Avoiding normalization uncertainties in deep learning architectures for end-to-end communication

Avoiding normalization uncertainties in deep learning architectures for end-to-end communication

Simon Bos, Evgenii Vinogradov, Sofie Pollin
Department of Electrical Engineering, KU Leuven, Belgium
Email: {simon.bos, evgenii.vinogradov, sofie.pollin}@kuleuven.be

***Abstract*—Recently, deep learning is considered to optimize the end-to-end performance of digital communication systems. The promise of learning a digital communication scheme from data is attractive, since this makes the scheme adaptable and precisely tunable to many scenarios and channel models. In this paper, we analyse a widely used neural network architecture and show that the training of the end-to-end architecture suffers from normalization errors introduced by an average power constraint. To solve this issue, we propose a modified architecture: shifting the batch slicing after the normalization layer. This approach meets the normalization constraints better, especially in the case of small batch sizes. Finally, we experimentally demonstrate that our modified architecture leads to significantly improved performance of trained models, even for large batch sizes where normalization constraints are more easily met.**

***Index Terms*—End-to-end learning, deep learning, neural network, autoencoder, modulation**

I. Introduction

Digital communication is considered as a complex and mature engineering field in which modeling and tractable analytical models have shown to enable incredible solutions. Currently, digital communication systems are partitioned in independent blocks which are individually optimized for their specific tasks (e.g., source & channel encoding or modulation) This makes the digital communication systems of today versatile and controllable, but could lead to a suboptimal end-to-end performance. Additionally, most signal processing algorithms are optimized for tractable analytical (channel) models while relying on some assumptions. The performance of the system could be lower under practical conditions. The application of deep learning to an end-to-end digital communication system shows potential to solve these issues [1].

The idea of using deep learning for end-to-end digital communication was first proposed in [1] and has been shown to have a competitive performance on additive white Gaussian noise (AWGN) and Rayleigh fading channels. Their proposed general end-to-end system architecture is outlined in Fig. 1 (a more detailed architecture is found in Fig. 2). The aim of this system is to transmit a number of bits K over the channel which the receiver can reconstruct.

Every permutation of K bits is represented by a corresponding message m_i, $\forall i \in I = \{1,2,\ldots,M\}$ with $M = 2^K$. The transmitter applies the function[1,2] $f_T : I \to \mathbb{C}$ to the message identified by i to generate a transmitted signal x_i. Values of x_i are usually constrained by the hardware. Next, the channel is described by its conditional probability density function $p(y|x_i)$, with y the received signal. The receiver applies the function $f_R : \mathbb{C} \to I$ to y to give an estimate $\hat{i}$ for the message that was transmitted.

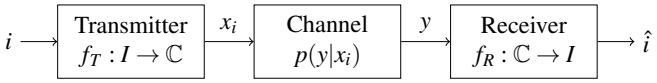

Fig. 1. General end-to-end digital communication system architecture.

The transmitter and receiver functions f_T and f_R are represented by a (deep) neural network (hardware constraints are represented by a normalization layer). This system is trained in a supervised fashion by uniformly sampling a batch of message identifiers, passing it through the system, computing the estimation error and updating the weights of both networks using backpropagation to obtain a minimal end-to-end estimation error. In this approach, performing backpropagation over the channel requires a differentiable model of the channel in which the system will be deployed. This model should include all possible real-world effects; otherwise the end-to-end performance is poor [2]. However, in practice only simplified channel models are available. Therefore, this approach cannot fully unlock the potential of deep learning.

Several works tackled the limitation of the original approach: [2] investigated over-the-air fine-tuning, [3], [4] applied generative adversarial network to create a differentiable channel model and [5] proposed a method based on policy gradients. A disadvantage of the neural networks used in [1]–[5] is that only an approximation of the normalization constraints is used while training the networks. As we show in Sections III and IV, this leads to suboptimal trained models, even for larger batch sizes where the normalization approximation is improved.

