**Efficient data transfer using Auto-Encoder on FPGA**

DISSERTATION

By

ALLA JAYADHAR

2019HC04284

Dissertation work carried out at

MathWorks India Private Limited, Hyderabad



BIRLA INSTITUTE OF TECHNOLOGY &

SCIENCE

PILANI (RAJASTHAN)

March 2022

**Efficient data transfer using Auto-Encoder on FPGA**

DISSERTATION

Submitted in partial fulfillment of the requirements of the MTech Data Science and Engineering Degree programme

By

ALLA JAYADHAR

2019HC04284

Under the supervision of

Mr. PARASHURAMA HOSAMANI

Manager @ MathWorks Pvt Ltd., Hyderabad



BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani (Rajasthan) INDIA

March 2022

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI CERTIFICATE**

This is to certify that the Dissertation entitled “Efficient Data Transfer Using Auto-Encoder on FPGA” and submitted by Mr./Ms. ALLA JAYADHAR ID No. 2019HC04284 in partial fulfillment of the requirements of DSECLZG628T Dissertation, embodies the work done by him/her under my supervision.



Signature of the Supervisor

Name: Mr. Parashurama Hosamani

Designation: Manager,

The MathWorks PVT. LTD., Hyderabad.

Place: Hyderabad

Date: 27/02/2022

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI SECOND SEMESTER 2020-21**

DSECLZG628T **DISSERTATION**

Dissertation Title : **Efficient Data Transfer Using Auto-Encoder on FPGA**

Name of Supervisor : Mr. Parashurama Hosamani

Name of Student : Alla Jayadhar

ID No. of Student : 2019HC04284

# Abstract

Key Words: Deep Learning, Graphical Processing Units, Field-programmable gate arrays, Software Defined Radio (SDR), Central Processing Unit, Auto-Encoder, Decoder

The growing trend in use of Deep Learning applications has been a boon for Graphic card industry. Though Graphical Processing Units (GPU) are outperforming, there are few imperfections with I/O operations. FPGAs stood as a perfect replacement for GPUs by providing more throughput and high performance than traditional GPUs.

Currently, Field-programmable gate arrays (FPGA) are widely used in Software Defined Radio (SDR) in Communication Industry. And with the help of device drivers we can tune, send/ receive data from SDR without much effort, and with the help of design tools in-hand, modelling FPGAs has been made a lot easier.

Though FPGAs can process lot of data at extremely high speeds, the requirement for sending/receiving the data onto the host device was the tricky one. The processing compatibility between Central Processing Unit (CPU) and FPGA, has always been a bottleneck. Because of this processing speed difference, lot of packets being dropped during transfer of data between host system and the SDRs.

Auto-Encoders and decoders have a perfect fit for this issue, by using the encoders on FPGAs we can compress the signal data and remove the noise from the signal and using decoder on host system we can then decompress and recreate the original data. Thus, removing noise and compressing the data reduces the amount of the data to be sent over the wire network. We can further optimization by detect the signal based on requirement and there by removing the signal which is not required. There by, we are decreasing the amount of data received to the system.

A picture containing text, opener

Description automatically generated

(Signature of Student) (Signature of Supervisor)

Name: Alla Jayadhar Name: Parashurama Hosamani

Table of Contents

[Abstract 4](#_Toc96957194)

[LIST OF FIGURES 8](#_Toc96957195)

[Acknowledgments 9](#_Toc96957196)

[TOOLS REQUIRED 10](#_Toc96957197)

[CHAPTER 1 11](#_Toc96957198)

[1.1 INTRODUCTION 11](#_Toc96957199)

[1.1.1 Background 11](#_Toc96957200)

[1.1.2 Problem Statement 13](#_Toc96957201)

[1.1.3 Objective of the project 14](#_Toc96957202)

[1.1.4 Scope: 14](#_Toc96957203)

[1.1.5 Uniqueness of the Project 15](#_Toc96957204)

[1.1.6 Benefit to the organization 15](#_Toc96957205)

[1.2 Model Architecture 16](#_Toc96957206)

[1.2.1 Model Workflow: 17](#_Toc96957207)

[The below diagram illustrates the workflow of the current implementation 17](#_Toc96957208)

[CHAPTER 2 19](#_Toc96957209)

[2.1 NOISE REDUCTION 19](#_Toc96957210)

