectively by the loss functions $l\left(\boldsymbol{f}_{\Theta_R}(\tilde{\boldsymbol{y}}), \mathbf{1_k}\right)$ and $l\left(\boldsymbol{f}_{\boldsymbol{\Theta_R}}(\boldsymbol{y}), \mathbf{1_k}\right)$. With regard to the residual generator, we can rewrite the residual generating function $f_{\Theta_{GR}}: C^n \mapsto C^n$ as $f_{\Theta_{GR}}(s) = \tilde{\boldsymbol{y}} - \mathrm{s} = f_{\Theta_{GR}}(s) - s$, where we have the residual generator $f_{\Theta_{GR}}(s)$ set up with condition s to identify the distinction between the produced signal $\tilde{y}$ and transmitted signal s.

Following, the gradient that we use to update weights of $\boldsymbol{T}$ can be computed from $\boldsymbol{R}$ of the ***T-R*** link and the gradient of this link is calculated as

$$\begin{aligned}\nabla_{\boldsymbol{\Theta_T}}\tilde{\mathcal{L}}(\boldsymbol{\Theta_T}) &= \frac{1}{W}\sum_{j=1}^{W}\frac{\partial l}{\partial \boldsymbol{f}_{\boldsymbol{\Theta_R}}}\frac{\partial \boldsymbol{f}_{\boldsymbol{\Theta_R}}}{\partial \boldsymbol{f}_{\boldsymbol{\Theta_G}}}\frac{\partial \boldsymbol{f}_{\boldsymbol{\Theta_{G^R}}}}{\partial \boldsymbol{f}_{\boldsymbol{\Theta_T}}}\nabla_{\boldsymbol{\Theta_{Tf_{\Theta_T}}}}\left(\mathbf{1}_{\boldsymbol{k}^{(j)}}\right) \\ &\quad + \frac{1}{W}\sum_{j=1}^{W}\frac{\partial l}{\partial \boldsymbol{f}_{\boldsymbol{\Theta_R}}}\frac{\partial \boldsymbol{f}_{\boldsymbol{\Theta_R}}}{\partial \boldsymbol{f}_{\boldsymbol{\Theta_G}}}\nabla_{\boldsymbol{\Theta_{Tf_{\Theta_T}}}}\left(\mathbf{1}_{\boldsymbol{k}^{(j)}}\right) \\ &= \frac{1}{W}\sum_{j=1}^{W}\frac{\partial l}{\partial \boldsymbol{\delta}^{(j)}}\frac{\partial \boldsymbol{\delta}^{(j)}}{\partial \tilde{\boldsymbol{y}}^{(j)}}\frac{\partial \boldsymbol{f}_{\boldsymbol{\Theta_{G^R}}}^{(j)}}{\partial \boldsymbol{s}^{(j)}}\nabla_{\boldsymbol{\Theta_{Tf_{\Theta_T}}}}\left(\mathbf{1}_{\boldsymbol{k}^{(j)}}\right) \\ &\quad + \frac{1}{W}\sum_{j=1}^{W}\frac{\partial l}{\partial \boldsymbol{\delta}^{(j)}}\frac{\partial \boldsymbol{\delta}^{(j)}}{\partial \tilde{\boldsymbol{y}}^{(j)}}\nabla_{\boldsymbol{\Theta_{Tf_{\Theta_T}}}}\left(\mathbf{1}_{\boldsymbol{k}^{(j)}}\right). \end{aligned} \quad (8)$$

Remarkably, (8) is the gradient we use to update the weights of $\boldsymbol{T}$ through training. To optimal gradient is obtained when the derivative $\frac{\partial f_{\Theta_{GR}}}{\partial f_{\Theta_T}}$ closed to value $\tau(j) - 1$.

3.2 *Overfitting mitigation with a regularizer*

When training with GANs, overfitting can occur due to the high number of trainable DNN weights in the generator and discriminator. To prevent this, a regularizer is added to improve the generation of synthetic signals to better match real signals. As can be observed in Fig. 3, the loss function is reconstructed after each iteration. Minimizing the newly created loss function is the algorithm's goal in (6), thus, making the weights $\boldsymbol{\Theta}_i$, $\mathrm{i} \in \{\boldsymbol{R}, \boldsymbol{T}, \boldsymbol{G}, \boldsymbol{D}\}$ close to 0. This is because a large value of weights will signify the small noise from the input data, which severely downgrades the output while small weights do not. Therefore, working with small weights is simpler and the problem of overfitting can be avoided more effectively [Goodfellow 2016]. As a result of this employment, we have a residual $\boldsymbol{G}$ that is free of overfitting problems and performs well with different channel data.

