Autoencoders Based Digital Communication Systems

Anjana P K (Student)[1]
Department of ECE College of Engineering Trivandrum
Trivandrum, India

Ameenudeen P E (Assistant Professor)[2]
Department of ECE College of Engineering Trivandrum
Trivandrum, India

Abstract:- To demonstrate over the air transmission, it is essential to frame, alternate and execute a transmission systems repressed by neural networks. Autoencoders are used to train the entire system composed of transmitters and receivers. Estab- lishing a vital novel style of thinking regarding communications network design as a point to point regeneration task that seeks to optimise Tx and Rx systems into a single process by interpreting a transmission system as an autoencoder is performed. In this, several autoencoders such as deep encoder, convolutional autoencoder and a simplest possible autoencoder is simulated in Python. Lastly, BLER versus Eb/N0 for the (2,2) and (7,4) autoencoder is plotted.

Keywords:- *Autoencoder, Deep Learning, End-to-End Communication.*

I. INTRODUCTION

The basic issue of communication involves ”replicating at one end either exactly or almost a signal selected at some other end”, or, reliably conveying a message from a source to a recipient over a medium using a Tx and a Rx [1]. To obtain a theoretically ideal clarification to the given problem in practise, Tx and Rx are often separated into many computa- tional units, each of which is dedicated for a certain sub-task, like encoding, channel coding, modulating, and equalisation. Despite the fact that such an architecture is considered to be suboptimal. It offers the benefit of allowing each element to be separately studied and tuned, resulting in today’s highly effective and reliable systems. DL routing algorithms, on the other hand, return to the basic formulation of the breakdown in communication and strive to optimise transmitter and receiver together without the need for any arbitrarily added block structure. Even though today’s schemes have now been deeply optimised over the past centuries, and it appears challenging to compare with them in terms of efficiency, we are drawn to the conceptual simplification of a transmission network which is to be trained to broadcast over any kind of medium with no prior arithmetical modelling and analysis.

A DNN that has been programmed to recreate the source at the output is referred to as an autoencoder. Because data should transit through each level, the system should discover a strong depiction of the input signal at each level. An auto- encoder is a form of ANN that uses machine algorithms to develop optimum encoding of untrained input. By trying to recreate the data by encryption, the code is checked and enhanced. By instructing the network to disregard inconse- quential input (”noise/interference”), the autoencoder creates a pattern for a set of information generally for feature extraction. In this paper, section II describes the autoencoder concept, section III gives an explanation of how the autoencoder is simulated, section IV presents the result and finally the conclusion in section V.

II. AUTOENCODER CONCEPT

A channel autoencoder is depicted in Fig. 1. A one-hot vector represents the input symbol. Tx includes multiple thick layers of a FNN. For every encoded input symbol, last thick layer has modified to obtain two values of output that depict complex numbers containing real part and an imaginary part. The physical terms on x is decided by normalization layer. An AWGN with an immobile variance depicts the channel. Here Eb/N0 represents energy (Eb) to noise power spectral density (N0) ratio. The Rx was used as FNN. Softmax activation is used in the last layer. Every training instance has a different noise value. In the forward pass, a noise layer is used to alter the given signal. It is rejected by the rearward pass.

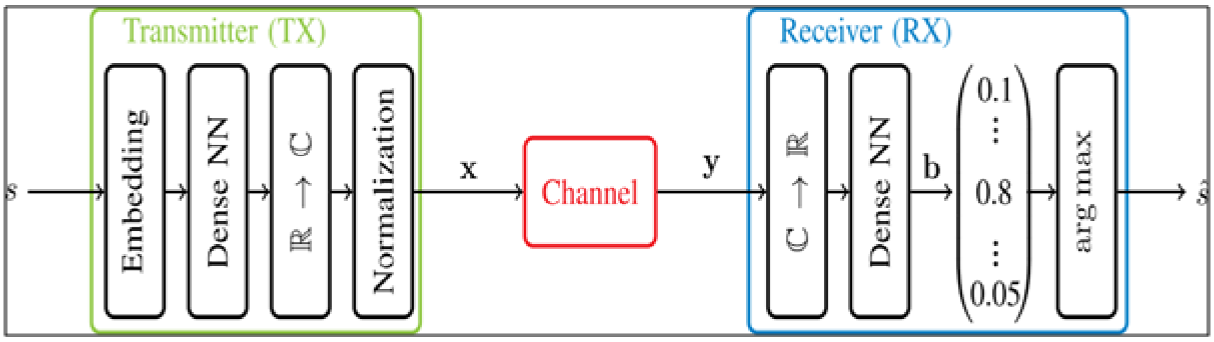


Fig 1 Autoencoder Representation

DL (also referred to as structured learning) is a ML ap- proach that relies on ANNs and reinforcement learning. Learn- ing can be supervised, semi supervised, or not supervised.

