

Deep Deterministic Policy Gradient for End-to-End Communication Systems without Prior Channel Knowledge

Bolun Zhang and Nguyen Van Huynh

School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, UK

Abstract—End-to-End (E2E) learning-based concept has been recently introduced to jointly optimize both the transmitter and the receiver in wireless communication systems. Unfortunately, this E2E learning architecture requires a prior differentiable channel model to jointly train the deep neural networks (DNNs) at the transceivers, which is hardly obtained in practice. This paper aims to solve this issue by developing a deep deterministic policy gradient (DDPG)-based framework. In particular, the proposed solution uses the loss value of the receiver DNN as the reward to train the transmitter DNN. The simulation results then show that our proposed solution can jointly train the transmitter and the receiver without requiring the prior channel model. In addition, we demonstrate that the proposed DDPG-based solution can achieve better detection performance compared to the state-of-the-art solutions.

Index Terms—End-to-end communications, signal detection, channel estimation, deep deterministic policy gradient, and deep learning.

I. INTRODUCTION

The existing communication system is designed based on multiple signal processing blocks where each is separately implemented and optimized for a particular task. This multi-block architecture has achieved decent performance, but at the cost of increasing design complexity [1]. Besides, the transmitter and the receiver cannot be jointly optimized under this conventional architecture [2]. Recently, deep learning (DL) technique has been applied to the communication system design, which interprets the E2E communication system as a DNN-based auto-encoder over an interference channel [3]. This E2E-based learning method theoretically assumes the availability of the explicit channel model, which is perceived as an intermediate layer connecting the transmitter and the receiver. The auto-encoder architecture can therefore be trained in a supervised manner to jointly optimize for both the transmitter and the receiver. The biggest shortcoming of this E2E-based learning approach is that the training process is implemented based on the assumption of the prior differentiable channel model [4]. However, the channel model in actual scenarios is usually considered as a black box, where the channel transfer function is typically non-differentiable and the channel gradients are difficult to estimate, especially in wireless communications.

To address this problem, an alternating training approach is proposed in [5], where the authors perform supervised learning to train the receiver and reinforcement learning (RL)

to train the transmitter. This approach demonstrated that the E2E communication systems can be trained without any prior assumption of channel model. In [6], an improved version of alternating training scheme with noisy feedback link is proposed. Although these methods can perform training without the knowledge of channel model, they can only work well with simple channel models, e.g., Additive White Gaussian Noise (AWGN) channel. Meanwhile, these solutions require long training time as it uses a “Transformer” network to estimate the channel responses, and a “Discriminator” network to recover the distorted signal.

In this paper, we propose a novel deep reinforcement learning (DRL) based E2E communication system using DDPG algorithm to address these challenges, where both the transmitter and the receiver can be trained over an unknown channel. Our major contributions are summarized as follows.

- We develop a DDPG-based E2E communication system, which can jointly optimize the transmitter and the receiver without the prior knowledge of channel model.
- By utilizing convolutional neural network (CNN), our proposed scheme can achieve notable performance enhancement compared to the alternating training scheme for both the Rayleigh and Rician fading channels.
- Our solution can achieve a lower steady state of block error rate (BLER) compared to the alternating training scheme within the same training time.

II. SYSTEM MODEL

A. Auto-Encoder based End-to-End Communication Systems

We consider an auto-encoder based E2E communication system, as illustrated in Fig. 1. The goal of E2E communication system is to jointly optimize the transceivers for better communication performance. In particular, it expresses the transmitter and the receiver as two independent DNNs such that the traditional signal processing blocks at the transmitter are represented as an encoder and the blocks at the receiver are represented as a decoder. The transmitter aims to reliably deliver symbol s to the receiver over the noisy channel, and the receiver aims to recover the distorted signal $\mathbf{y} \in \mathbb{C}^n$ to the original symbol s . The symbol s is firstly converted to one-hot encoding vector $\mathbf{m} \in \mathbb{M}$, then the transmitter encodes the message \mathbf{m} into the encoded signal $\mathbf{x} \in \mathbb{C}^n$, where \mathbb{C}^n

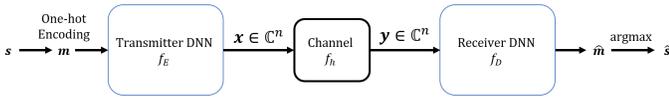


Fig. 1: Overview of the typical E2E communication system.

