Backpropagating Through the Air: Deep Learning at Physical Layer Without Channel Models

Vishnu Raj and Sheetal Kalyani

***Abstract*— Recent developments in applying deep learning techniques to train end-to-end communication systems have shown great promise in improving the overall performance of the system. However, most of the current methods for applying deep learning to train physical-layer characteristics assume the availability of the explicit channel model. Training a neural network requires the availability of the functional form all the layers in the network to calculate gradients for optimization. The unavailability of gradients in a physical channel forced previous works to adopt simulation-based strategies to train the network and then fine tune only the receiver part with the actual channel. In this letter, we present a practical method to train an end-to-end communication system without relying on explicit channel models. By utilizing stochastic perturbation techniques, we show that the proposed method can train a deep learning-based communication system in real channel without any assumption on channel models.**

***Index Terms*— Artificial neural networks, communication systems, optimization.**

I. Introduction

THE idea of end-to-end learning of communication systems has recently drawn a lot of attention from the wireless communication community. The fascinating idea of learning and optimizing the transmitter and the receiver jointly in a wireless communication system could improve the communication system architecture, which is currently optimized block by block for performance. O'Shea and Hoydis [1] present a novel way of rethinking the communication system as an end-to-end reconstruction problem, and applied autoencoder [2] based deep architecture. As a practical demonstration, they showed how deep learning systems can be used to derive modulation and decoding schemes for complex channel models and showed competitive performance to traditional communication systems.

The generalization power of deep neural networks on a wide range of complex problems enabled the application of learning techniques over a broad spectrum of communication challenges. In [3], a deep learning based Sparse Code Multiple Access (SCMA) technique is proposed which achieves lower BER than conventional techniques with less computational demands. For the problem of reducing Peak to Average Power Ratio (PAPR) in Orthogonal Frequency Division Multiplexing (OFDM) systems, [4] proposed an autoencoder based method for jointly minimizing Bit Error Rate (BER) and PAPR and showed that their method can outperform conventional methods. Channel estimation and signal detection has been cast as a deep learning problem in [5] and the results show robust performance over a wide range of Signal-to-Noise Ratio (SNR) with competitive performance to traditional approaches, but with less computational complexity. Other notable works where deep learning is applied to physical layer of wireless communication include Multiple-Input-Multiple-Output (MIMO) detection [6], equalization and synchronization in OFDM [7], etc.

In a neural network, the training proceeds by calculating the error gradients of a differentiable loss function with respect to the parameters at each layer of the network and then uses a technique called *backpropagation* to update the weights of each nodes sequentially. In an end-to-end communication system, the channel is also considered as a layer in the neural network and hence the previous works required to know the precise functional form of the channel to calculate the error gradients. Even though the works in [1], [4], and [8] clearly shows a superior performance for deep learning based methods over the traditional hand-crafted communication system design, all of them have to assume a channel model (functional form of the channel) to train the network using backpropagation. Hence, a synthetic channel model is used to train the models in simulation over which the gradients can be calculated. However, in real systems, due to the impairments in signaling and detection schemes, hardware imperfections, varying channel conditions etc., the channel seen by the trained network could become significantly different from the one used for training the network. These issues can adversely impact the system performance. As the application of deep learning techniques in physical layer is showing great promise in improving the system performance, we need methods which can robustly handle such situations.

In order to circumvent the problem of physical channel blocking the backpropagation of error gradients, [8] used a two stage approach. After training the deep learning model with a synthetic channel model in simulation, the receiver layers are fine tuned to match the channel characteristics over the real channel. In order to approximate the characteristics of observed channel, [9]–[11] used Generative Adversarial Networks to train a surrogate channel network and then used this surrogate network in place of channel for backpropagation. By applying supervised training at receiver layers and a variant of reinforcement learning in transmitter layers, [12] eliminated the need to approximate the observed channel. However, all these methods relied on a two phase alternating training paradigm, which takes more samples to converge.