➢ *Deep Learning*

ML (particularly DL) serves strong self-operating meth- ods in transmission setups for understanding the information available in the spectra and adjust to movements within the spectra in order to achieve the duty stated in this ideology. Wireless communication is a combination of a variety of waveforms, channels, congestions, and interference effects, each with an individual complicated structure that changes rapidly. Because of the distributed character of air transmission channels, data within the wireless communications obtain higher magnitude as well as greater pace and faces through intervention along with many other safety hazards. Conven- tional designs and ML tactics frequently fail to capture the feeble relation among greatly distributed information spectra and transmission models, whereas DL develops a possible method of satisfying wireless communication system data rate, speed, dependency, and security needs. One motivating example comes from classifying signals, where a receiver must distinguish signals received based on features of the waveform, such as modulation done at the transmitter that adds the carrying signal's data via variation of its parameters.

➢ *End To End Communication*

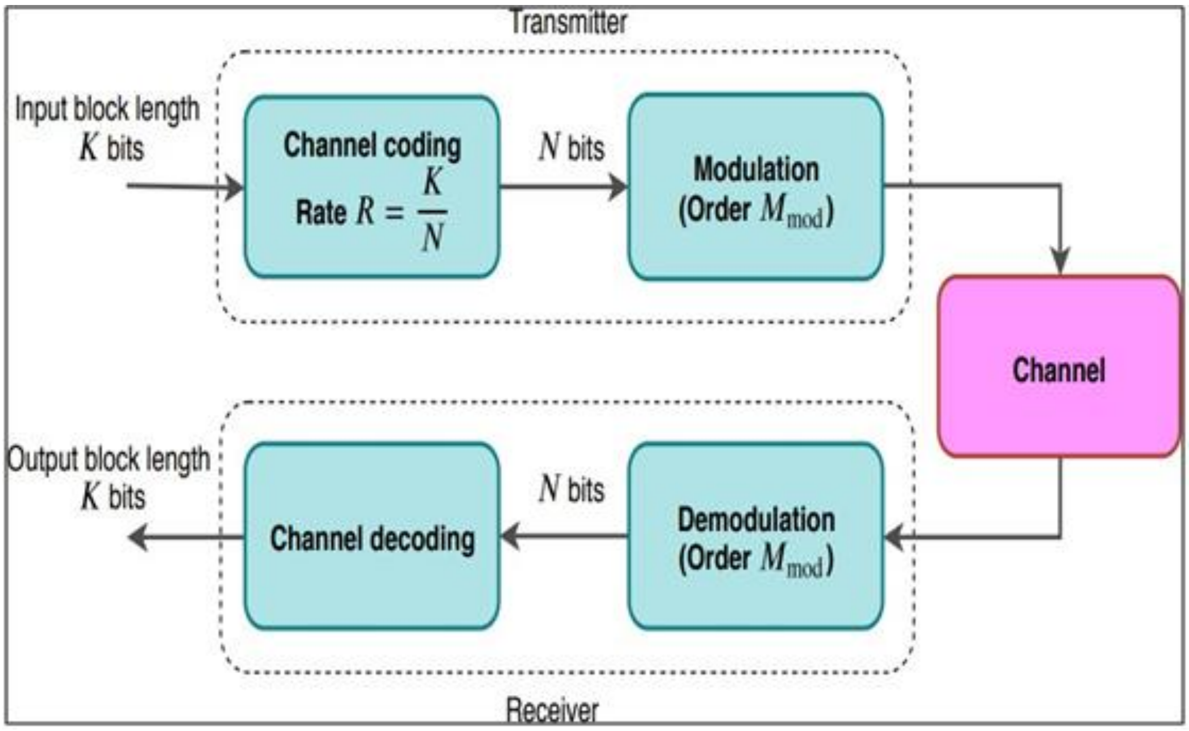
Enhanced DNNs are used for transmitting and receiving in a start-to-finish setup for communication. However, the un- known CSI prevents the start-to-finish setup without knowing by blocking the backward propagation of errors, which are used for tutoring the weights of DNNs.