Analytically, when a weight penalty item $\Omega(\boldsymbol{\Theta})$ is added to the loss function so that we can constrain the representation ability, we obtain

$$\widehat{\boldsymbol{H}}(\boldsymbol{\Theta_i}) = \tilde{\mathcal{L}}(\boldsymbol{\Theta_i}) + \varphi\Omega(\boldsymbol{\Theta_i}), i \in \{\boldsymbol{R}, \boldsymbol{T}, \boldsymbol{G}, \boldsymbol{D}\}, \quad (9)$$

where we have the modified and the original loss functions denoted respectively as $\widehat{\boldsymbol{H}}(\boldsymbol{\Theta_i})$ and $\tilde{\mathcal{L}}(\boldsymbol{\Theta}_i)$. The hyper-parameter φ is deployed to balance the $\tilde{\mathcal{L}}(\boldsymbol{\Theta}_i)$ and the penalty item. It should be noted that $\boldsymbol{D}$ is simplified using the regularization method. Besides, $\boldsymbol{D}$ is set with relatively low learning rate so that it will converge slowly. This setting is necessary to avoid the gradient vanishing by maintaining a matching convergence rate between $\boldsymbol{G}$ and $\boldsymbol{D}$.

The E2E training process based on RA-GAN can be observed below in Fig. 3. The inputs are weights $\boldsymbol{\Theta}_i, \mathrm{i} \in \{\boldsymbol{R}, \boldsymbol{T}, \boldsymbol{G}, \boldsymbol{D}\}$, number of data used for training N_{train} and iteration number Epoch. The outputs are the updated weights $\boldsymbol{\Theta}_{\boldsymbol{T}}$ and $\boldsymbol{\Theta}_{\boldsymbol{R}}$.

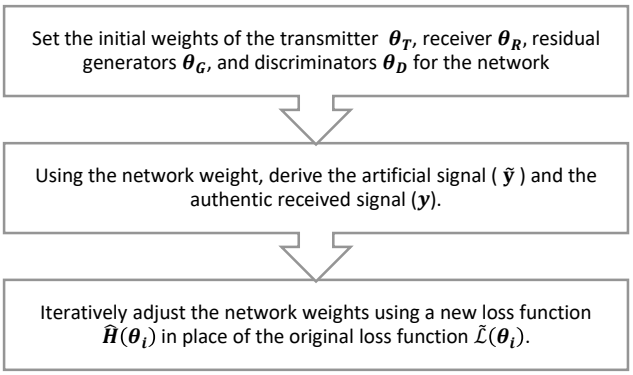

Figure 3. Algorithm RA-GAN based E2E training for M2M system.

4 RESULT AND DISCUSSION

This section presents the performance analysis of the RA-GAN scheme by means of BLER. Specifically, two performance comparison studies between the RA-GAN scheme and the RL-GAN scheme in the AWGN and the Rayleigh fading channel [Aoudia 2019]. The results in general show that the proposed model performs relatively well in comparison with the case when the channel is known over the AWGN and Rayleigh fading channel. The extra gradients are generated with the RA-GAN counteract the gradient vanishing problem and the regularizer added to the loss function has helped the system to overcome the overfitting.

As a reference for the comparison study, an optimal scheme with known channel was employed. In the optimal case, it was assumed that the transmitter knew the real channel, so that the gradient $\nabla_{\Theta_T}\tilde{\mathcal{L}}$ can be used for its training (8). Additionally, we compared the generation ability of RA-GAN with the Reinforcement learning (RL) scheme. The system was trained with parameters as : $E_b/\mathrm{N0} = 10(dB), K = 16, W = 320, Epoch = 200, N\_train = 10^6, n = 7$ *and* $\varphi = 3$ in the dataset used for validation consisted of 10^6 one-hot vectors [Jiang 2011], and learning rate for training both the $\boldsymbol{T}$ and $\boldsymbol{R}$ is 0.001. In which $\boldsymbol{R}$ and $\boldsymbol{D}$ under AWGN and Rayleigh fading channel model are trained with different learning rate, i.e., respectively, 0.0005 and 0.0001.