denotes that the encoded signal \mathbf{x} is presented as complex numbers, making n discrete uses of channel. The process of E2E communication system can be demonstrated as the cascade of three individual functions, which can be expressed as:

$$\hat{s} = f_D(f_h(f_E(s; \theta_E)); \theta_D), \quad (1)$$

where f_E represents the encoder function which maps the original symbol to the encoded signal, i.e., $\mathbf{x} = f_E(s; \theta_E)$, with θ_E denotes the trainable weights at the transmitter. f_h represents the channel impairments with channel realization \mathbf{h} , i.e., $\mathbf{y} = f_h(\mathbf{x})$. f_D represents the decoder function that aims to recover the received signal to the estimated symbol, i.e., $\hat{s} = f_D(\mathbf{y}; \theta_D)$, and θ_D denotes the trainable weights at the receiver. This E2E system is trained in an supervised manner which aims to minimize the loss function, i.e., $L = \mathcal{L}(m, \hat{m})$. The loss function $\mathcal{L}(s, \hat{s})$ is regarded as the objective function which measures the distance between the original symbol s and the estimated message \hat{s} for such E2E framework.

Such an E2E learning approach regards the whole system as an auto-encoder and optimizes the DNNs for the transmitter and the receiver in a data-driven and supervised manner. The training process is implemented by computing the error gradients of the differentiable loss function with respect to the parameters at each layer of the DNNs. The calculated error gradient is then back-propagated through the network, allowing each layer to adjust its weights and biases to minimize the E2E loss. Therefore, the E2E learning approach requires to mathematically formulate a differentiable channel function in advance to allow the error gradients to be back-propagated from the receiver to the transmitter, hence obtaining the global optimization.

B. Limitations of End-to-End Communication Systems

The primary shortcoming of the E2E paradigm is that it is built on the assumptions of the explicit and differentiable channel model, which may not be always practical in real-world scenarios. The channel effects in the actual communication systems are subject to various factors, such as signal attenuation, additive noise, multi-path fading, and time-varying channel realization. As a result, the actual channel models are often non-differentiable, making it challenging to optimize the system using conventional gradient-based method. The non-differentiable channel model potentially blocks the back-propagation process of the error gradients, which may result in local optimizations. This scenario can cause the system focus solely on optimizing the receiver network, neglecting the transmitter network and hindering the overall learning of the system. In addition, acquiring perfect CSI in practical

scenarios can be challenging due to several factors, including hardware limitations and dynamic channel conditions [7]. Consequently, it can be difficult to precisely model the channel behaviour, and the assumptions of the explicit and differentiable channel functions might limit or even impair the performance of E2E approach in practical communication scenarios.

In this paper, we propose a DDPG-based E2E communication system to overcome the limitation of non-differentiable channel models and imperfect CSI. By employing experience replay technique [8], the transmitter and the receiver can be jointly optimized without knowing any prior knowledge of the channel models. In the following sections, we will present our proposed DDPG solution in detail.

III. DEEP DETERMINISTIC POLICY GRADIENT FOR E2E COMMUNICATION SYSTEMS

This section first provides the fundamentals of reinforcement learning and deep reinforcement learning. Then, the proposed DDPG-based framework for E2E communications is discussed.

A. Deep Reinforcement Learning Basics

A typical RL system consists of an agent and an environment, and the goal of RL is to train the agent to take actions in the unknown environment so as to collect the experience and maximize the cumulative reward by constant explorations. In particular, at time step t , the agent observes the environment, i.e., state s_t , and makes action a_t based on its current policy π . After performing action a_t , the agent observes the immediate reward r_t and the next state s_{t+1} . For a given policy π , the action-value function, i.e., Q-function, which measures the expected return for a given state-action pair, can be mathematically expressed as:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a], \quad (2)$$

where $G_t = \sum_{t=1}^{\infty} \gamma^t r_t$ indicates the cumulative discounted reward for one trajectory, and $\gamma \in (0, 1]$ represents the discount factor which measures the importance of the long-term rewards to immediate rewards. In Q-learning, the update rule of the Q-value used for learning optimal policy can be expressed as:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [R(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a)], \quad (3)$$

where $R(s, a)$ denotes the immediate reward given state s and action a . s' denotes the next state that the agent transits to after performing action a in state s , a' is the next action that the agent can take at state s' , and α is the learning rate which determines how much weights is given to the new information. This algorithm iteratively improves the policy by updating the Q-values and selecting the action with the highest Q-value at each state, which aggressively select local optimum for every decision during exploration.