III. END TO END LEARNING OF CODED SYSTEMS

We evaluate the effectiveness of an autoencoder to end- to-end communication to that of traditional data transmission using channel coding. A traditional data transmission is com- posed of many pieces for channel encoding or decoding, as well as modulation or demodulation. An autoencoder based approach lacks such explicit blocks, meanwhile attempting to enhance the system from end-to-end while conforming to structure characteristics. We examine the performance of autoencoder models that are analogous to standard channel coded communications systems over the AWGN channel using these system characteristics.

➢ *System Model*

To reduce complexity during processing, the blocks are split into different stages. Coding rate, R = K/N, where N repre- sents the output size after channel encoding and transmitter, information block of size K bits is fed to channel encoder where a block of size N is output after channel encoding and the N. Then the data block is fed to the modulator of order Mmod where the data bits are divided into codewords of size kmod = log2(Mmod), and each codeword is mapped to a point in the signal constellation with given amplitudes for the I,Q signals. At the receiver, the reverse process of the above happens where incoming symbols are mapped to codewords and the codewords are grouped serially to produce the block. It is then fed to the channel decoder where the N bit sized block is converted to K bit sized block after channel decoding which is the estimation of the transmitted information block.


Transmitter
Input block length K bits
Channel coding Rate $R = \frac{K}{N}$
N bits
Modulation (Order M_{mod})
Channel
Output block length K bits
Channel decoding
N bits
Demodulation (Order M_{mod})
Receiver
Fig 2 A Standard Communications System Setup made up of Encoding and Decoding Blocks

IV. RESULTS

A simplest possible autoencoder is shown in Fig. 3. The autoencoder is trained for 50 epochs. Autoencoder may reach upto a value of validation loss of around 0.09. The first row is the input and second row is the output. When looking to the output, a little bit of input is loss with this basic approach.

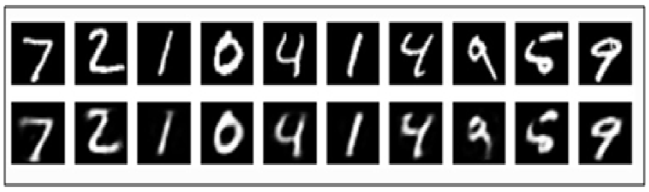


Fig 3 Simple Autoencoder

A deep autoencoder is shown in Fig. 4. The autoencoder is trained for 100 epochs. A loss of test near to 0.10 and train loss of nearly 0.11 is obtained. Autoencoder may reach upto a value of validation loss of around 0.01 (Difference between train loss and test loss). The first row is the input and second row is the output. The output is pretty similar to that of input when using deep autoencoder.

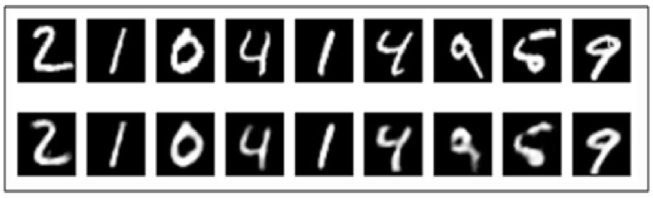


Fig 4 Deep Autoencoder

A convolutional autoencoder is depicted in Fig. 5. It makes sense when using convolutional neural network since the inputs are images. Conv2D and MaxPooling2D layers are there in the encoder, whereas there is Conv2D and UpSampling2D layers in the decoding section. These representations are 8x4x4, so we modify them to 4x32 in such a way that it is able to display them as grayscale images as in Fig. 6.

For image denoising, here, a convolutional autoencoder is used. Compared to the previous convolutional autoen- coder(Fig. 5), to increase the performance of the regenerated output, we will use a small different model with high number of filters per each layer. The autoencoder is trained for 100 epochs. A noisy image is obtained as in Fig. 7 and the denoised output is displayed in Fig. 8. That is, an autoencoder is used to regenerate the input at its output without any prior knowledge. Image denoising is one of the greatest requirements in the image processing field.

