Machine Learning Autoencoder Applied to Communication Channels

Eduardo Dadalto Camara Gomes
Department of Electronics, Optronics and Signal Processing (DEOS)
ISAE-SUPAERO
Toulouse, France
Email: edadaltocg@gmail.com

***Abstract*—Communication channel error correction is key to enable digital critical digital communication systems to works efficiently. Although noisy channels has already a mathematically proven optimal solution for error error rate minimization: the maximum a posteriori (MAP) decoder; it's implementation cripples system's applicability, since it introduces a large delay for large codewords. In this context, a deep neural network (DNN) is proposed to optimize the channel with an end-to-end autoencoder. The DNN autoencoder outperforms MAP decoder in terms of delay, because of its *one-shot* capability when it is trained. Moreover, it has similar BER compared to MAP for a range of signal to noise ratio (SNR).**

***Index Terms*—communication system, machine learning, autoencoder, channel decoding, maximum a posteriori (MAP) decoder**

I. Introduction

A. Motivation

Communication systems are present in everyday life and have different forms. These designs are less or more complex; nevertheless, they share a common core. The simpler architecture defines the system as an exchange of information between two terminals through a noisy channel. As a result of channel imperfections, Shannon theorized [1] in 1948 the ultimate reliable data rate that can be transmitted through a communication system.

Since then, the research community in digital communication developed a series of algorithms to minimize bit error rate (BER) over a channel. However, the challenge of finding an *efficient* solution (i.e., at same time low latency and with low error probability) for low signal to noise ratio (SNR) channels, remains. Based on this vision, Tim O'Shea and Jakob Hoydis [2] pertinently noted that traditional algorithms in the field have foundations in probability theory (e.g. maximum a posteriori (MAP) estimation used as a channel decoder). Hence, they are usually built on top of mathematically convenient models. Even though they are theoretically optimal error corrector machines, these models oftenly does not account for all the real system's imperfections, what leads to errors when in practice.

Accordingly, machine learning (ML) algorithms does not require rigidly designed models and can take non-linearities effortlessly into account. These characteristics make these algorithms candidates for being used as bit error corrector. Moreover, with ML based channel encoder and channel decoder, the design of communication systems as independently working blocks becomes obsolete, as a deep neural network (DNN) is able to actuate end-to-end in the system. As a result, an optimized autoencoder with a stochastic layer that models channel's imperfections can substitute the block based representation. Therefore, ML based communication systems could be a better representation of realistic systems and could optimize information transmission of different blocklengths and with low decoding latency, resulting ultimately in gain of bandwidth over standard methods.

Low latency and high bandwidth wireless communication are key to critical systems, such as airplanes, satellites, cellular communication and 5G operations. The latter was studied by F. D. Calabrese et al. in [3] which demonstrated that individual based radio resource management (RRM) algorithms were outperformed by a general learning framework, resulting in significant expense reductions, while increasing performance of the network. Thereby, ML algorithms are cardinal for state of the art communication applications.

B. Related work

Recently, a significant amount of work in radio communication theory has emerged, introducing ML elements to the communication system. O'Shea et al. in [13] developed a channel autoencoder with optimized impairment and regularization layers to emulate channel impairments. They studied this architecture over a binary symmetric channel (BSC) and others, founding "some promising initial capacity" for this schema. In their research, results in terms of BER over SNR for a DNN based autoencoder and for a convolutional neural network (CNN) based autoencoder were treated. They used a range of SNR - from -10dB to 15dB - with QPSK and QAM16 modulation as benchmarks. This analysis was conducted for a binary input message.

T. Gruber et. al. in a 2017 paper [14] proved that a deep learning-based channel decoder could actually learn a decoding algorithm rather than just being a simple classifier. They introduced codewords that were not been used in the training set, and the trained NN was able to correctly decode it. They also observed that structured codes are easier to learn than unstructured ones. NN for structured codes are able to generalize to the full codebook even if they have not

seen all the training examples. They trained the NN for very short blocklengths ($N \leq 64$) in order to compare with MAP decoding performance.