[2.1.1 Data preprocessing 20](#_Toc96957211)

[2.1.2 Model Training 20](#_Toc96957212)

[2.1.3 Results 20](#_Toc96957213)

[CHAPTER 3 22](#_Toc96957214)

[3.1 DATA COMPRESSION 22](#_Toc96957215)

[3.1.1 Data Preprocessing 22](#_Toc96957216)

[3.1.2 Model Training: 22](#_Toc96957217)

[CHAPTER 4 24](#_Toc96957218)

[4.1 Signal Detection 24](#_Toc96957219)

[4.1.1 Data Preprocessing 25](#_Toc96957220)

[4.1.2 Model Training 26](#_Toc96957221)

[CHAPTER 5 27](#_Toc96957222)

[5.1 DEEP LEARNING ON FPGA 27](#_Toc96957223)

[CHAPTER 6 28](#_Toc96957224)

[6.1 Results after Integration 28](#_Toc96957225)

[FUTURE SCOPE AND CONCLUSION 30](#_Toc96957226)

[Bibliography 31](#_Toc96957227)

[APPENDIX 33](#_Toc96957228)

[Duly Completed Checklist 34](#_Toc96957229)

# LIST OF FIGURES

[Figure 1 : Silicon chips comparison 12](#_Toc96973511)

[Figure 2 : Comparing core types for edge AI Hardware 13](#_Toc96973512)

[Figure 3 : Model Architecture 16](#_Toc96973513)

[Figure 4 : Model Embedded on FPGA 17](#_Toc96973514)

[Figure 5 : Model workflow 18](#_Toc96973515)

[Figure 6 : Auto-Encoder and decoder network 19](#_Toc96973516)

[Figure 7 : Training results of the I signal 20](#_Toc96973517)

[Figure 8 : Training results for the Q signal 21](#_Toc96973518)

[Figure 9 : Layers in Auto-Encoder 21](#_Toc96973519)

[Figure 10 : Auto Encoders 22](#_Toc96973520)

[Figure 11 : Encoder and decoder for compressing the data 23](#_Toc96973521)

[Figure 12 : packet format 24](#_Toc96973522)

[Figure 13 : Deep learning layers with neurons 25](#_Toc96973523)

[Figure 14 : Model Training Result 26](#_Toc96973524)

[Figure 15 : Port Deep learning network on FPGA 27](#_Toc96973525)

[Figure 16 : Memory allocation 28](#_Toc96973526)

[Figure 17 : Decoded signal 29](#_Toc96973527)

[Figure 18 : confusion matrix 29](#_Toc96973528)

# Acknowledgments

I’m very much grateful to the supervisor of this project Mr. Parashurama Hosamani for providing his genuine thoughts and inputs while reviewing my work despite his busy schedule and work pressure. I thank my BITS faculty guide Mrs.Uma Maheswari Natarajan for his timely grading on assigned slots and bearing with my delay in response for setting the time slot due to my health condition at the beginning of Dissertation. I would also thank BITS-Pilani institute for giving me this opportunity to do my masters along with my work. I would not have completed this work without the support of my family who took care of major household work for 2 years. Finally, I thank God who has blessed me with great people around me who constantly motivate me to go forward in life.

# TOOLS REQUIRED

1. **Software Defined Radio**
2. **MATLAB**
   1. **MATLAB**
   2. **Deep learning Toolbox**
   3. **HDL coder**
   4. **Embedded Coder**
3. **Xilinx Vivado 2019.2 and Above**
4. **Python**
   1. **Editor: Visual Studio Code/ PyCharm/Jupyter Notebook**
   2. **Version: 3.8.1**
   3. **Packages Used:**
      1. **TensorFlow- Keras /pyTorch**
      2. **Matplotlib**
      3. **Pandas**
      4. **NumPy**

# CHAPTER 1

## 1.1 INTRODUCTION

### 1.1.1 Background

Though, the definitions and theory of Artificial intelligence was there from 1956, the applications have been limited, due to various bottlenecks. Availability of data for training and computation power are among them. With the rapid increase in big data and storage capacities, we can be able to store and process huge datasets very quickly. As stated in Moore’s law, number of transistors on a microchip doubles every two years. And this has direct impact in improving the computation power exponentially for every two years. The increase in usage of Graphical Processor Unit (GPU)’s has fuelled the computational power.

