Machine Learning Autoencoder Applied to Communication Channels

E. Dadalto Camara Gomes[1]

[1]ISAE-SUPAERO
Université de Toulouse
31055, Toulouse, France
Email: eduardo.dadalto-camara-gomes@student.isae-supaero.fr

ISAE-SUPAERO, 2019

Outline

Context I

Digital communication system definition

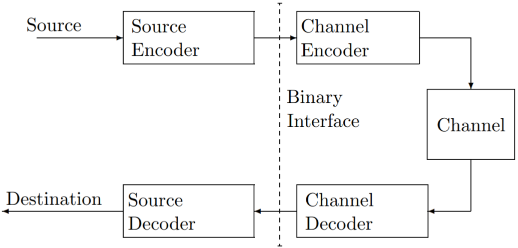

Figure: Block diagram representation of a communication system.

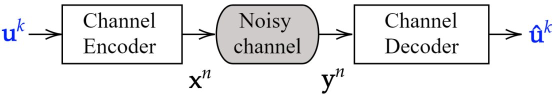

Figure: Simplified communication system. Where $\mathbf{u}^k$ is a source message, $\mathbf{x}^n$ a code word, $\mathbf{y}^n$ the received code word and $\hat{\mathbf{u}}^k$ the decoded message.

Context II

Machine learning applied to communication system

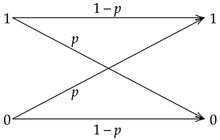

Figure: Binary symmetric channel (BSC) representation.

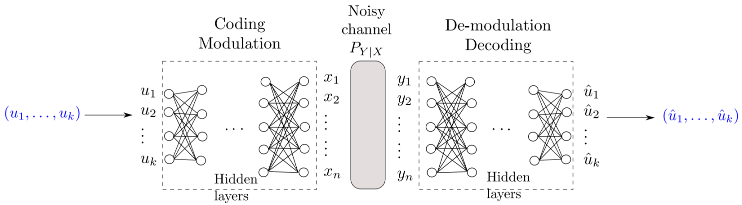

Figure: Deep neural network (DNN) based autoencoder.

Motivation & Challenges

- Low latency and high bandwidth wireless communication are key to critical systems.
  - i.g. airplanes, satellites, cellular communication and 5G operations.

Motivation & Challenges

- Low latency and high bandwidth wireless communication are key to critical systems.
  - i.g. airplanes, satellites, cellular communication and 5G operations.
- As opposite to structured algorithms, machine learning (ML) algorithms do not require rigidly designed models and can take non-linearities effortlessly into account.

Motivation & Challenges

- Low latency and high bandwidth wireless communication are key to critical systems.
  - i.g. airplanes, satellites, cellular communication and 5G operations.
- As opposite to structured algorithms, machine learning (ML) algorithms do not require rigidly designed models and can take non-linearities effortlessly into account.
- ML based communication systems could be a better representation of realistic systems and could optimize information transmission of different blocklengths.
  - It translates to lower decoding latency, resulting ultimately in gain of bandwidth over standard methods.

Problem Statement

- Could ML based decoders and autoencoders for a BSC perform similarly to the MAP decoder in real applications and have lower communication delay?

Problem Statement

- Could ML based decoders and autoencoders for a BSC perform similarly to the MAP decoder in real applications and have lower communication delay?
- This work aims to contribute to set up a higher standard in terms of performance in bit-error correction and demonstrate that ML based models can reduce delay for digital communication applications.

- The concept of a MAP decoder algorithm for sequences is choosing a message which maximizes the a posterior probability.

$$f(y) = \arg\max_{x \in \mathcal{X}} P_{X|Y}(x|y)$$

Algorithm 1 MAP rule for BSC and linear block code.

Input: received block $\mathbf{y}^n \in \{0,1\}^n$, code word set $\mathcal{X}$ and generator matrix $G_{k \times n}$.