To further improve the convergence of Q-learning, deep Q-learning is proposed by replacing the Q-table by a DNN,

namely deep Q-network (DQN) [9]. It utilizes the power of DNN in solving regression problems, which takes the state-action pair as the input and approximates the corresponding Q-value by making predictions. The main advantage of deep Q-learning over Q-learning is the efficiency in handling complex tasks while Q-learning cannot obtain the optimal policy for complex problems in reasonable time. Another advantage of deep Q-learning is the experience replay method which stores the accumulated information in the buffer allowing the agent to sample from the past experience to optimize the policy network. The experience replay technique makes efficient use of the cumulative experience and solves the data correlation problem in the observation sequence by random sampling. However, the biggest shortcoming of deep Q-learning is that it cannot directly perform on continuous action space, while the action spaces in many real-world applications are continuous, e.g., the encoded signal in communication systems is time-varying and cannot be measured by discrete states. The second issue of deep Q-learning is that the high update frequency of the target network causes the behaviour network “chasing” a moving target. This paper introduces an advantaged deep Q-learning algorithm, namely Deep Deterministic Policy Gradient (DDPG), to not only handle the continuous action space but can also jointly train the transmitter and the receiver without prior channel information.

B. End-to-End Communication Systems using DDPG

DDPG is an off-policy actor-critic algorithm taking the advantages of deep Q-learning and policy gradient [10]. There are several significant features of the proposed DDPG-based system: (i) it uses deterministic policy for network optimizations, which allows the agent to directly operate on the continuous space action, (ii) it adopts actor-critic method, where the actor network is to generate the deterministic action for a given state and the critic network is to produce a score which measures the performance of the current action, and (iii) it trains two target networks respectively for the actor and the critic, and thus further increasing the training stability. The target networks are the time-delayed copies of their original networks that slowly track the behaviour networks, which significantly improves the stability in training process.

An overview of the proposed DDPG-based E2E communication system is presented in Fig. 2, where the observation state is the input message, i.e., $s_t = \mathbf{m}$. The action is the encoded signal, i.e., $\mathbf{a}_t = \mathbf{x}$. \mathbf{y} is the received signal and $\hat{\mathbf{m}}$ is the estimated output message. s_b and \mathbf{a}_b are the random state-action batches sampled from the experience buffer for training the receiver. The channel model and the receiver model are the major components of the environment. The DDPG agent consists of the actor and the critic networks, where the actor network, i.e., transmitter model, maps the input message \mathbf{m} to the encoded signal \mathbf{x} and sends it to the environment. The channel model in the environment distorts the encoded signal \mathbf{x} and sends the damaged signal \mathbf{y} to the receiver model which eventually decodes the damaged signal \mathbf{y} to the estimated message $\hat{\mathbf{m}}$. The loss of the output is calculated

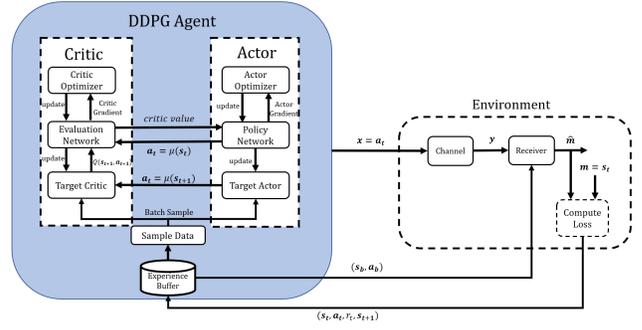


Fig. 2: Overview of DDPG-based E2E communication system.

using categorical cross-entropy loss. As mentioned, the loss of the receiver DNN will be fed back to the transmitter to update its DNNs. In particular, the reward at time t after making an action at state s_t is the negative value of the computed loss, which can be expressed as follows:

$$r_t = \frac{1}{n} \sum_{i=1}^n (s_t)_i \cdot \log[f_{\theta_R}(\mu((s_t)_i | \theta_T))], \quad (4)$$

where θ_R is the parameter of the receiver network, θ_T is the parameter of the transmitter network, and $\mu(s_t | \theta_T)$ is the action performed by the policy μ at the given state s_t based on the transmitter parameters θ_T . f_{θ_R} represents the process at the receiver model to map the damaged signal to the estimated output signal, and n refers to the length of the input message.