Since late 2000’s, the use of Machine learning applications has been widespread to various fields, and had given exceptional results in speech recognition, natural language processing, computer vision etc., A subset of artificial neural networks has emerged as achieving higher accuracy and performance across a very large set of Machine learning applications, compared to traditional state-of-the-art algorithms. This subset of artificial neural networks includes both Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs).

Due to this there is rapid development of deep learning applications, the significance of deep learning is no longer just the combination of multi-layer neural networks, and even the machine learning, which is broader than the concept of deep learning, is gradually turning to the intelligent direction.

So, we need to design of hardware framework that runs deep learning algorithms with high performance, low power, flexibility, and high throughput. This has opened the new requirements in the field of chip manufacturing.

Till that time CPUs are primarily used to run these artificial networks, but CPUs are designed for serial operations for supporting advanced logic. Though it was available with more cache to fetch complex operations. Still, they are poor at performance. GPUs came as a perfect replacement for them, the availability of more cores and parallel execution of more threads, GPUs provide better performance. Figure 1 will illustrate the flexibility and efficiency of FPGAs.

Graphical user interface, application, Teams

Description automatically generated

Figure 1 : Silicon chips comparison

FPGAs are available in the market by that time, the complexity of designing the gates and hardware has made it bit complex to enter the performance industry.

FPGAs can produce circuits with thousands of memory units for computation, so they work similarly to GPUs. FPGAs have adaptable architecture, enabling additional optimisations for an increase in throughput. Thus, the possible volume of calculations makes FPGAs a viable solution to GPUs. The speed metrics of FPGA’s and GPUs are sited here [1]. ASICs also provide better performance when compared to FPGA, but they are not flexible and restricted to fixed functionality. This table gives good comparison between different infrastructure. [2]

Table

Description automatically generated

Figure 2 : Comparing core types for edge AI Hardware

Comparatively FPGAs have lower power consumption and can be optimal for embedded applications. They are also an accepted standard in safety-critical operations such as ADAS (Advanced Driver Assistance Systems) in automotive. Currently, around FPGA acceleration research have designed a multi-FPGA architecture to accelerate a pre-training algorithm of deep learning.  FPGAs can implement custom data types whereas GPUs are limited by architecture. The advantages of FPGA are cited here [3]

With the exponential growth in the ways and means by which people need to communicate, signal communication has become business critical. Software defined radio (SDR) technology brings the flexibility, cost efficiency and power to drive communications forward, with wide-reaching benefits realized by service providers and product developers through to end users.

### 1.1.2 Problem Statement

SDRs typically used for easy communication and modulation of demodulation of radio signals. Because of limited resources and various other reasons like analysing the received signals, we tend to get the signals from SDR to host device. With the increase in requirement for high-speed signals, the amount of data to be exchanged between SDR to host system has increased significantly. With the data consumption capacity haven’t increased at the same rate, this layer always forms as a bottleneck between host and SDRs. Though there are number of buffers provided using DDR memory both on host and SDR, the gap hasn’t been fulfilled and the loss in data becoming inevitable at high rates. To fill this gap, we are using Auto-Encoders on FPGA to reduce the dimensionality of the signals, so that we can decode the data back on host.

With this technique, we can extend our scope to resolve some other issues, like filling the overflow data with predicted data using Variational Auto-Encoders, removing the noise in the signal and detection of various signals and filter them.

### 1.1.3 Objective of the project

Implement Auto-Encoders on FPGA to remove the noise and dimensionality of the signals in test on real-time signals. Replace the preamble detector technique with CNN layer in signal detection and gather results on accuracy.

1. Make sure not much data was lost during decoding
2. Train the network more rigorously, so that it won’t affect the accuracy between traditional preamble detector and CNN layer.
3. Make sure time for running the model, won’t add more overhead on the SDR device

### 1.1.4 Scope:

1. Improve data transferring capability
2. Ease out the way for detecting the signal
3. Predict and fill the missed out/overflow packets
4. Denoise the signal, so that we can remove unnecessary data for processing and analysing.

1.1.4.1 Limitations**:**

Data accuracy can become a problem, because we don’t want to loose any data while decoding or noise reduction.

### 1.1.5 Uniqueness of the Project

We are dealing with three types of problems which will occur in getting the signals from SDR and trying to find a solution using deep learning application.