Output: message estimation $\hat{\mathbf{u}}^k \in \{0,1\}^k$.

procedure MAP DECODER($y, \mathcal{X}, G$)

  $p \leftarrow$ channel crossover probability

  for i in $range(2^k)$ **do**

    distances$[i] \leftarrow d_H(\mathbf{y}, word[i] \in \mathcal{X})$

  $\hat{\mathbf{x}} \leftarrow argmin$(distances)

  $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{x}} G^{-1}$ **return** $\hat{\mathbf{u}}$

Neural Network's Design and Architecture Basics

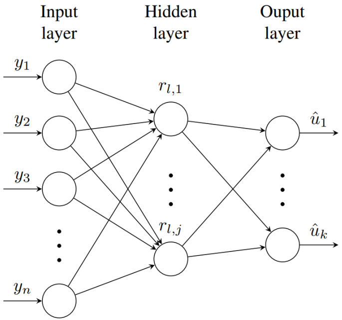

Figure: NN representative diagram, where $\mathbf{y}^n$ is the input vector, $\mathbf{r}_l^j$ is a hidden layer vector and $\hat{\mathbf{u}}^k$ is the output vector.

Array Decoder DNN Architecture and Implementation

Table: DNN array decoder architecture and training parameters. The input is a code word of size n and the output is the decoded message of size k.

Decoder	Dense: 128, activation: ReLU, input size: n
	Dense: 64, activation: ReLU
	Dense: 32, activation: ReLU
	Dense: k, activation: Sigmoid
Total parameters: 12776	

Loss function	Optimizer	N. Epochs	Batch Size
Binary cross-entropy	Adam	2^{16}	256

One-hot Decoder DNN Architecture and Implementation

Table: DNN one-hot decoder architecture and parameters. The input is a code word of size n and the output is a unique one-hot vector of size 2^k which is mapped into its correspondent message.

Decoder	Dense: 256, activation: Softmax, input size: n
Total parameters: 4352	

Loss function	Optimizer	N. Epochs	Batch Size
Binary cross-entropy	Adam	2^{14}	256

DNN Autoencoder Design

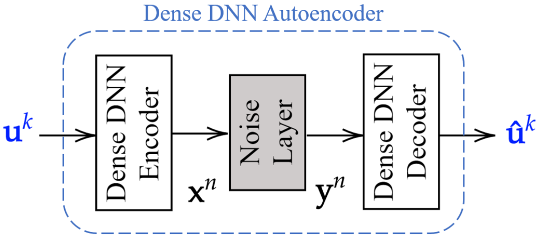

Figure: Representation of a DNN autoencoder composed of dense layers. Each block represents a DNN which together learn the channel encoding and decoding in an end-to-end manner. They also operate together in bit error correction.

Array Autoencoder DNN Architecture and Implementation

This design yielded best error correction results out of 80 models tried

Table: DNN array autoencoder architecture and training parameters.

Encoder	Dense: 512, activation: ReLU, BN[1], input size: 8 Dense: 256, activation: ReLU, BN Dense: 16, activation: Sigmoid
Channel	Lambda: $Round(\mathbf{x})$, input size: 16 Lambda: $\mathbf{x} \oplus$ noise
Decoder	Dense: 128, BN, input size: 16 Dense: 64, activation: ReLU, BN Dense: 8, activation: Sigmoid
Total parameters: 154072	

Loss function	Optimizer	N. Epochs	Batch Size
MSE	Adam	2^{17}	256

[1] Batch Normalization (BN)

One-hot Autoencoder DNN Architecture

This design yielded best error correction results out of 71 models tried

Table: DNN one-hot autoencoder architecture and training parameters.

Encoder	Dense: 196, activation: ReLU, BN, input size: 256
	Dense: 128, activation: ReLU, BN
	Dense: 96, activation: ReLU, BN
	Dense: 64, activation: ReLU, BN
	Dense: 32, activation: ReLU, BN
	Dense: 16, activation: Sigmoid
Channel	Lambda: $Round(\mathbf{x})$, input size: 16
Decoder	Dense: 128, activation: ReLU, BN, input size: 16
	Dense: 256, activation: Softmax
Total parameters: 134052	

Loss function	Optimizer	N. Epochs	Batch Size
MSE	Adam	2^{16}	256

DNN Array Decoder Error Correction Performance

For each of 20 different channels, 100000 messages were sent. The average error was calculated and traced. The same procedure was applied throughout the work.