The pseudocode of the whole training process is depicted in Algorithm 1. At the beginning stage, the actor and the critic networks are initialized with random parameters θ^μ and θ^Q , respectively. Two target networks μ' and Q' are initialized with the same weights for initial balance. A replay buffer with capacity of C is initialized to store the cumulative information. During the training stage, a random observation state, i.e., input message $s_t = \mathbf{m}$ is generated from the environment at the beginning of each episode. For every time step in each episode, the actor network maps the current observation state s_t to the action, i.e., encoded signal $\mathbf{a}_t = \mathbf{x}$. The current observation state and action are sent to the environment, where the observation state is used as the label of the estimated output to calculate the loss value. The environment will then return the next observation state s_{t+1} and the calculated reward r_t as feedback to the agent for updating the actor and critic networks. A collection of state, action, reward and next state will be stored in the experience buffer of the agent. After collecting the data, a mini-batch of experiences is randomly sampled from the replay buffer, and the target Q-value for each state-action pair in the mini-batch is computed using the following equation,

$$\mathbf{y}_t = r_t + \gamma Q(s_{t+1}, \mathbf{a}_{t+1}). \quad (5)$$

During the optimization of the actor and the critic networks, we aim to maximize the expected return, i.e., the expected value of the Q-function $Q(s, \mathbf{a})$ by iteratively updating the

Algorithm 1 DDPG for E2E Communication Systems

- 1: Randomly initialize actor network $\mu(s | \theta^\mu)$ and critic network $Q(s, \mathbf{a} | \theta^Q)$
- 2: Initialize target network Q' and μ' with the same weights
- 3: Initialize replay buffer with capacity C and batch size B
- 4: **for** $episode = 1$ to E **do**
- 5: Initialize a random state for the input message $s_t = \mathbf{m}$
- 6: Initialize the episodic reward as zero
- 7: **for** $step = 1$ to N **do**
- 8: Assign next state from last iteration to current state: $s_t = s_{t+1}$
- 9: Select the action (encoded signal) from actor network: $\mathbf{x} = \mathbf{a}_t = \mu(s | \theta^\mu)$
- 10: Feed the state s_t and action \mathbf{a}_t to the environment
- 11: Observe the new state s_{t+1} , reward r_t and done information
- 12: Store the transition $(s_t, \mathbf{a}_t, r_t, s_{t+1})$ to Buffer
- 13: Sample a batch of B transitions $(s_t, \mathbf{a}_t, r_t, s_{t+1})$
- 14: Set $\mathbf{y}_t = r_t + \gamma Q(s_{t+1}, \mathbf{a}_{t+1})$
- 15: Update critic network by minimizing the loss:

$$L = \frac{1}{N} \sum_t (\mathbf{y}_t - Q(s_t, \mathbf{a}_t | \theta^Q))^2$$

- 16: Update actor network using sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \nabla_{\mathbf{a}} Q(s, \mathbf{a}) \nabla_{\theta^\mu} \mu(s | \theta^\mu)$$

- 17: Update target networks for both actor and critic:

$$\theta^{Q'} = \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

$$\theta^{\mu'} = \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

- 18: Sample a random batch of D transitions (\mathbf{a}_t, s_t)
 - 19: Train Receiver model with the sampled batch
 - 20: **if** $done == True$ **then**
 - 21: Break
 - 22: **end if**
 - 23: **end for**
 - 24: **if** $stop\ criterion\ met$ **then**
 - 25: Break Main Training Loop
 - 26: **end if**
 - 27: **end for**
-

actor and the critic networks. The expected value of the Q-function can be expressed as:

$$J(\theta) = \mathbb{E}[Q(s, \mathbf{a}) | s=s_t, \mathbf{a}_t=\mu(s_t)], \quad (6)$$

where s_t denotes the current state of the environment, and \mathbf{a}_t denotes the current action selected by the policy function, i.e., $\mathbf{a}_t = \mu(s_t)$. The next essential step of the optimization of the actor network, i.e., the policy function $\mathbf{a}_t = \mu(s_t)$ is to take the gradient of the expected Q-value with respect to the policy parameters θ_μ , which allows it to improve the expected cumulative reward. The gradient of the expected Q-value with

respect to the policy parameter can be expressed as:

$$\nabla_{\theta^\mu} J \approx \nabla_{\mathbf{a}} Q(s, \mathbf{a}) \nabla_{\theta^\mu} \mu(s | \theta^\mu). \quad (7)$$