1. Denoising the data: signal denoising is a natural restoration strategy consists of thresholding the coefficients of the noisy signal at a level that will remove most of the noise but preserve the few significant coefficients in the signal. We can use traditional thresholding technique to remove the noise. Here we are using Auto-Encoders to remove the noise form the signal.
2. Data transfer between host and SDR: We used to send the data using specified hardware in some cases. Example: Some USRP devices can send data up to 10 Gbps, so we need to have special NIC card for communicating the data. In this project, we are using Auto-Encoders to compress/ reduce the dimensions of data and decompress the data on host.
3. Handling missing data: In most of the SDR based algorithms, we will take precautions to avoid the overflows and if there are any missing data, we need to investigate regarding missing data. Here we can fix that data using by predicting the missing data from the encoder and get the final data from the decoder.
4. Detecting the signal: Traditionally, we need to unwrap the packet and check the bits in the preamble packet, based on that preamble packet we can decide the type of signal. We can ease that work using the CNN’s and for that we are going to train model with the label-encoded dataset.

### 1.1.6 Benefit to the organization

For the purpose of high-speed data transfer, we need to maintain different hardware and for adding the different hardware into the Host system, we need to maintain separate slot. And the number of slots is limited to two in CPU. Because of this we can connect the maximum of two SDR devices. And in case of laptops, we can’t even connect this hardware. This will add lot of flexibility in transferring the data from SDR device to host.

With single variational Auto-Encoders model we can decrease the number of applications, earlier we need separate implementation for Denoising, Handling missing data.

By detecting the signal using CNN’s we can decrease the overhead of checking the preamble every time for detecting the signal.

## 1.2 Model Architecture

Diagram

Description automatically generated

Figure 3 : Model Architecture

Here we have 3 models trained a ported on to FPGA, the first model was for signal denoising, then we can send that signal to another CNN network, here we are going to detect the signal, using signal denoising in previous step will increase the accuracy of the model. Then we are going to send the selected signal for data compression using Auto-Encoder. Here we are using trained Auto-Encoder model and on FPGA we are only using the Encoder part.

**Diagram

Description automatically generated**

Figure 4 : Model Embedded on FPGA

The encoded and compressed signal will be taken onto the host machine over the Ethernet and using trained decoder, we can unwrap to original data.

### 1.2.1 Model Workflow:

We can reduce the efficiency of data transfer from FPGA to host in two ways.

1. Detect the signal and send the selected signals from FPGA to host
2. Reduce the data using Auto-Encoders, here we will place the encoder model on FPGA and decoder model on Host.

### The below diagram illustrates the workflow of the current implementation

Diagram

Description automatically generated

Figure 5 : Model workflow

First, we need to search for the suitable datasets for signal detection, here we need labelled dataset to classify the signals.

# CHAPTER 2

## 2.1 NOISE REDUCTION

Random Noise always exists in the wireless signals. Studies have been conducted to get a clean data from non-stationary noisy signal, which is one of the areas in signal detection and various other fields. Since conventional methods rely on first-order statistics, the effort to eliminate noise using deep learning method is intensive. In the real environment, many types of noises are mixed with the target sound, resulting in difficulty to remove only noises. However, most of previous works modeled a small amount of non-stationary noise, which is hard to be applied in real world. To cope with this problem, we propose a novel deep learning model to enhance the auditory signal with adversarial learning of two types of discriminators. One discriminator learns to distinguish a clean signal from the enhanced one by the generator, and the other is trained to recognize the difference between eliminated noise signal and real noise signal. In other words, the second discriminator learns the waveform of noise. Besides, a novel learning method is proposed to stabilize the unstable adversarial learning process. Compared with the previous works, to verify the performance of the proposed model, we have added noise to the signal and tried to regenerate the original data from that.

Auto-Encoders are neural network architectures that consists of three layers, namely, encoder, decoder networks and connected with latent space.

A screenshot of a computer

Description automatically generated with low confidence

Figure 6 : Auto-Encoder and decoder network

The decoder network will be transpose of the encoder network and the input and output layer have same number of dimensions. There are various types of Auto-Encoders, and we can use based on the requirement. [4]. We are using denoising Auto-Encoders for this project.

### 2.1.1 Data preprocessing

The dataset consists of 2555904 samples and each sample consists of 1024 I/Q signals.