Manuscript received July 11, 2018; revised July 28, 2018; accepted August 27, 2018. Date of publication August 31, 2018; date of current version November 12, 2018. The associate editor coordinating the review of this paper and approving it for publication was A. Al-Fuqaha. *(Corresponding author: Vishnu Raj.)*

The authors are with the Department of Electrical Engineering, IIT Madras, Chennai 600 036, India (e-mail: ee14d213@ee.iitm.ac.in; skalyani@ee.iitm.ac.in).

Digital Object Identifier 10.1109/LCOMM.2018.2868103

1558-2558 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: Indian Institute of TechnologyDharwad. Downloaded on July 14,2022 at 09:59:15 UTC from IEEE Xplore. Restrictions apply.

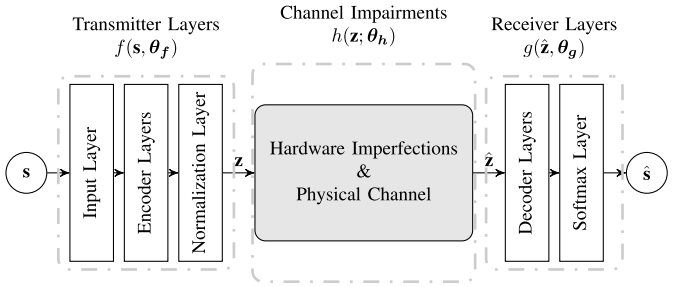


Fig. 1. Autoencoder based communication system.

Further, the surrogate network used to capture the probability distribution of channel is still an approximation to the actual channel. At the cost of more sample complexity, this could be improved by training the surrogate with more samples.

In this letter, we propose a novel online single stage approach to train deep learning based communication systems from end-to-end in a real channel without any assumptions about the channel model. To overcome the problem of unknown channel gradients, we use a stochastic approximation technique to calculate approximate gradients for the model. With both the training and deployment of the system happening in the actual channel, the trained network should be well optimized for the problem.

II. Proposed Approach

A conventional wireless communication system consists of multiple blocks, each of which is individually optimized for specific tasks such as source encoding, channel coding, signal shaping etc. At the transmitter, let $\mathbf{s}$ denotes the data symbols to be sent through the channel to the receiver. The Transmitter maps $\mathbf{s}$ to a representation $\mathbf{x}$ which is the ideal form for transmission. At the receiver, a corrupted version of $\mathbf{x}$, denoted by $\hat{\mathbf{x}}$, is received and a reconstruction of symbol $\mathbf{s}$, denoted by $\hat{\mathbf{s}}$, is obtained. A transmission is considered successful if $\hat{\mathbf{s}}$ is a faithful reproduction of $\mathbf{s}$. The challenge in wireless communication lies at faithful reconstruction of $\mathbf{s}$ at receiver end under adverse channel conditions. In this letter, we consider the impairments due to hardware imperfection as well as corruption by physical channel as channel impairments; a robust communication system should be able to handle any type of corruption to the data $\mathbf{x}$.

In Fig.1, we provide the architecture of an autoencoder based communication system. Instead of using different blocks optimized for specific purposes, a deep neural network which includes the functionality of both transmitter and receiver is used. First few layers of the network (labeled as *Transmitter Layers*) encodes the data $\mathbf{s}$ into a representation $\mathbf{z}$, which on passing through the channel layer gives a corrupted version, $\hat{\mathbf{z}}$. The last few layers of the network (labeled as *Receiver Layers*) produce an estimate $\hat{\mathbf{s}}$ of the data $\mathbf{s}$ from $\hat{\mathbf{z}}$. The channel layer in this architecture captures both the hardware impairments and the corruption by physical channel. An exact knowledge of the functional form $h(\cdot)$ as well as the value of parameters $\boldsymbol{\theta_h}$ of the channel layer are unknown while training the model over real physical channel. In the case of design of traditional communication systems, approximations of $h(\cdot)$ (channel models) are utilized in optimizing individual blocks in the system.