After the optimization of the behaviour networks of the actor and the critic, we update their target networks based on soft update rule to slowly track the parameters of the learned networks. The optimization process of the target networks can be expressed as:

$$\theta' = \tau \theta + (1 - \tau) \theta', \quad (8)$$

where θ' denotes the parameters of target networks, θ denotes the parameters of behaviour networks, and τ is a hyper-parameter called “soft update rate”, which controls the blending between the behaviour network and target network weights. τ is defined as a positive value which is smaller than 1 for slow update, i.e., $\tau \ll 1$. The receiver model is trained in a supervised manner with the state-action pair batch randomly sampled from the buffer. In this way, the proposed solution can jointly optimize both the transmitter and the receiver without knowing the channel model in advance.

C. Network Architecture of Transmitter and Receiver

The network architectures of the transmitter and the receiver are illustrated in Fig. 3. On the transmitter side, the symbol s is encoded to one-hot vector \mathbf{m} of size M , i.e., $M = 2^k$, where k indicates the number of bits. The message \mathbf{m} is fed to the transmitter. More specifically, the transmitter in Fig. 3a consists of two 1-Dimensional convolutional (Conv1D) layers, followed by an L2 normalization layer [11] which normalizes the total power of the encoded signal, i.e., $\|\mathbf{x}\|^2 \leq n$. The first convolutional layer uses M filters and is followed by a batch normalization layer and Exponential Linear Unit (ELU) activation function. The second convolutional layer uses $2n$ filters to reshape the length of the encoded signal. The encoded signal \mathbf{x} is a one-dimensional vector with length of $2n$, representing the complex-valued symbol. For the receiver, the estimated channel response $\hat{\mathbf{h}}$ and the received signal \mathbf{y} are together fed to the receiver network. The receiver consists of two Conv1D layers, followed by another Conv1D layer with Softmax activation function which produces a probability distribution vector of all possible output messages $\hat{\mathbf{m}}$. The algorithm then chooses the index with the highest probability to obtain the estimated symbol \hat{s} .

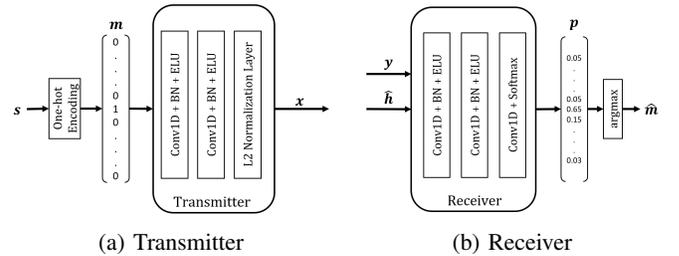


Fig. 3: Network architectures of (a) the transmitter and (b) the receiver.

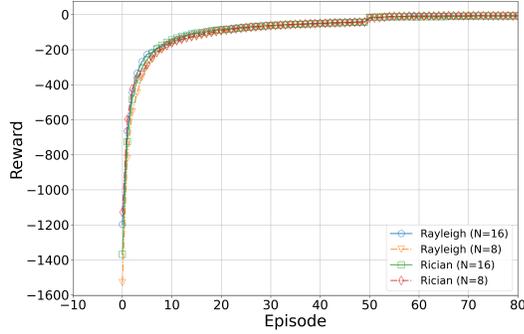


Fig. 4: Episodic reward of Rayleigh and Rician fading channels.

IV. SIMULATION RESULTS

Several experiments have been implemented on Rayleigh and Rician fading channels to demonstrate the effectiveness of the proposed DDPG-based solution. The baseline for comparisons is the alternating training scheme proposed in [5]. The following experiments are evaluated by BLER over different signal-to-noise ratio (SNR), ranging from 0 dB to 20 dB.