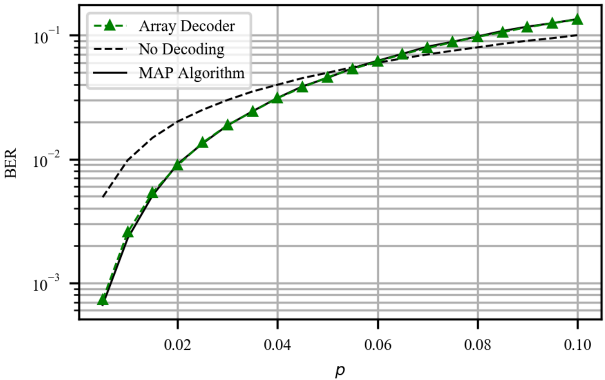

Figure: Array decoding BER performance. DNN trained with a channel crossover probability error of $p_t = 0.07$. The decoder successfully learned the reference MAP algorithm.

DNN One-hot Decoder Error Correction Performance

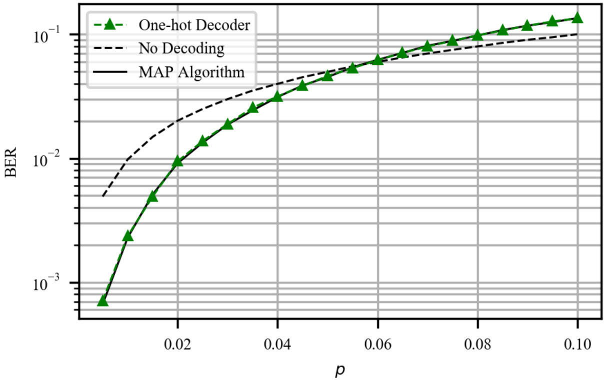

Figure: One hot decoding BER performance. NN decoder trained with a channel crossover probability error of $p_t = 0$. The decoder successfully learned the reference MAP algorithm.

DNN Array Autoencoder Error Correction Performance I

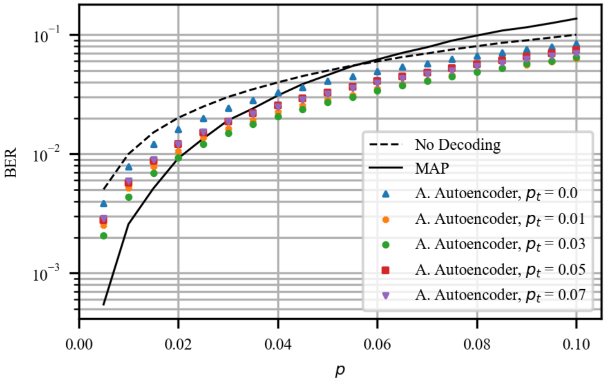

Figure: Training crossover probability simulation for the array autoencoder. $P_t = 0.03$ demonstrated to have best performance to this particular architecture.

DNN Array Autoencoder Error Correction Performance II

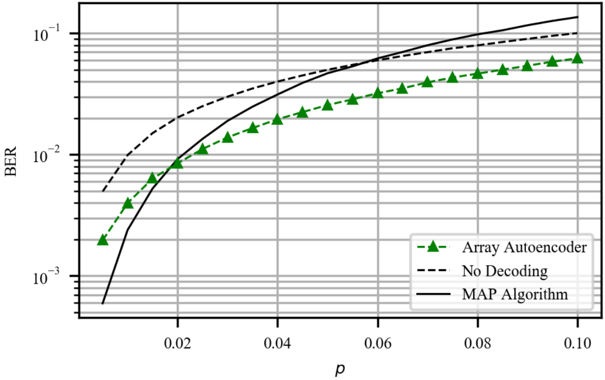

Figure: Array autoencoder BER performance. DNN array autoencoder trained with a channel crossover probability error of $p_t = 0.03$. The array autoencoder could not outperform the reference MAP algorithm for channels with $p < 0.02$.

DNN One-hot Autoencoder Error Correction Performance

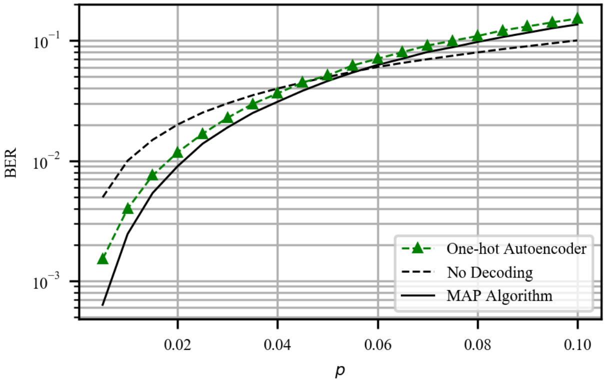

Figure: One-hot autoencdoer BER performance. Trained without a noisy channel. No architecture could outperform the reference MAP algorithm in terms of BER.