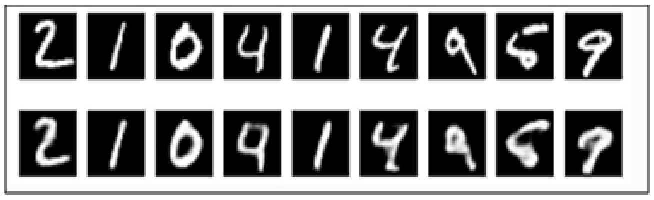


Fig 5 Convolutional Autoencoder

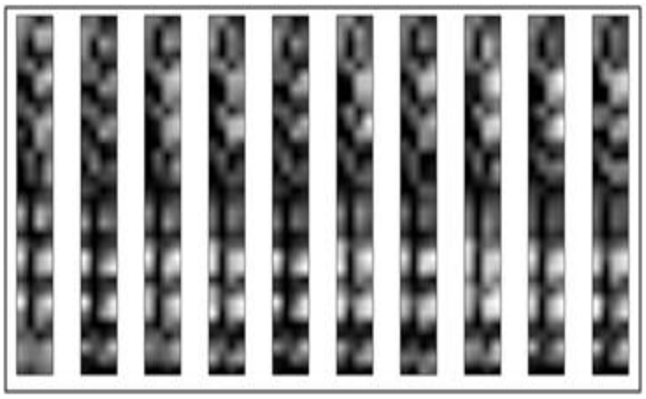


Fig 6 Grey Scale Image

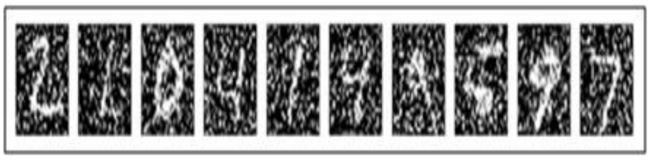


Fig 7 Noisy Image

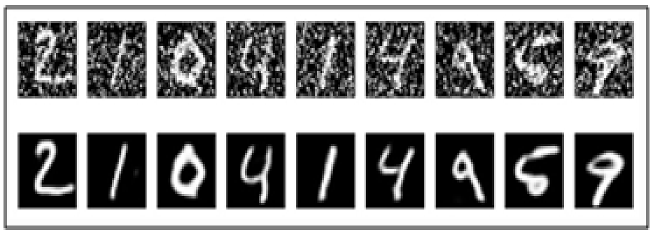


Fig 8 Denoised Output

For an autoencodder in an end-to-end communication sys- tem, it attempts to learn representations x of the messages s that are resilient to channel impairments mapping x to y (e.g., noise, fading, distortion, and so on), such that the delivered message may be retrieved with a low likelihood of a mistake. In other words, while conventional autoencoders remove redundancy from input data in order to compress it, this autoencoder (the ”channel autoencoder”) frequently adds redundancy, learning an intermediate representation that is resistant to channel perturbations [2].

Figure 9 depicts the learned images x of messages when (n, k) equals (2,2) as complex constellation coordinates, where the x and y axes correlate to the initial and next transmitted signal correspondingly. Fig. 10 and fig. 11 depict a similar comparison, but this time for a (2,2) and (7,4) communications network. Interestingly, when the autoencoder obtains relatively similar BLER as unencrypted BPSK for (2,2), it exceeds it for (7,4) over the whole Eb/N0 range. This suggests that it has learned some type of combined coding and modulation strategy that results in coding gain. This solution must be evaluated to a significantly greater modulation technique em- ploying a channel code (or the optimum sphere packed in eight dimensions) for a genuinely fair comparison. A quality mea- surement comparison for multi-channel types and parameters (n, k) with varied baselines is beyond the scope of this work and will have to wait for future research. BER Performance.