A. Parameter Setting

For the hyper-parameters settings, the soft update rate τ is 0.005, which enables slow updates of the target networks. The learning rates of the actor and critic networks are set at 0.0002 and 0.0001, respectively. The numbers of training episodes and time step for every episode are set as 30,000 and 500, respectively. The adjacent states are uncorrelated due to the nature of the E2E communication system that every observation state, i.e., input message, is generated randomly. Therefore, the discount factor of Bellman function is 0.1. The training of the DDPG-based solution is implemented on Rayleigh and Rician fading channels, with SNRs of 20 dB and 10 dB. The size of the input message M is set at 256, and the numbers of channel uses n are set at 16 and 8 for different experiments. An additional pilot signal was leveraged for channel estimation. The Rician factor is set as 1 indicating the portions of the Line-of-sight components and None-line-of-sight components are equivalent.

B. Performance Evaluation

The evolutions of the averaged reward over the first 80 episodes of both the Rayleigh and Rician fading channels with training SNR of 20 dB are presented in Fig. 4, averaged over the last 50 episodes. It suggests that the episodic reward of both fading channels can converge close to zeros within 80 episodes, indicating the feasibility of the proposed DDPG-based solution for training the E2E communication systems. The following figures demonstrate the BLER performance of the proposed DDPG-based scheme against the alternating training scheme over Rayleigh and Rician fading channels. It can be observed that our proposed solution significantly

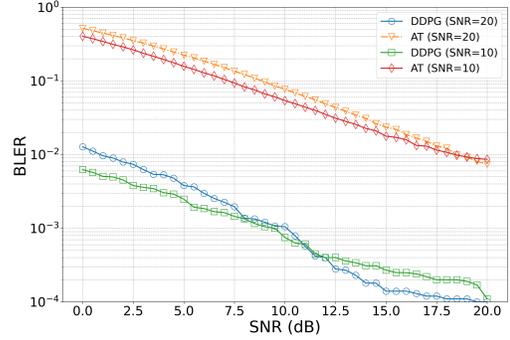


Fig. 5: BLER of Rayleigh fading channel with channel size of 16.

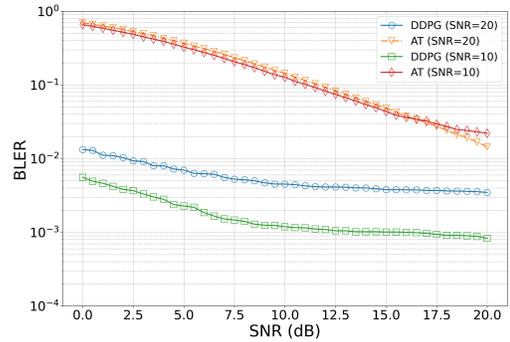


Fig. 6: BLER of Rayleigh fading channel with channel size of 8.

outperforms the alternating training method. With regards to channel size n of 16 under Rayleigh channel shown in Fig. 5, the DDPG-based method with the training SNR of 10 dB shows better performance before 10 dB, but worse after 10 dB compared to that with the training SNR of 20 dB. This indicates that the model trained with SNR of 10 dB is able to detect the distorted signal with SNRs lower than 10 dB but difficult to detect the distorted signal with SNRs higher than 10 dB. The similar trend happens in the Rician fading channel with channel size of 16 as shown in Fig. 7. In particular, two BLER curves of the DDPG-based method, intersect at about 16 dB, indicating that the trained DDPG model with SNR of 10 dB outperforms that with SNR of 20 dB before the intersecting point. For channel size of 8, the proposed scheme with the training SNR of 10 dB shows better performance for the whole SNR range under both Rayleigh and Rician fading channels, as shown in Fig. 6 and Fig. 8.

The BLER over training time of the two methods under different channel sizes are presented in Fig. 9 and Fig. 10, where AT refers to the alternating training scheme. The results show that the proposed DDPG-based solution can achieve lower steady state of BLER compared to the alternating training scheme. More specifically, as shown in Fig. 9, the

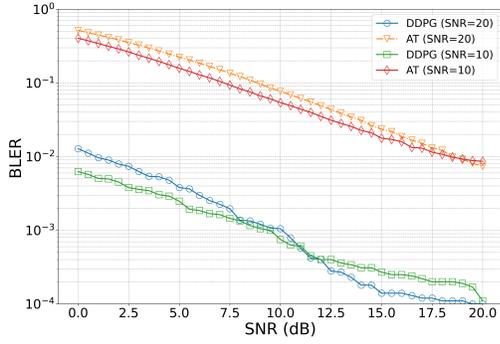


Fig. 7: BLER of Rician fading channel with channel size of 16.