C. Problem statement

This research will implement a ML autoencoder for a BSC that performs similar to the MAP decoder in real applications for a range of SNR from -10dB to 10dB. In its most simple form, a channel autoencoder includes an encoder, a noisy channel and an decoder. Using state-of-the art DNN algorithms to find the best solution of the problem, this work hopes to contribe to set up a higher standard in terms of performance in bit-error correction and reduced delay for digital communication applications. In a near future, this disruptive methodology for error correction using ML could replace mathematically optimal decoders which are the current guideline.

II. Theoretical Background

A. Maximum a posteriori decoder

The concept of a MAP decoder or Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm is choosing a message which maximizes the MAP probability of the corrupted information received by the decoder [4]. This algorithm is known for its optimality in error correction for white noise interference and it's variants: Log-MAP and Max-Log-MAP, are widely used as decoders for turbo codes [7].

The MAP decoder is practically nonviable to implement [6]. Thus, its variants cited in the paragraph above are better suited for practical cases. The Log-MAP is an optimal decoder, having equivalent performance to the MAP decoder. While the Max-Log-MAP is a suboptimal decoder and will not be treated in this paper. Refer to [6] for specific details in the derivation of the Log-MAP algorithm.

Turbo decoding using Log-MAP decoder for an additive white Gaussian noise (AWGN) channel and for a BSC was studied in [5]. They analyzed the BER for small SNR and concluded that the results are sensible to SNR estimation of the real channel, since the estimation of this parameter is required for the metric calculation of the algorithm [6]. It means that if the real SNR is smaller, the decoder will not perform well. Hence, for channel characteristics that change over time, the application of a MAP algorithm is debatable. This conclusion serve as motivation for ML autoencoders, which could adapt to different SNR conditions, outperforming the MAP decoder for realistic applications.

B. Neural network basics

Neural networks (NN) are a set of connected *neurons* able to approximate any kind of function through an architecture composed of several single processing units (neurons) connected together forming a network. They are defined by [9] as parallel distributed, learning- and self-organizing information processing systems.

In the context of learning algorithms implemented by a NN, the key issue is to find a suitable architecture that delivers the best results. To define and build this infrastructure, the activation function, the loss function and the structure must be decided. M. Nielson [8] in his book explores a vast amount of NN architectures and how they work.

Multi-layer feed forward neural network (MLNN) as shown in Fig. 1 will be applied in this work. They are known for the universal approximation property [9] and for the versatility of increasing the number of layers, creating a DNN able to undertake complex classification problems with low classification error and low dimensionality. In more general terms, a MLNN with L layers, also called *depth*, and parameter θ is a mapping of an input $\mathbf{r}_0 \in \mathbb{R}^{N_0}$ to an output $\mathbf{r}_L \in \mathbb{R}^{N_L}$ through L iterative steps. For a fully-connected network, each layer vector is calculated iteratively in the forward propagation. Each neuron is composed of a linear combinator and an activation function defined as

$$f_l\left(\mathbf{r}_{l-1}; \theta_l\right) = \sigma\left(\mathbf{W}_l \mathbf{r}_{l-1} + \mathbf{b}_l\right) \tag{1}$$

where $\mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}$ is the weight matrix between layers $l-1$ and l, $\mathbf{b}_l \in \mathbb{R}^{N_l}$ is the bias vector and σ is the activation function [2].

In order to accurately classify an input, a *perceptron* processing unit is chosen to solve a classification problem for each input message. A perceptron maps a binary vector input into a single binary output digit [8]. Mathematically, it means that the function defined in (1), outputs 1 or 0, depending on a threshold value.

For a classification problem in communication, the *categorical cross-entropy* loss function $l(\mathbf{p}, \mathbf{q}) : \mathbb{R}^{N_L} \times \mathbb{R}^{N_L} \mapsto \mathbb{R}$ defined in (2) is most common. It is derived from the log-likelihood of a training set where q_j is the estimated probability of the outcome j and p_j is the true probability. Basically, the function measures the dissimilarity between p_j (what you expected) and q_j (what you obtained) [10] where $j = 1, ..., N_L$.