The signal should be continuous, but our current dataset will be of [2555904 1024 2] size. We can use reshape to change the dimensions of the dataset. The targeted dimension of the dataset will be [2555904\*1024 2].

For training we need to have signal without and with noise datasets. So, we need to add noise to the existing dataset.

Equation1 Add gaussian noise to the existing signal

### 2.1.2 Model Training

For denoising, we are using nine hidden layers, among that three bi-directional LSTM layers. And at the output layer, we will be using regression layer as we need output ranging from various non-fixed values. We are dividing the input into 1024 samples per batch and running up to 20 epochs. And we are using Adam optimization algorithm for updating the network weights during training process. The learning rate of the model was 0.001.

### 2.1.3 Results

We have trained for I/Q samples separately, so that we can have accuracy of the individual signal dimension.

The error rate (RSME) for I signal was plotted in the Figure 7.

Graphical user interface, application, table, Excel

Description automatically generated

Figure 7 : Training results of the I signal

The error rate(RSME) for Q signal was plotted in the Figure 8

Graphical user interface, application

Description automatically generated

Figure 8 : Training results for the Q signal

The layers in the current implementation of Auto-Encoder model was added in Figure 9

Diagram

Description automatically generated

Figure 9 : Layers in Auto-Encoder

# CHAPTER 3

## 3.1 DATA COMPRESSION

The compression algorithms for the dimensional reductions play an important role in various fields. PCA is one such algorithm that helps in reducing the dimensions in the data. However, for nonlinear data, the results of the PCA are slightly outperformed by Auto-Encoders.

Figure 10 will give quick insight on Auto-Encoders.

Diagram

Description automatically generated

Figure 10 : Auto Encoders

Here for signal compression, we are interested in bottleneck layer. We need to train the combined Encoder and Decoder and then will run the Encoder on FPGA, the output of the encoder will be sent through Wire to the host and on the host we are going to decode the data.

3.1.1 Data Preprocessing:

The output data from the denoising auto-encoder model will be linked to this layer.

### 3.1.2 Model Training:

For compressing the data at the encoder, we are using six hidden layers, among that two 1D-Convolutional layers. Decoder will have eight layers and will be exact transpose of the encoder. And we are using Adam optimization algorithm for updating the network weights during training process. The learning rate of the model was 0.001. The encoder and decoder model was added in Figure 11 : Encoder and decoder for compressing the data

Diagram

Description automatically generated

Figure 11 : Encoder and decoder for compressing the data

# CHAPTER 4

## 4.1 Signal Detection

The detection of a signal in noise depends on the signal duration, amplitude, and the corresponding noise process. This becomes more difficult if correlated noise, or interfering signals, are also in the same band as the signal you wish to detect. Accurate identification of the signal type in shared spectrum networks is critical for efficient resource allocation and fair coexistence. Most of the present-day wireless systems are packet based and preamble detection is a technique which is widely used for signal detection in packet-based system. Here we will decode the information from the initial packet and decide the signal type. [5]

A picture containing application

Description automatically generated

Figure 12 : packet format

Common spectrum sensing techniques are likelihood or feature-based e.g., cyclo-stationarity, preamble detection. In these techniques, signal detection is performed under certain assumptions of the underlying waveforms, e.g., their modulation and coding scheme, protocol behavior, probability distributions, etc., which strongly depend on the decoded signal. In addition to relying on specific model-based assumptions, conventional sensing approaches often assume that spectrum dynamics are slowly varying. [6]

The structure of a DNN model is shown in Fig. 6. Generally, DNNs are deeper versions of ANNs by increasing the number of hidden layers in order to improve the ability in representation or recognition. Each hidden layer of the network consists of multiple neurons, each of which has an output that is a nonlinear function of a weighted sum of neurons of its preceding layer, as shown in Fig. 6. The output layer consists of nonlinear function which may be the Sigmoid or any other activation function. Hence, the output of the network z is a cascade of nonlinear transformation of input data I , mathematically expressed as

z=f(I,θ)=f(L−1)(f(L−2)(f(1)(I)))Equation 2 Neural network output

where L stands for the number of layers and θ denotes the weights of the neural network. The parameters of the model are the weights for the neurons.

The weights of the hidden layers are usually learned on a training set, with known output

Diagram

Description automatically generated

Figure 13 : Deep learning layers with neurons

### 4.1.1 Data Preprocessing

The dataset consists of 2555904 samples and each sample consists of 1024 I/Q signals. The dataset was embedded with labels of signal type. For this project we will be working on 14 modulation types labelled signal dataset.