Let $\boldsymbol{\theta_f}$ and $\boldsymbol{\theta_g}$ denote the set of parameters for the autoencoder network. Training the autoencoder is the process of optimizing the parameters for a loss function $\mathcal{L}(\mathbf{s}, \hat{\mathbf{s}})$ using stochastic gradient descent algorithm (or a variant of it). In this work, we consider one-hot representation for the input symbol $\mathbf{s}$. The output of the network is a probability distribution $\hat{\mathbf{s}}$ over the set of possible symbols. We use cross entropy over a batch of N symbols, defined in (1), as the loss function to be minimized.

$$\mathcal{L}(\mathbf{s}_{1:N}, \hat{\mathbf{s}}_{1:N}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{s_i \in \mathbf{s}_n} s_i \log(\hat{s}_i) \tag{1}$$

Here s_i denotes the i^{th} element in vector $\mathbf{s}$.

A. Challenge in Training Over Physical Channel

The transfer function of the modeled communication system can be written as $\hat{\mathbf{s}} = g\left(h\left(f\left(\mathbf{s}, \boldsymbol{\theta_f}\right); \boldsymbol{\theta_h}\right), \boldsymbol{\theta_g}\right)$. While training the neural network, we can update the values for parameters $\boldsymbol{\theta_f}$ and $\boldsymbol{\theta_g}$. But, $\boldsymbol{\theta_h}$ is part of the channel which we do not have control over and hence, cannot be updated. In order to update any parameter $\boldsymbol{\theta}$, we can use the gradient descent based update given in (2) with a step size of α.

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\partial \mathcal{L}(\mathbf{s}, \hat{\mathbf{s}})}{\partial \boldsymbol{\theta}} \tag{2}$$

For the receiver layers, the updation of parameters $\boldsymbol{\theta_g}$ is possible because we know the function $g(\cdot)$ and have control over the parameters $\boldsymbol{\theta_g}$. In an abstract way, the update for receiver layer parameters $\boldsymbol{\theta_g}$ can be written as

$$\boldsymbol{\theta_g} \leftarrow \boldsymbol{\theta_g} - \alpha \frac{\partial \mathcal{L}(\mathbf{s}, \hat{\mathbf{s}})}{\partial g} \frac{\partial g}{\partial \boldsymbol{\theta_g}}, \tag{3}$$

and can be exactly calculated. However, to update the parameters $\boldsymbol{\theta_f}$ of the transmitter layers, the gradient update will take the form

$$\boldsymbol{\theta_f} \leftarrow \boldsymbol{\theta_f} - \alpha \frac{\partial \mathcal{L}}{\partial g} \frac{\partial g}{\partial h} \frac{\partial h}{\partial f} \frac{\partial f}{\partial \boldsymbol{\theta_f}}. \tag{4}$$

While training on a real channel, this poses a problem as we do not have access to the terms $\frac{\partial g}{\partial h}$ and $\frac{\partial h}{\partial f}$ because of the unavailability of a functional form for $h(\cdot)$. This necessitated previous works [1], [5]–[7] to train the model in simulation and then, fine tune only the receiver part with real channel. Since the gradients over the physical channel are impossible to calculate without the knowledge of exact channel model, we propose to use a gradient free optimization technique called Simultaneous Perturbation Stochastic Optimization (SPSA) to train the communication system.