Delay Time Analysis

Calculations done by CPU

Table: Decoding time comparison between the reference MAP algorithm and the DNN decoders and autoencoders. The data is normalized to the average MAP algorithm decoding time. The decoders had around a 25% improvement in delay time in comparison to the MAP.

MAP	Array Decoder	One-hot Decoder
1.00 ± 0.02	0.74 ± 0.03	0.76 ± 0.02
Array Autoencoder		One-hot Autoencoder
1.33 ± 0.05		3.02 ± 0.06

Conclusions

- The feasibility of a machine learning based channel decoder were demonstrated for a communication system with a binary symmetric channel.
- The designed decoders improved around 25% the channel delay when compared to the maximum a posteriori rule for a $(16, 8)$ code.
- Their small number of parameters aligned with a GPU equipped decoding computer is promissory to be a substitute of the MAP rule and achieve same bit error correction rate in a real application.
- We also proved that autoencoders can learn a non linear encoding function and its decoding function for a BSC.
- There are still some challenges in training to solve.

Future Work

- A more rigorous hyper-parametric analysis could find a combination of parameters which yields better BER performance for the autoencoders.
- Using more advanced NN topology such as a recurrent neural network (RNN) or a generative adversarial network (GAN).
- Real experimental implementation of the DNN decoders could be tested to confirm their capabilities.

Acknowledgment

I would like to thank ...

- The guidance and advice from my supervisor Mme. Meryem Benammar, Ph.D;
- My colleague Rémy Zawislak who worked in the same theme and collaborated indirectly to the research;
- Mme. Marjorie Grzeskowiak Lucas, Ph.D, who through the Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO) enabled the confection of the project; and
- The evaluators, the LACS member and the audience.

Bibliography I

C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, pp. 3–55, Jan. 2001.

F. D. Calabrese, L. Wang, E. Ghadimi, G. Peters, and P. Soldati, "Learning radio resource management in 5g networks: Framework, opportunities and challenges," *CoRR*, vol. abs/1611.10253, 2016.

T. J. O'Shea and J. Hoydis, "An introduction to machine learning communications systems," *CoRR*, vol. abs/1702.00832, 2017.

T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to Communicate: Channel Auto-encoders, Domain Specific Regularizers, and Attention," *arXiv e-prints*, Aug. 2016.

Bibliography II

D. Goldin and D. Burshtein, "Performance Bounds of Concatenated Polar Coding Schemes," *arXiv e-prints*, Oct. 2017.

E. Worm, S. Member, P. Hoeher, S. Member, and N. Wehn, "Turbo-decoding without snr estimation," *IEEE Communications Letters*, pp. 193–195, 2000.

A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, April 1967.

P. Robertson, P. A. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding.," *European Transactions on Telecommunications*, vol. 8, no. 2, pp. 119–125, 1997.

Bibliography III

H. J., R. P., and P. L., “iterative turbo decoding of systematic convolutional codes with the map and sova algorithms,” pp. 21 – 29, 10 1994.

M. Jordan and R. Nichols, “The effects of channel characteristics on turbo code performance,” pp. 17 – 21 vol.1, 11 1996.

M. Ibnkahla, “Applications of neural networks to digital communications-survey,” *Signal Processing*, vol. 80, pp. 1185–1215, 07 2000.

M. A. Nielsen, “Neural networks and deep learning,” 2018.

K. P. Murphy, *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013.

Bibliography IV

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016.

F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
PMID: 16764513.

Bibliography V

- M. Benammar and P. Piantanida, "Optimal training channel statistics for neural-based decoders," in *52nd Asilomar Conference on Signals, Systems, and Computers, ACSSC 2018, Pacific Grove, CA, USA, October 28-31, 2018* (M. B. Matthews, ed.), pp. 2157–2161, IEEE, 2018.
- N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *CoRR*, vol. abs/1609.04836, 2016.
- S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.