In this article, we propose a modified version of the end-to-end architecture presented in [1] to better adhere to the normalization constraints; especially in the case of small batch sizes. The rest of this paper is organized as follows. In Section II, a more detailed version of Fig. 1 is explained. In Section III, the proposed modified architecture is outlined together with the rationale. In Section IV, a comparison of performance between system architectures is conducted and the conclusions are drawn in Section V.

[1]Complex numbers are represented by $\mathbb{R}^2$ in most implementations, due to unsupported complex operations in deep learning frameworks.

[2]In [1], an extension to multiple channel uses is considered (i.e. $\mathbb{C}^n$).

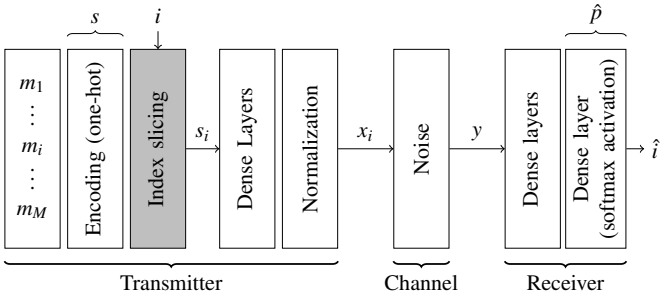

Fig. 2. Detailed end-to-end communication system over an AWGN channel, as proposed in [1].

II. Detailed end-to-end communication system

Fig. 2 shows the more detailed architecture for an AWGN channel presented in [1]. In its first part, the transmitter applies pre-processing to obtain s_i, the one-hot encoded version of message m_i (i is the input to the model). In the second part, the transmitter consists of a sequence of fully-connected/dense layers with a Rectified Linear Unit (ReLU) activation function for the first layers and a linear activation function for the last layer (which consists of 2 neurons representing $\mathbb{C}$). Afterwards, normalization is performed to ensure either a fixed power constraint $|x_i|^2 = P,\ \forall i \in I$ or an average power constraint $\mathrm{E}\left[|x_i|^2,\ \forall i \in I\right] = P$ for some fixed transmitter power P.

The AWGN channel is represented by an additive noise layer with noise variance σ^2 derived from a fixed signal-to-noise ratio (SNR) defined as $SNR \triangleq P/\sigma^2$. Note that SNR and P can be considered implicit inputs of the end-to-end system.

The receiver consists of a sequence of fully-connected layers with a ReLU activation function for the first layers, but a softmax activation function at the last layer. The output of this layer $\hat{p}$ is a probability vector over all messages, and the index with the highest probability is taken as $\hat{i}$.

The end-to-end system is trained in a supervised fashion by firstly uniformly sampling a batch B of message indices from I, secondly computing the batch estimations by applying the full end-to-end model, thirdly computing the categorical cross-entropy[3] of s_i and their corresponding $\hat{p}$, $\forall i \in B$ and lastly performing stochastic gradient descent and backpropagation to update the model weights. This training step is repeated until a stop criterion is reached.

III. Modified architecture of end-to-end communication system

In training neural networks, hyperparameters such as learning rate and batch size can influence the training speed and resulting accuracy notably. Likewise, the authors of [1] noted that increasing the batch size during training helps to improve the accuracy of the learned model. As a matter of fact, in the training method outlined above, the batch size influences the normalization layer heavily. For a fixed power constraint

[3]As mentioned in [1], the sparse categorical cross-entropy between i and $\hat{p}$ is effectively the same loss function. However, with this loss the one-hot encoding layer can be replaced with an *embedding* layer.