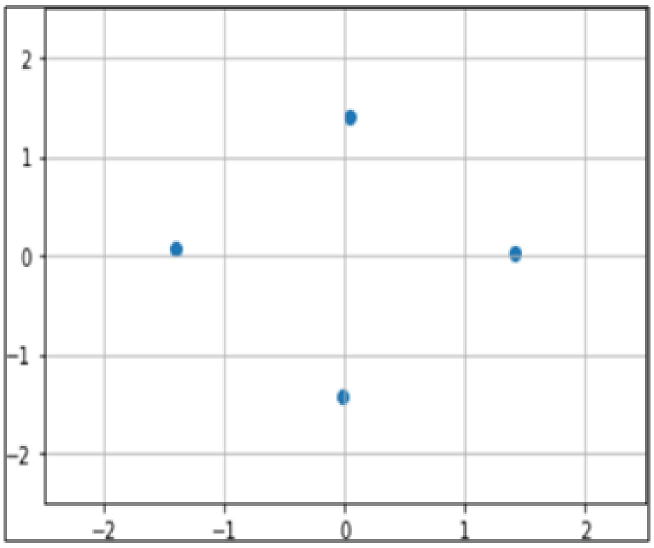


Fig 9 Constellation Produced by (2,2) Autoencoder

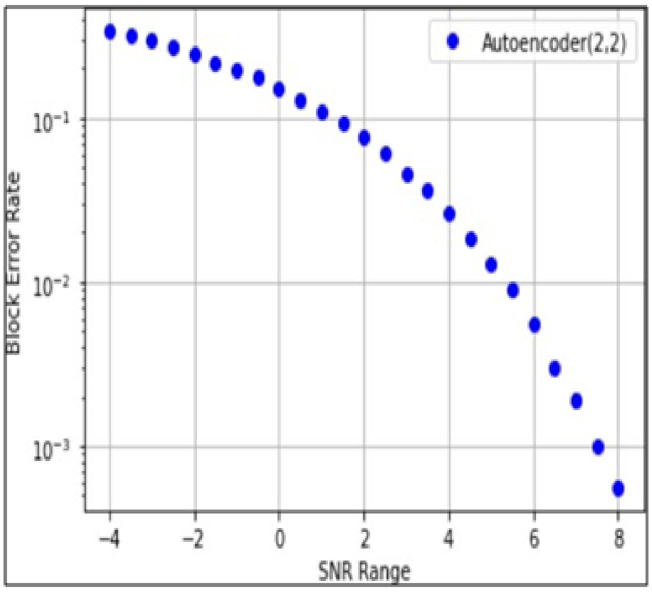


Fig 10 BLER vs Eb/No for (2,2) Autoencoder

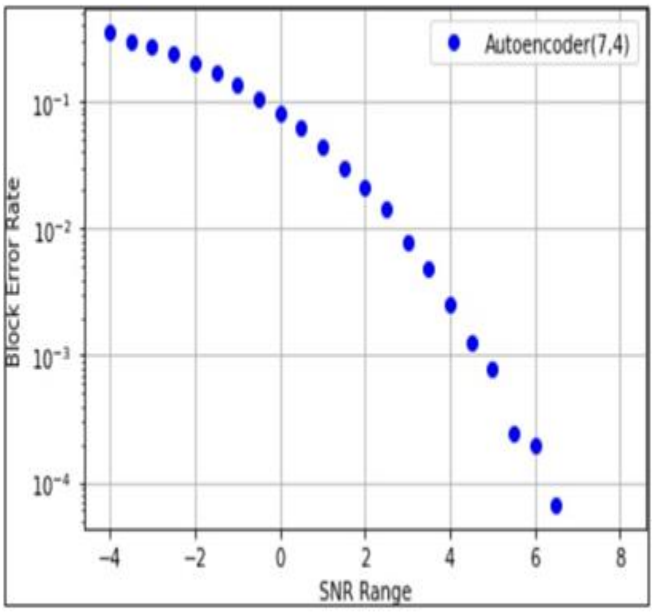


Fig 11 BLER vs Eb/No for (7,4) autoencoder

Figures 12-14 show simulated BER performances of different autoencoder models with R = 1/2, 1/3, 1/4 and their baseline systems with convolutional coding with respective code rates and BPSK modulation scheme. The selected block length for baseline system is K = 800 and the constraint length of the convolutional encoder/decoder is taken as 7. It can be observed that the BER performance of the autoencoder improves when message size is increasing. For a given code rate, M = 2 model has almost the same performance with uncoded BPSK while M = 256 model has resulted in a much improved BER performance closer to the baseline. This improvement is achieved since the model has more degrees of freedom and more flexibility for a better end-to-end optimization when the message size is high.