B. Approximating the Error Gradients

SPSA [13] is a descent based optimization algorithm which works by approximating gradients. It is used in the cases where complex systems need to be optimized but the transfer function of the system is unknown [13]. SPSA proceeds by perturbing

Authorized licensed use limited to: Indian Institute of TechnologyDharwad. Downloaded on July 14,2022 at 09:59:15 UTC from IEEE Xplore. Restrictions apply.

the parameters to be updated, observes the loss with perturbed parameters and calculate approximate gradients for parameter updates. Let J denote the complex loss function to be optimized and $\boldsymbol{\theta}$ denotes the associated parameter vector. The SPSA algorithm is given in Alg. 1. The distribution $\mathcal{D}$ should be selected such that the elements in perturbation vector $\boldsymbol{\Delta}$ are independent and symmetrically distributed about 0 with finite inverse moments. A popular choice for $\mathcal{D}$ is Rademacher distribution. Provided the above discussed conditions are met, SPSA is proved to almost surely converge to minima of the loss function [14].

Algorithm 1 SPSA Algorithm

Parameters: $a > 0, A \geq 0, c > 0, \alpha \in (0,1], \gamma \in (1/6, 1/2]$ and a distribution $\mathcal{D}$.
for $k = 1, 2, 3, \ldots$ **do**
  Sample a vector $\boldsymbol{\Delta} \sim \mathcal{D}$
  $a_k = \frac{a}{(k+A)^{\alpha}}$
  $c_k = \frac{c}{k^{\gamma}}$
  $\hat{g} = \frac{J(\boldsymbol{\theta}+c_k\boldsymbol{\Delta}) - J(\boldsymbol{\theta}-c_k\boldsymbol{\Delta})}{2\ c_k\boldsymbol{\Delta}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - a_k \cdot \hat{g}$
end for

By using SPSA, the parameters of the autoencoder network can be updated without the knowledge of channel transfer function, $h(\cdot)$. At k^{th} step, after observing loss $\mathcal{L}(\cdot)$, the parameters $\boldsymbol{\theta}$ can be updated as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - a_k \frac{\mathcal{L}(\boldsymbol{\theta} + c_k\boldsymbol{\Delta}) - \mathcal{L}(\boldsymbol{\theta} - c_k\boldsymbol{\Delta})}{2c_k\boldsymbol{\Delta}}. \tag{5}$$

We provide the procedure to train the autoencoder network channel model using SPSA in Alg. 2.

Algorithm 2 Proposed: SPSA for training

Parameters: Set parameters (a, c, α, γ) for SPSA
Initialization: Initialize $\boldsymbol{\theta_f}, \boldsymbol{\theta_g}$ with random values
for $k = 1, 2, 3, \ldots$ **do**
  Transmit a set of symbols through the transmitter
  Obtain estimate $\hat{\mathbf{s}}$ at receiver
  Calculate loss given by (1)
  Update $\boldsymbol{\theta_f}, \boldsymbol{\theta_g}$ using Alg. 1
end for

However, being an approximate gradient method, SPSA will require more samples to converge. Only transmitter layers are affected by the unavailability of gradient updates in the absence of actual channel model. In order to accelerate the training process, Stochastic Gradient Descent (SGD) (or its variants) can be used to update the weights at the receiver layer and SPSA based optimizer can be used to update the weights at the transmitter layer. Since both the optimizers are proved to converge to the optimal point under given assumptions, the proposed training method should also converge to optimal point with sufficient training. We provide the procedure for combining SGD and SPSA for efficiently training the system in Alg. 3.

Algorithm 3 Proposed: SPSA+SGD for Training

Parameters: Set parameters (a, c, α, γ) for SPSA
Parameters: Set parameters for SGD (or a variant)
Initialization: Initialize $\boldsymbol{\theta_f}, \boldsymbol{\theta_g}$ with random values
for $k = 1, 2, 3, \ldots$ **do**
  Transmit a set of symbols through the transmitter
  Obtain estimate $\hat{\mathbf{s}}$ at receiver
  Calculate loss given by (1)
  Update $\boldsymbol{\theta_g}$ using SGD (or a variant)
  Update $\boldsymbol{\theta_f}$ using Alg. 1
end for