The label data available will be categorical data rather than numerical values. But machine learning algorithms cannot work with categorical data. To convert categorical data to numerical data, we have two famous techniques available.

1. Integer Encoding
2. One-hot Encoding

In Integer Encoding, the integer will have direct one-on-one relationship with the label i.e.,

Label will be hardcoded to integer and one integer represents a label.

The data will be divided into three parts, namely training, validation and testing datasets. The training part consists of 80 percent of the randomly chosen samples, whereas validation and testing datasets contains 10 percent of the randomly chosen samples from the whole dataset.

The training data will be used in training the model, validation set will be used in improving the model performance by fine tuning the model by calculating the loss and changing the hyper-parameters after each epoch. The testing set will be used in finding the model accuracy.

### 4.1.2 Model Training

In general, the model will be trained based on the requirements in the use case. For signal prediction, the model needs to be trained to classify the signal type and the required result from the final output layer of the model should give the predicted signal modulation.

The current Deep learning model consists of 28 hidden layers. The input layer will take 1024 samples of I/Q data. ReLU activation function was used after every convolution layer

Equation 3 ReLU activation function

and SoftMax activation function

Equation 4 SoftMax activation function

will be used at the classification layer.

The output will be between [1, 14] and based on the output we can map the encoded label and get the predicted result.

Table

Description automatically generated with medium confidence

Figure 14 : Model Training Result

# CHAPTER 5

## 5.1 DEEP LEARNING ON FPGA

FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects. Compared to other chips, FPGAs provide a combination of programmability and performance. FPGAs make it possible to achieve low latency for real-time inference requests. Batching can cause latency because more data needs to be processed. Implementations of neural processing units don't require batching; therefore, the latency can be many times lower, compared to CPU and GPU processors. And FPGAs can be reconfigured to deploy different Machine learning and deep learning algorithms.

For porting trained deep learning model onto FPGA we are using Deep Learning HDL Toolbox [7]. The deep learning series networks which were trained earlier will be ported onto FPGA and SoC boards. After porting the model, we can get the predicted result to MATLAB from the target FPGA board using MATLAB functions.

Table

Description automatically generated with low confidence

Figure 15 : Port Deep learning network on FPGA

# CHAPTER 6

## 6.1 Results after Integration

As mentioned in the architecture, the compressed data will be gotten into MATLAB, and we will decode the data in MATLAB. The difference in the memory allocated before and after decoding we shown in Figure 16

Chart, bar chart

Description automatically generated

Figure 16 : Memory allocation

After decoding the signal will look like

A picture containing graphical user interface

Description automatically generated

Figure 17 : Decoded signal

This signal will be sent for signal detection model to find the final output and they will find the accuracy on the test data. The result will be one of the modulation types of signals.

The test accuracy was 94%. Figure 18 will represent the confusion matrix with test accuracy.

Chart

Description automatically generated

Figure 18 : confusion matrix

# FUTURE SCOPE AND CONCLUSION

Though we have high accuracy and better data compression results, in this project we have dealt with fixed type of noise and limited types of modulation signals. We have limited this functionality to receiver side for transferring the data. Though we can implement similar approach by keeping encoder on host and decoder on FPGA. But for that we need to create another HDL model and bitstream for running that HDL model on the FPGA for preprocessing the predicted data.

The current implementation manages to port the model on the FPGA, but it didn’t able to add the preprocessing and other helper blocks onto FPGA, so because of that we are tightly bounded with MATLAB for every prediction. As a future work, we need to focus on generating an application-oriented reference design and bitstream that can take a preprocessing and other helper block onto FPGA, so that we can remove the latency time of communicating with MATLAB.