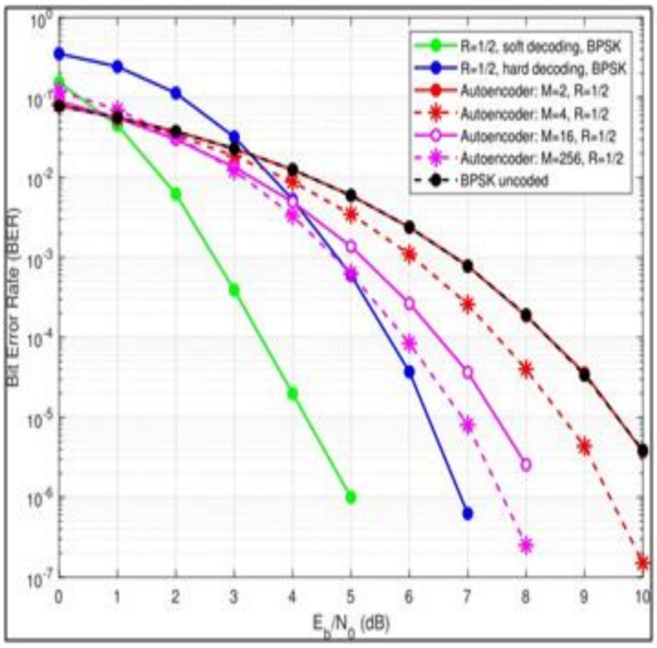


Fig 12 R=1/2 Systems

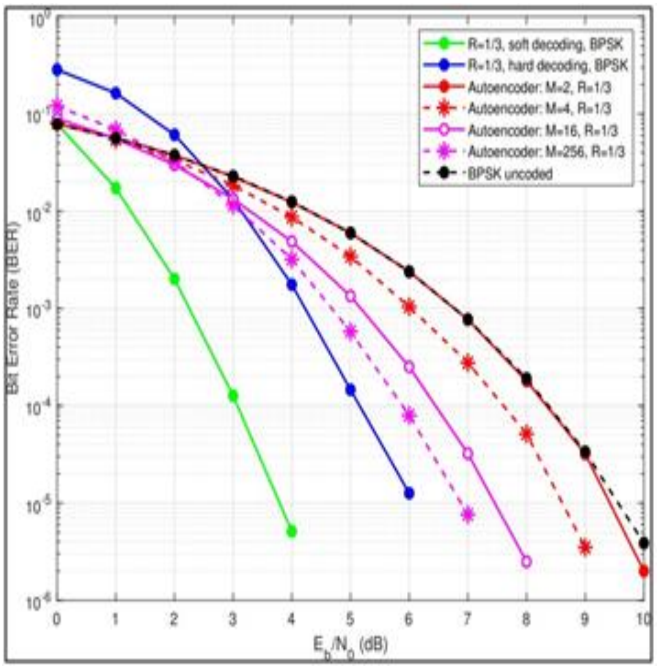


Fig 13 R=1/3 System

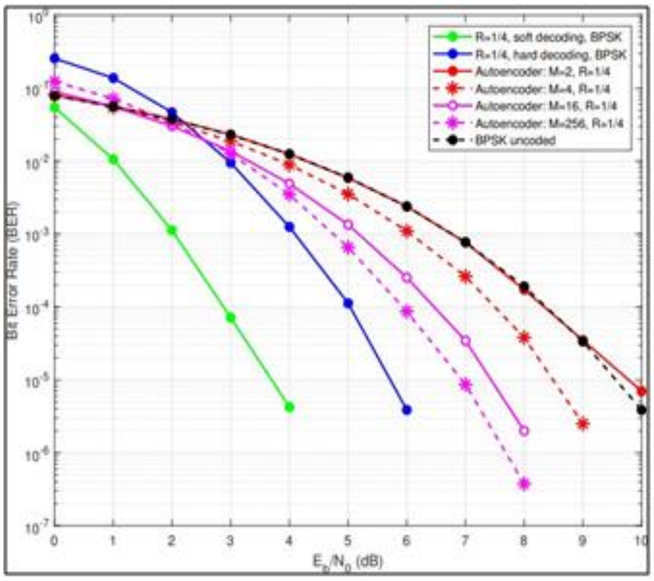


Fig 14 R=1/4 System

V. CONCLUSION

Autoencoders are widely used in communication as well as for image denoising. The concept of autoencoders helps to simplify the entire communication system. By the invention of autoencoders, programs can be made easily without any prior knowledge. Various autoencoders are built in Python and is compared the same with the efficiency while using hamming codes. Various applications to DL in the physical layer is seen in this project. BER comparison of different autoencoders of R=1/2, 1/3, 1/4 with M=2, 4, 16, 256 is observed. A key question is how we can run a structure for an arbitrary medium such that the channel model is unknown. Furthermore, it is critical to enable on-the-fly finetuning in order to adjust the system to shifting channel circumstances for which it was not previously trained. Another critical challenge is determining how to increase the block size in order to improve the autoencoder's efficiency. Overcoming the limitation, we believe some changes can be implemented in the coming phase.

REFERENCES

[1]. S. Do¨rner, S. Cammerer, J. Hoydis and S. t. Brink, ”Deep Learning Based Communication Over the Air,” in IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 132-143, Feb. 2018, doi: 10.1109/JSTSP.2017.2784180.

[2]. O'shea, T., Hoydis, J. (2017). An introduction to deep learning for the physical layer. IEEE Transactions on Cognitive Communications and Networking, 3(4), 563-575.

[3]. F. A. Aoudia and J. Hoydis, "End-to-end learning of communications systems without a channel model," in 2018 52nd Asilomar Conference on Signals, Systems, and Computers, pp. 298–303, IEEE, 2018.

[4]. H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning based semantic communications: An initial investigation," in GLOBECOM 2020-2020 IEEE Global Communications Conference, pp. 1–6, IEEE, 2020.

[5]. S. Do¨rner, 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, 2018.