TABLE I
TRAINING PARAMETERS

Parameter	Value
Input Layer dimension	4
Number of Encoder layers	2
Hidden Units in Encoder Layers	{4,2}
Number of Decoder layers	2
Hidden Units in Decoder Layers	{4,4}
Activation for Hidden layers	$ReLU$
Softmax Layer dimension	4
Adam Parameters	$\alpha = 0.01, \beta_1 = 0.99, \beta_2 = 0.999$
SPSA Parameters	$a = 0.05, c = 0.1, \alpha = 0.9, \gamma = 0.30$
$\mathcal{D}$	Rademacher Distribution
Mini-batch size	500 symbols
Train SNR (Eb/N_0)	4 dB

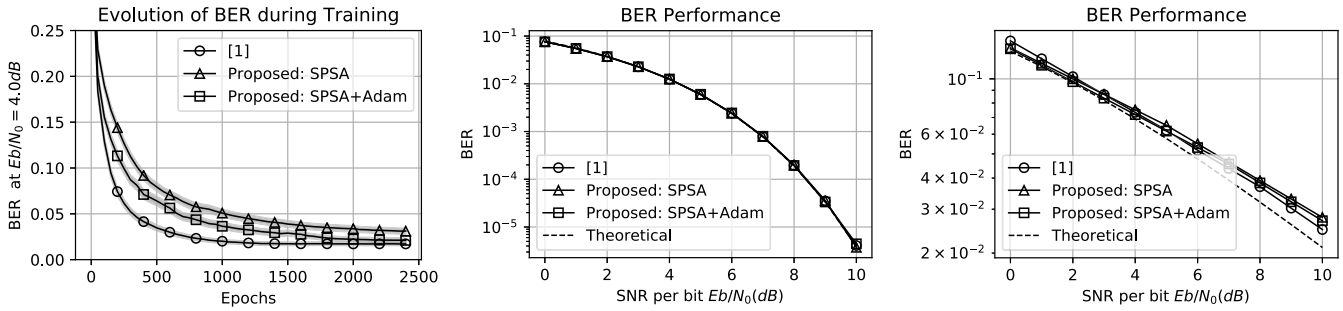
III. RESULTS

To validate the usefulness of the proposed approach, we consider the setting used in [1] for learning the modulation and coding scheme. We provide results for a QPSK system with Gray coding over a single use of I-Q channel. We use a variant of SGD known as *Adam* [15] for training the receiver layers in the case of proposed hybrid method. The parameters used for training are given in Table I.

For evaluation of the training efficiency of the proposed methods, we provide the BER evolution during the training phase, when trained on AWGN channel at an SNR of $Eb/N_0 = 4dB$, in Fig. 2a. The result provided is the average observed performance of 250 independent models trained with each of the discussed training methods. A validation set of 10000 symbols is used for monitoring the progress of training the models. We can observe that the training method which assumes the knowledge of the channel [1] is able to converge early. However, the proposed *SPSA+Adam* is able to match the performance of [1] with only slightly more training epochs. The method using SPSA alone to optimize both transmitter and receiver layers takes more training epochs to converge. This is expected as the hybrid method accelerates the training by using actual error gradients for optimizing receiver layer while using approximate gradient from SPSA for optimizing transmitter layer. As the associated error bars provided for all the cases are of similar magnitude, we can conclude that, even though the proposed approaches, utilize only approximate error gradients, they can perform stable training.

In Fig. 2b, we provide the BER performance of the models trained in AWGN channel. For calculating BER, we first

Authorized licensed use limited to: Indian Institute of TechnologyDharwad. Downloaded on July 14,2022 at 09:59:15 UTC from IEEE Xplore. Restrictions apply.



(a) BER evolution during training (AWGN) (b) AWGN Channel (c) Rayleigh Block Fading Channel

Fig. 2. Performance of trained models.