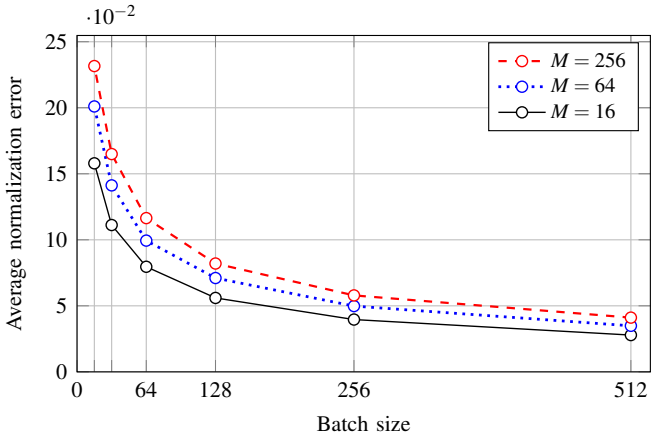

Fig. 3. Average normalization error for various batch sizes and number of messages M. Experiment performed on randomly initialized transmitter networks, with batches drawn uniformly from I, and using a fixed $E_b = 1$, i.e. $P = \log_2(M)$, to enable a fair comparison for varying M. An average of 30 transmitter network initializations (comprising 2 hidden layers with 60 neurons each) and 1000 uniformly sampled batches is presented.

$|x_i|^2 = P,\ \forall i \in I$, the normalization factor is calculated independently for each batch element; and therefore the effects of the normalization layer are independent of the batch size. For an average power constraint $\mathrm{E}\left[|x_i|^2,\ \forall i \in I\right] = P$, this is not the case. In this case, normalization is performed to ensure an average power for the specific batch B with batch size B_s:

$$\frac{1}{B_s}\sum_{i\in B}|x_i|^2 = P.$$

However, for some batches this constraint may not reflect the desired normalization $\mathrm{E}\left[|x_i|^2,\ \underline{\forall i \in I}\right] = P$. This effect is visualized in Fig. 3. By picking the batch size too low, in general the batch does not represent I and the average normalization error[4] of the batch will be high. For a higher number of messages M, the batch size should be increased to have a constant average normalization error.

A high batch size relative to the number of messages M seems to solve this normalization error issue. However, this is not always convenient, especially for a higher number of messages. To counter normalization errors in another way, we propose changing the architecture of Fig. 2 to the architecture of Fig. 4. By shifting the position of the batch slicing after the normalization layer, normalization is performed on the messages $m_i,\ \forall i \in I$, and hence normalization will ensure the desired constraint of $\mathrm{E}\left[|x_i|^2,\ \underline{\forall i \in I}\right] = P$. Therefore, the proposed architecture has no normalization errors and normalization is performed independently of any batch size. The training of this proposed system is also modified slightly: backpropagation should now be computed over a slicing layer. Frameworks such as Tensorflow [6] (used in our experiments) have built-in support for these kinds of operations.

[4]We define the average normalization error of a batch B as $(\sum_{i\in B}|x_i - x_i'|)/B_s$ with x_i and x_i' the transmitted symbols computed by applying the full transmitter network to respectively B and I.

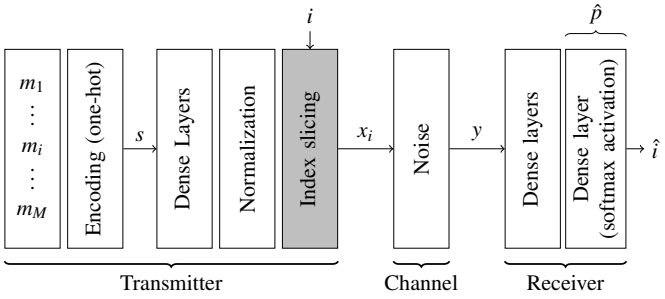

Fig. 4. We propose shifting the position of the batch slicing in the end-to-end architecture after the normalization layer.