The AWGN channel's and Rayleigh fading channel's output is respectively described by $\mathrm{y} = \boldsymbol{s} + \nu$, and $\mathrm{y} = \tau\,\boldsymbol{s} + \nu$, where we have the transmitted signal s, Gaussian noise ν, and coefficient $\tau \sim \mathcal{CN}(0,1)$. The BLER results for the AWGN (see Fig. 4) and Rayleigh (see Fig. 5) are with different scales because if the same scale is employed, the gains on one fading channel will be relatively insignificant in comparison with the other. It should be noted that we use the Nash equilibrium as the stopping criteria for the training of GAN. In particular, when $\boldsymbol{D}$ outputs approximate value of 0.5, the training is stopped because the system cannot recognize anymore the synthetic signal from the real received signal [Sustika 2020].

Fig. 5 presents the BLER performance of the three schemes as we vary the $E_b/N0$. In general, the RA-GAN scheme performs better than the RL scheme. In particular, for $E_b/N0$ below 12(dB), we can observe that the three curves are almost identical. However, as we increase the $E_b/N0$, the gaps between them grow as well larger. This is because the representation ability of the scheme has been restrained by the regularizer. At BLER = 0.1, the performance The performance difference between the RA-GAN scheme and the RL scheme is 3(dB).

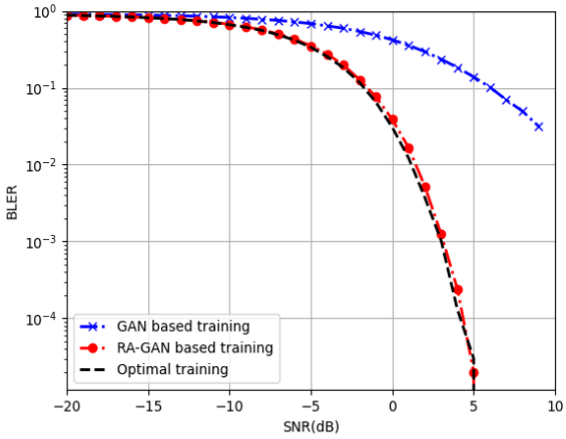

Figure 4. Performance comparison with regard to BLER in AWGN channel.

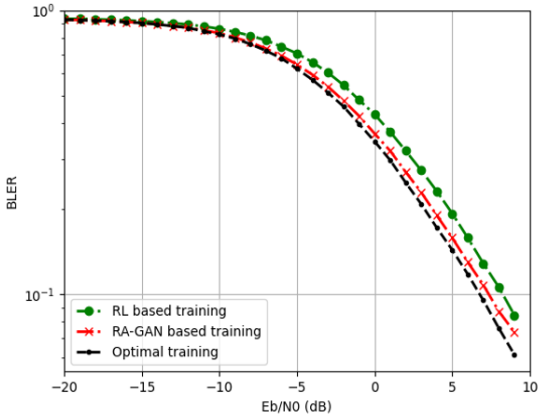

Figure 5. Performance comparison with regard to BLER in Rayleigh fading.

Alternatively, by reducing the hyperparameter φ in (9), the ideal BLER system performance is achievable, albeit at a reduced BLER performance when the $E_b/N0$ value is low. It's important to note that during the training process, there may be some negative outcomes due to training randomness, but these can be recovered in the next epoch. As a result, it can be concluded that the RA-GAN-based training method is capable of producing signals that more closely resemble the actual received signals. In other words, the trained residual $\boldsymbol{G}$ is superior to the conventional *G* when it comes to signal generation performance.

5 CONCLUSIONS

In conclusion, an E2E learning scheme for channel-agnostic communication system so-called RA-GAN has been proposed and analyzed in this paper. The RA-GAN scheme was improved from the GAN scheme by implementing the gradient vanishing and over-fitting issues using the residual learning approach. During the paper, the formulation of the system and the two problems were derived. To assess the RA-GAN scheme, we compared its performance in terms of the BLER with the optimal scheme and the RL scheme. Results from the simulation indicated that the RA-GAN outperformed the RL and was almost

optimal. Besides, the derived formulations can be used for further comparison studies with other DL methods. This study has focused on the two most popular fading channel models, the fading channel used by Rayleigh and the AWGN. In the framework of Smart Factory, this is very essential since the technology helps to scope with emerging higher data payloads, and reduce the burden of sophisticated hardware and interactions between machines in modern machinery. This work can be expanded to fully automated machining workshop or 3D printing plants, where the M2M system requires effective communication to perform sophisticated machine programming, controlling loading and unloading parts, monitoring manufacturing parameters, etc.