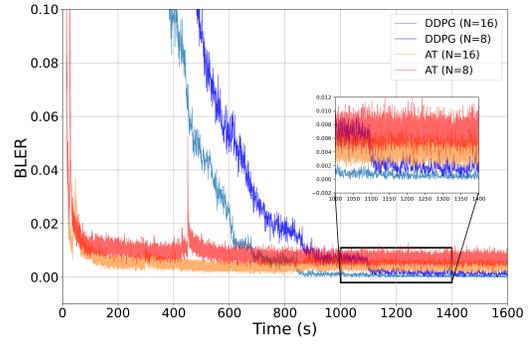


Fig. 10: BLER vs. training time on Rician fading channel.

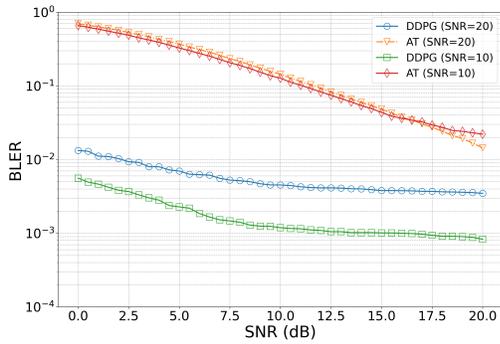


Fig. 8: BLER of Rician fading channel with channel size of 8.

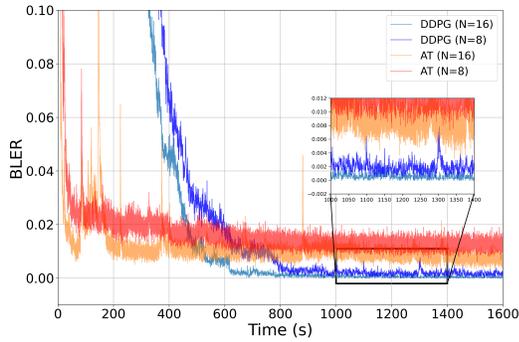


Fig. 9: BLER vs. training time on Rayleigh fading channel.

proposed scheme can converge to the optimal policy within 800 seconds while the alternating training cannot converge after 1,600 seconds for Rayleigh fading channel. In Fig. 10, the proposed scheme can converge within 1,000 seconds while the baseline scheme cannot converge after 1,600 seconds for Rician fading channel.

V. CONCLUSION

In this article, we have proposed a DDPG-based E2E learning solution to relax the requirement of the prior channel model. In particular, with the DDPG algorithm, the transmitter can update its DNN by learning from the reward, i.e., loss value, sent from the receiver, given the current state, i.e., bitstream, and the chosen action, i.e., encoded symbol. In this way, implicit information about the training process of the receiver DNN can be learned by the transmitter to adapt its DNN, and thus improving the whole system's performance. The simulation results have demonstrated the effectiveness of our proposed solution compared to existing solutions.

REFERENCES

- [1] E. Zehavi, "8-PSK trellis codes for a Rayleigh channel," *IEEE Transactions on Communications*, vol. 40, no. 5, pp. 873-884, May 1992.
- [2] S. Dörner, S. Cammerer, J. Hoydis, and S. T. Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132-143, Feb. 2018.
- [3] T. O'shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563-575, Oct. 2017.
- [4] O. Jovanovic, M. P. Yankov, F. D. Ros, and D. Zibar, "Gradient-free training of autoencoders for non-differentiable communication channels," *Journal of Lightwave Technology*, vol. 39, no. 20, pp. 6381-6391, Aug. 2021.
- [5] F. Aoudia and J. Hoydis, "End-to-end learning of communications systems without a channel model," *52nd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, 28-31 Oct. 2018.
- [6] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep reinforcement learning autoencoder with noisy feedback," *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, Avignon, France, 03-07 Jun. 2019.
- [7] V. Raj and S. Kalyani, "Backpropagating through the air: Deep learning at physical layer without channel models," *IEEE Communications Letters*, vol. 22, no. 11, pp. 2278-2281, Nov. 2018.
- [8] S. Zhang and R. S. Sutton, "A deeper look at experience replay," [Online]. Available: arXiv:1712.01275.
- [9] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," *International conference on machine learning*, 2016.
- [10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," [Online]. Available: arXiv:1509.02971.
- [11] T. V. Laarhoven, "L2 regularization versus batch and weight normalization," [Online]. Available: arXiv:1706.05350.