# Bibliography

|  |  |
| --- | --- |
| [1] | A. singh, "Hardware for Deep Learning: Know Your Options," [Online]. Available: https://towardsdatascience.com/deep-learning-hardware-know-your-options-9e95026b5d5e. |
| [2] | Z. Tsai, "Embedded Hardware for Processing AI at the Edge," [Online]. Available: https://blog.adlinktech.com/en/2021/02/19/embedded-hardware-processing-ai-edge-gpu-vpu-fpga-asic/. |
| [3] | F. v. G. f. D. Learning, "Intel," [Online]. Available: https://www.intel.la/content/www/xl/es/artificial-intelligence/programmable/fpga-gpu.html#:~:text=FPGA%20vs.%20GPU%20for%20Deep%20Learning%20FPGAs%20are,circuitry%20that%20can%20be%20reprogrammed%20for%20different%20functionalities.. |
| [4] | R. Khandelwal, "Deep Learning — Different Types of Autoencoders," 2018. [Online]. Available: https://medium.datadriveninvestor.com/deep-learning-different-types-of-autoencoders-41d4fa5f7570. |
| [5] | S. K. a. C. S. Sumeeth Nagaraj, "On Preamble Detection in Packet-Based Wireless," 2006. [Online]. Available: http://www.ece.ualberta.ca/~hcdc/Library/NagKhaSchBur06.pdf. |
| [6] | M. F. M. K. a. A. H. Y. A. Wenhan Zhang, "Signal Detection and Classification in Shared Spectrum," [Online]. Available: https://wireless.ece.arizona.edu/sites/default/files/Infocom\_2021\_wenhan.pdf. |
| [7] | "Supported Networks, Layers, Boards, and Tools," [Online]. Available: https://in.mathworks.com/help/deep-learning-hdl/ug/supported-networks-layers-boards-and-tools.html. |
| [8] | L. Reese, "COMPARING HARDWARE FOR ARTIFICIAL INTELLIGENCE," [Online]. Available: http://lreese.dotsenkoweb.com/2019/03/30/comparing-hardware-for-artificial-intelligence-fpgas-vs-gpus-vs-asics/. |
| [9] | G. Y. L. B.-H. J. Hao Ye, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," *IEEE Wireless Communications Letters,* vol. 7, no. 1, 2018. |
| [10] | J. T. narengerile, "Deep Learning for Signal Detection in Non-Orthogonal Multiple Access Wireless Systems," *2019 UK/ China Emerging Technologies (UCET),* 2019. |
| [11] | N. A. Amran, M. D. Soltani, M. Yaghoobi and M. Safari, "Deep Learning Based Signal Detection for OFDM VLC Systems," *2020 IEEE International Conference on Communications Workshops (ICC Workshops),* vol. 1, 2020. |
| [12] | G. Lacey, G. Taylor and S. Areibi, "Deep Learning on FPGAs: Past, Present, and Future," 2016. [Online]. Available: https://arxiv.org/pdf/1602.04283v1.pdf. |
| [13] | A. Sento, "Image compression with auto-encoder algorithm using deep neural network (DNN)," *2016 Management and Innovation Technology International Conference (MITicon),* vol. 1, no. 1, 2016. |
| [14] | A. Moore and R. Wilson, FPGAs for Dummies, Intel. |
| [15] | "Prototype Deep Learning Networks on FPGA," [Online]. Available: https://in.mathworks.com/help/deep-learning-hdl/deployment.html. |

# APPENDIX

1. out = awgn(in,snr) Equation 1 Add gaussian noise to the existing signal
2. z=f(I,θ)=f(L−1)(f(L−2)(f(1)(I))) Equation 2 Neural network output
3. Equation 3 ReLU activation function
4. Equation 4 SoftMax activation function

# Duly Completed Checklist

|  |  |  |
| --- | --- | --- |
| Item Number | Item | Yes/No |
| 1 | Is the Cover page in proper format? | Yes |
| 2 | Is the Title page in proper format? | Yes |
| 3 | Is the Certificate from the Supervisor in proper format? Has it been signed? | Yes |
| 4 | Is Abstract included in the Report? Is it properly written? | Yes |
| 5 | Does the Table of Contents page include chapter page numbers? | Yes |
| 6 | Does the Report contain a summary of the literature survey? | Yes |
| 7 | Are the Pages numbered properly? | Yes |
| 8 | Are the Figures numbered properly? | Yes |
| 9 | Are the Tables numbered properly? | Yes |
| 10 | Are the Captions for the Figures and Tables proper? | Yes |
| 11 | Are the Appendices numbered? | Yes |
| 12 | Does the Report have Conclusion / Recommendations of the work? | Yes |
| 13 | Are References/Bibliography given in the Report? | Yes |
| 14 | Have the References been cited in the Report? | Yes |
| 15 | Is the citation of References / Bibliography in proper format? | Yes |