ACKNOWLEDGMENTS

This work was supported by the project Innovative and Additive Manufacturing Technology —New Technological Solutions for the 3D Printing of Metals and Composite Materials(CZ.02.1.01/0.0/0.0/17_049/0008407) and by the European Regional Development Fund in the Re-search Centre of Advanced Mechatronic Systems project, CZ.02.1.01/0.0/0.0/16_019/0000867 within the Operational Programme Research, Development, and Education and the project SP2022/60 Ap-plied Research in the Area of Machines and Process Control supported by the Ministry of Education,Youth, and Sports.

REFERENCES

[Sun 2018] Sun, H., et al. Learning to optimize: Training deep neural networks for interference management. IEEE Transactions on Signal Processing, Vol. 66, No. 20, pp. 5438–5453, 2018.

[Zhu 2017] Zhu, J.-Y. et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proc. of the IEEE international conference on computer vision, 2017, pp. 2223–2232.

[Wu 2017] Wu, B. et al. An end-to-end deep learning approach to simultaneous speech dereverberation and acoustic modeling for robust speech recognition. IEEE Journal of Selected Topics in Signal Processing, Vol. 11, No. 8, pp. 1289–1300, 2017

[Pfeifer 1989] Pfeifer, R. et al. Generalization and network design strategies. Connectionism in perspective, pp. 143–155, 1989.

[He 2015] He, K. Delving deep into rectifiers: Surpassing human-level performance on image net classification. In: Proc. of IEEE Int. Conf. Computer Vision, 2015, pp. 1026–1034.

[Glorot 2010] Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In: Proc. of 13th Int. Conf. Artif. Intell. Stat., Y. W. Teh and M. Titterington, Eds., 2010, pp. 249–256.

[Rumelhart 1986] Rumelhart, D. E. et al. Parallel distributed processing: Explorations in the microstructure of cognition. vol.1. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362

[O'Shea 2017] O'Shea T. & Hoydis, J. An introduction to deep learning for the physical layer. IEEE Trans. Cogn. Commun. Netw., Vol. 3, No. 4, pp. 563–575, Oct. 2017.

[Yang 2015] Yang, S. and Hanzo, L. Fifty years of MIMO detection: The road to large-scale MIMOs. IEEE Commun. Surveys Tuts, vol. 17, no. 4, pp. 1941–1988, 2015.

[Dai 2020] Dai, L. et al. Deep learning for wireless communications: An emerging interdisciplinary paradigm. IEEE Wireless Commun., Vol. 27, No. 4, pp. 133–139, Aug. 2020.

[Ye 2018] Ye, H. et al. Power of deep learning for channel estimation and signal detection in OFDM systems. IEEE Wireless Communications Letters, vol. 7, no. 1, pp. 114-117, 2018.

[Dorne 2018] Dorne, S. et al. Deep learning based communication over the air. IEEE J. Sel. Topics Signal Process., vol. 12, no. 1, pp.132–143, 2018.

[Raj 2018] Raj, V. & Kalyani, S. Back propagating through the air: Deep learning at physical layer without channel model. in IEEE Commun. Lett., vol. 22, no. 11, pp. 2278–2281, 2018.

[Aoudia 2019] Aoudia, F. A., and Hoydis, J. Model-free training of end-to-end communication systems. IEEE J. Sel. Areas Commun., vol. 37, no. 11, pp. 2503—2516, 2019.

[Ye 2018] Ye, H. et al. Channel agnostic end-to-end learning based communication systems with conditional GAN. In: Proc. of EEE Globecom Workshops (IEEE GC Wkshps'18), Abu Dhabi, UAE, Dec. 2018, pp.1–5.

[Aoudia 2018] Aoudia, F. A. & Hoydis, J. End-to-end learning of communications systems without a channel model. arXiv preprint arXiv:1804.02276, 2018.

[Ye 2020] Ye, H. et al. Deep learning-based end-to-end wireless communication systems with conditional GANs as unknown channels. IEEE Trans. Wireless Commun., vol. 19, no. 5, pp. 3133–3143, 2020.