TABLE II
TIME TAKEN FOR TRAINING EACH MODEL

Channel Model	Training Method	Training Time (sec)
AWGN Channel	[1]	8.192 ± 0.060
	Proposed: SPSA	5.961 ± 0.016
	Proposed: SPSA+Adam	7.337 ± 0.025
Rayleigh Block Fading	[1]	9.713 ± 0.110
	Proposed: SPSA	6.748 ± 0.255
	Proposed: SPSA+Adam	8.001 ± 0.021

calculated SER from the samples and then used the standard conversion for SER to BER assuming Gray coding. We can observe that even without the knowledge of exact error gradients, the proposed methods are able to meet the theoretical BER for AWGN channel. This result show that the proposed methods can achieve theoretical results even without any assumption about the channel model. However, we would like to note that the proposed hybrid approach required slightly more training epochs that the method in [1], a price that is paid for being agnostic to the channel model.

In order to evaluate the usability of proposed methods under fading channel, we conducted experiments on Rayleigh block fading channel. We compare the proposed approaches with [1] in Fig. 2c. We can see that all the methods are able to closely follow the theoretical BER.

Based on an elaborate study conducted on fine tuning the parameters of the algorithms, we found that, by setting Adam parameters to the widely accepted thumb rule ($\beta_1 = 0.99, \beta_2 = 0.999$), the trained models reached a performance competitive to theoretical results faster. For SPSA, we found that lower values of c and γ (as given in Table I) are able to provide faster convergence.

We trained our model on a $i7@3.6GHz$ CPU with $16GB$ RAM, running Linux. The observed training time for each technique, averaged over 250 models, is given in Table II. During the training process, we observed that both the SPSA optimizer and hybrid optimization approach took less time to train than *Adam* optimizer. This could be because *Adam* operates by calculating exact gradients at each node in the computation graph which is a computationally expensive operation, while SPSA operates by inducing perturbations which is relatively inexpensive operation.

IV. CONCLUDING REMARKS

In this work, we proposed a practical way to train an end-to-end communication system using deep learning. By using approximate gradient method, we demonstrated how to train a communication system when the channel model is not available or is so complex that gradient computation is difficult. The proposed SPSA method can be parallelized to gain a further improvement in training time (wall clock time). It will also be interesting to see if one can improve SPSA by including momentum like higher order characteristics of calculated gradient as in *Adam* to accelerate the rate of convergence.

REFERENCES

[1] T. J. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[2] D. H. Ballard, "Modular learning in neural networks," in *Proc. AAAI*, 1987, pp. 279–284.

[3] M. Kim, N.-I. Kim, W. Lee, and D.-H. Cho, "Deep learning-aided SCMA," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 720–723, Apr. 2018.

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

[5] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 1, pp. 114–117, Feb. 2018.

[6] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," in *Proc. IEEE 18th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2017, pp. 1–5.

[7] A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. ten Brink. (2018). "OFDM-autoencoder for end-to-end learning of communications systems." [Online]. Available: https://arxiv.org/abs/1803.05815

[8] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132–143, Feb. 2018.

[9] T. J. O'Shea, T. Roy, N. West, and B. C. Hilburn. (2018). "Physical layer communications system design over-the-air using adversarial networks." [Online]. Available: https://arxiv.org/abs/1803.03145

[10] T. J. O'Shea, T. Roy, and N. West. (2018). "Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1805.06350

[11] H. Ye, G. Y. Li, B.-H. F. Juang, and K. Sivanesan. (Jul. 2018). "Channel agnostic end-to-end learning based communication systems with conditional GAN." [Online]. Available: https://arxiv.org/abs/1807.00447

[12] F. A. Aoudia and J. Hoydis. (2018). "End-to-end learning of communications systems without a channel model." [Online]. Available: https://arxiv.org/abs/1804.02276

[13] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins APL Techn. Dig.*, vol. 19, no. 4, pp. 482–492, 1998.

[14] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 332–341, Mar. 1992.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

Authorized licensed use limited to: Indian Institute of TechnologyDharwad. Downloaded on July 14,2022 at 09:59:15 UTC from IEEE Xplore. Restrictions apply.