$$l(\mathbf{p}, \mathbf{q}) = -\sum_j p_j \log\left(q_j\right) \tag{2}$$

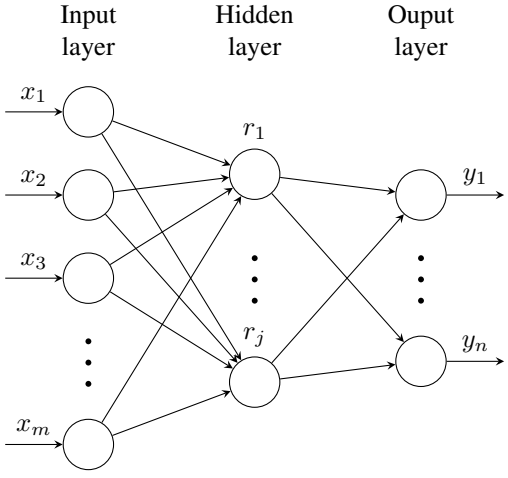

Fig. 1. MLNN representative diagram. Where $\mathbf{x}$ is the input vector of the NN, $\mathbf{r}_l$ is a hidden layer vector and $\mathbf{y}$ is the output vector.

With all these elements set, training the neural network to calculate an accurate weight matrix $\mathbf{W}$ for forward propagation requires a labeled *training data set*. This set is composed of pairs $(\mathbf{r}_{0,i}, \mathbf{p}_i)$, where $i = 1, ..., S$ and S is the number of training sets. It matches the input and its respective desired output. The objective is to minimize the overall loss function defined in (3) in terms of the parameter θ [2].

$$L(\theta) = \frac{1}{S} \sum_{i=1}^{S} l\left(\mathbf{p}_{L,i}, \mathbf{r}_{L,i}\right) \tag{3}$$

This problem is an optimization problem and can be solved through different algorithms. For NN, an efficient way of computing this minimization is implementing the back-propagation (BP) algorithm. BP is classified as a supervised learning algorithm that is able to optimize a function based on some parameter and is highly parallelizable in computational terms [8] [9].

The great advantage of NN is its ability do generalize the training set, delivering the right results even for data not contained in the training set. With the weight matrix trained, the NN find the correct putputs for unseen inputs. In order to validate this principle, a *validation data set* is used and the loss is mesured. Based on this value, we can evaluate the NN overall accuracy.

For implementing all the algorithms, Keras, a high-level python API for ML, will be used together with TensorFlow 1.13.1, a ML python friendly open-source library [11] [12]. Both are available for free.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[2] O'Shea, Tim & Hoydis, Jakob, "An Introduction to Machine Learning Communications Systems", 2017.

[3] F. D. Calabrese, L. Wang, E. Ghadimi, G. Peters, and P. Soldati, "Learning radio resource management in 5G networks: Framework, opportunities and challenges," arXiv preprint arXiv:1611.10253, 2016

[4] W. Alexander, H. Peter & W. Norbert, "Turbo-Decoding Without SNR Estimation",IEEE Communications Letter, vol. 4, no. 6, pp. 193-195, 2000.

[5] M. Jordan and R. Nichols, "The effects of channel characteristics on turbo code performance," in Proc. Milcom'96, McLean, VA, Oct. 1996, pp. 17–21.

[6] Robertson, P. , Hoeher, P. and Villebrun, E. (1997), Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding. Eur. Trans. Telecomm., 8: 119-125. doi:10.1002/ett.4460080202.

[7] J. Hagenauer,P. Robertson, L. Papke: iterative ("Turbo") decoding of systematic convolutional codes with the MAP and SOVA algorithms. In: ITG-Fachbericht 130. October 1994, p. 21-29.

[8] M. Nielsen, *Neural Networks and Deep Learning*.

[9] I. Mohamed, "Applications of neural networks to digital communications - a survey", 1997.

[10] M. Kevin, "Machine Learning: A Probabilistic Perspective", MIT, ISBN 978-0262018029, 2012.

[11] M. Abadi, A. Agarwal, P. Barham, et al., "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015. Software available from tensorflow.org.

[12] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[13] T. O'Shea, K. arra, T. C. Clancy, "Learning to Communicate: Channel Auto-encoders, Domain Specific Regularizers, and Attention", 2016.

[14] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning based channel decoding," accepted for CISS 2017, arXiv preprint arXiv:1701.07738, 2017.