[Dorner 2020] Dorner, S. et al. WGAN-based autoencoder training over-the-air. In Proc. IEEE Int. Workshop on Signal Process. Adv. in Wireless Commun. (IEEE SPAWC'20), May 2020, pp. 1–5.

[Creswell 2018] Creswell, A. et al. Generative adversarial networks: An overview. IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 53–65, 2018.

[Arjovsky 2017] Arjovsky, M. & Bottou, L. Towards Principled Methods for Training Generative Adversarial Networks. arXiv preprint arXiv:1701.04862, 2017.

[Gulrajani 2017] Gulrajani, I. et al. Improved training of wasserstein gans. Advances in neural information processing systems, 2017, pp. 5767–5777.

[Srivastava 2014] Srivastava, N. et al. Dropout: a simple way to prevent neural networks from over-fitting. The journal of machine learning research, vol. 15, no. 1, pp. 1929–1958, 2014.

[Prechelt 1998] Prechelt, L. Early stopping-but when?. Neural Networks: Tricks of the trade. Springer, 1998, pp. 55–69.

[Bin 2021] Bin, K. et al. Compressive Data Gathering with Generative Adversarial Networks for Wireless Geophone Networks. IEEE Geoscience and Remote Sensing Letters, vol. 18, no. 3, pp. 558-562, March 2021, doi: 10.1109/LGRS.2020.2978520.

[Hu 2021] T. Hu, et al. Channel Estimation Enhancement with Generative Adversarial Networks. IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 1, pp. 145-156, March 2021, doi: 10.1109/TCCN.2020.3013257.

[Balevi 2021] Balevi, E. & Andrews, J. G. Wideband Channel Estimation with a Generative Adversarial Network. IEEE Transactions on Wireless Communications, vol.

20, no. 5, pp. 3049-3060, May 2021, doi: 10.1109/TWC.2020.3047100.

[Shi 2021] Shi, Y. et al. Generative Adversarial Network in the Air: Deep Adversarial Learning for Wireless Signal Spoofing. IEEE Transactions on Cognitive Communications and Networking, vol. 7, no. 1, pp. 294-303, March 2021, doi: 10.1109/TCCN.2020.3010330.

[Btattacharya 2021] Btattacharya, S. et al. A Comparative Analysis on Prediction Performance of Regression Models during Machining of Composite Materials. MATERIALS, NOV 2021. Vol. 14, No. 21. Doi: 10.3390/ma14216689.

[Leng 2022] Leng, C. et al. GAN for Load Estimation and Traffic-aware Network Selection for 5G Terminals. In IEEE Internet of Things Journal, doi: 10.1109/JIOT.2022.31527

[Jiang 2011] Jiang, H. Residual-Aided End-to-End Learning of Communication System without Known Channel. arXiv preprint arXiv: 2102.10786, 2021.

[Goodfellow 2016] Goodfellow, I. et al. Deep Learning. MIT Press, 2016.

[Kingma 2015] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In Proc. 3rd International Conference on Learning Representations, San Diego, CA, USA, May 7–9, 2015.

[He 2016] He, K. et al. Deep residual learning for image recognition. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (IEEE CVPR'16), Jun. 2016, pp. 770–778.

[Glorot 2010] Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feed forward neural networks. Journal of Machine Learning Research, Vol. 9, pp. 249–256, 2010.

[Sustika 2020] Sustika, R. et al. Generative adversarial network with residual dense generator for remote sensing image super resolution. In: Proc. Int. Conf. Radar, Antenna, Microw., Electron., Telecommun. (ICRAMET'20), Nov. 2020, pp. 34–39.

CONTACTS:

Thu Dau Mot University, Thu Dau Mot City, Binh Duong Province, Vietnam; Website: https://www/tdmu.edu.vn

Cong-Danh Huynh, Mr.
E: danhhc@tdmu.edu.vn

Thanh-Khiet Bui, Ph.D.
E: khietbt@tdmu.edu.vn

Department of Machining, Assembly and Engineering Technology, Faculty of Mechanical Engineering, VSB—Technical University of Ostrava, 17. listopadu 2172/15, 708 00 Ostrava, Czech Republic

Ing. Jiri Hajnys, Ph.D.
E: jiri.hajnys@vsb.cz