[6]. D. Silver et al., "Mastering the game of go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[7]. M. Zorzi, A. Zanella, A. Testolin, M. D. F. De Grazia, and M. Zorzi, "Cognition-based networks: A new perspective on network optimization using learning and distributed intelligence," IEEE Access, vol. 3, pp. 1512– 1530, 2015.

[8]. M. Abadi et al., "TensorFlow: Large-scale machine learning on het- erogeneous distributed systems," arXiv:1603.04467, 2016. [Online]. Available: http://tensorflow.org/

[9]. Y.-S. Jeon, S.-N. Hong, and N. Lee, "Blind detection for MIMO sys- tems with low-resolution ADCs using supervised learning," IEEE Int. Conf. Commun., May 2016, pp. 1–6, doi: 10.1109/ICC.2017.7997434.

[10]. M. Abadi and D. G. Andersen, "Learning to protect communications with adversarial neural cryptography," arXiv:1610.06918, 2016.

[11]. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cam- bridge, MA, USA: MIT Press, 2016.

[12]. T. J. O'Shea, J. Corgan, and T. C. Clancy, "Unsupervised representation learning of structured radio communication signals," in Proc. IEEE Int. Workshop Sensing, Processing and Learning for Intelligent Machines (SPLINE), 2016, pp. 1–5.

[13]. R. Al-Rfou, G. Alain, A. Almahairi et al., "Theano: A Python framework for fast computation of mathematical expressions," arXiv preprint arXiv:1605.02688, 2016.

[14]. D. George and E. Huerta, "Deep neural networks to enable real-time multimessenger astrophysics," arXiv preprint arXiv:1701.00008, 2016.

[15]. D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based hy- perparameter optimization through reversible learning," in Proc. 32nd Int.Conf. Mach. Learn. (ICML), 2015.

[16]. J. Qadir, K.-L. A. Yau, M. A. Imran, Q. Ni, and A. V. Vasilakos, "IEEE Access Special Section Editorial: Artificial Intelligence Enabled Networking," IEEE Access, vol. 3, pp. 3079–3082, 2015.

[17]. Portilla, Javier, et al. ”Image denoising using scale mixtures of Gaus- sians in the wavelet domain.” IEEE Transactions on Image processing 12.11 (2003): 1338-1351.

[18]. Vincent, Pascal, et al. ”Extracting and composing robust features with denoising autoencoders.” Proceedings of the 25th international conference on Machine learning. ACM, 2008.

[19]. Python — Peak Signal-to-Noise Ratio (PSNR). Available: https://www.geeksforgeeks.org/ python-peak-signal-to-noise-ratio- psnr/