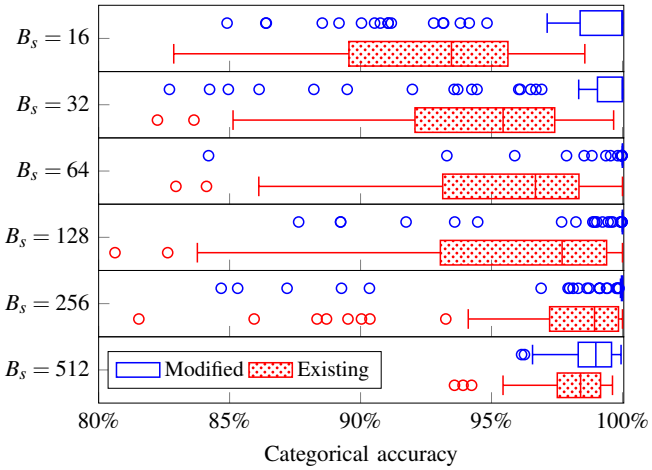

Fig. 5. Comparison of validation accuracy (averaged over 30 batches) after training both the existing architecture of Fig. 2 and our modified architecture of Fig. 4. Training is performed for various batch sizes B_s and a fixed $M = 128$, $SNR = 45$ dB, 76.800 data points (split in batches), with Adam optimizer with learning rate 0.008 and receiver and transmitter networks both comprising 2 dense layers with 100 neurons each. To have representative results, training is performed for 10 random network initializations and 10 data generators.

IV. Experiments

The aim of this section is to experimentally determine the influence of our modified architecture on the training and the performance of the model. To this end, we train both architectures on the same batches, while having the same network initializations. At the end of training, the categorical accuracy of the networks is determined. To only measure the effect of normalization errors while training, the validation set is chosen such that the normalization error is 0. Fig. 5 shows the results for a fixed $M = 128$ and $SNR = 45$ dB, but for other configurations we noticed similar results. Note: for $B_s \in \{16, \ldots, 256\}$, the median of the categorical accuracy for the modified architecture is 100%. Also, due to the low error, only outliers can be seen for $B_s = \{64, 128, 256\}$.

We make three main observations. Firstly, for each architecture, training on a higher batch size B_s (up to $B_s = 256$) on average leads to a better performing model. This is expected, since a batch size which is too small usually leads to an unstable learning curve. For a batch size $B_s = 512$, the accuracy drops slightly. In this case, we inspected the learning curve and noticed that it did not yet converge; which is explained by the fact that at a higher batch size B_s, the number of gradient updates are lower for a fixed number of data points. For lower batch sizes, the learning curves have converged: either to a stable constant 100% accuracy or oscillating around a sub-optimal solution with less than 100% accuracy.

Secondly, the accuracy of the resulting model increases significantly with our modified architecture. Normalization errors clearly have a negative impact on the training of end-to-end communication systems. This is clarified as follows: in the existing architecture, the receiver should not only learn to cope with an additive noise channel, but also with the introduced normalization errors. Therefore, the extra normalization errors make the learning task harder for the receiver.

Thirdly, with an increase in batch size B_s, the accuracy difference induced by our proposed modification diminishes. This is explained by the decline in normalization error with a growing batch size (see Fig. 3). Nevertheless, the accuracy difference remains significant, and therefore, using our proposed modification scales better in terms of batch size. This is especially important for a higher number of messages, which require a higher batch size to reduce the normalization errors.

V. Conclusion

In this paper, we have focused on the architecture of an end-to-end communication system and have proposed shifting the position of the batch slicing after the normalization layer. The advantage of the proposed modification is that it does not incur normalization errors when training the end-to-end system. We showed that normalization errors make the learning task of the receiver harder. Therefore, in general using our proposed modification leads to significantly improved performance of trained models than existing architectures (trained for similar learning configurations). Furthermore, the training of our modified architecture scales much better in terms of the batch size: even at batch sizes lower than the number of messages, performance remains competitive.

Acknowledgment

This research has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 101017171 (MARSAL project).

References

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

[2] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE JSTSP*, vol. 12, no. 1, pp. 132–143, 2018.

[3] T. J. O'Shea, T. Roy, and N. West, "Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks," in *ICNC'19*, 2019, pp. 681–686.

[4] H. Ye, G. Y. Li, B.-H. F. Juang, and K. Sivanesan, "Channel agnostic end-to-end learning based communication systems with conditional GAN," in *2018 IEEE GC Wkshps*, 2018, pp. 1–5.

[5] F. A. Aoudia and J. Hoydis, "Model-free training of end-to-end communication systems," *IEEE JSAC*, vol. 37, no. 11, pp. 2503–2516, 2019.

[6] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/