15EC496L-MAJOR PROJECT

On

# HARDWARE OPTIMIZATION OF DEEP LEARNING SYSTEMS USING STOCHASTIC COMPUTING

*Submitted for partial fulfilment for the award of the degree*

# BACHELOR OF TECHNOLOGY

In

## ELECTRONICS AND COMMUNICATION ENGINEERING

*Submitted By*

## V.UMESH KRISHNA RA1611004040011

**P.YESHWAND V.VIGNESHWARAN**

Under the guidance of

## GUIDE NAME Mr. A. DINESH BABU

**RA1611004040016 RA1611004040036**

(Assistant Professor, Department of Electronics and Communication Engineering)



## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

**FACULTY OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE & TECHNOLOGYY VADAPALANI CAMPUS, CHENNAI-600 026**

**MAY 2020**

**BONAFIDE CERTIFICATE**

Certified that this project report titled “**HARDWARE OPTIMIZATION OF DEEP LEARNING SYSTEMS USING STOCHASTIC COMPUTING**” is the bonafide work of **V UMESH KRISHNA** - **RA1611004040011, P YESHWAND - RA1611004040016 and V**

**VIGNESHWARAN** - **RA1611004040036** who carried out the project work under my supervision as a batch. Certified further, that to the best of my knowledge the work reported herein is not from any other project on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

In-charge Head of the Department

Mr. A. Dinesh Babu Dr. C. Gomathy,

Assistant Professor, Professor and Head,

Department of ECE, Department of ECE,

SRM IST, Vadapalani SRM IST, Vadapalani

Date:

*Submitted for University Examination held in MAY 2020 at SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Vadapalani, Chennai-600 026.*

Date: Internal Examiner External Examiner

# ACKNOWLEDGEMENT

We are indebted to many people who helped us to complete our project and assisted us since the inception of the idea till the day we documented it and even after that. First and foremost, our deep and sincere gratitude to our family members for their continuous and unparalleled love, help and support.

Secondly, we would like to express our gratitude to the MANAGEMENT OF SRM INSTITUTE OF SCIENCE AND TECHNOLGY, VADAPALANI without whose support our project could not have been completed successfully.

We would like to thank **Dr. C. Muthamizhchelvan**, Director and **Dr. K. Duraivelu,** Dean (E&T), Faculty of Engineering and Technology for providing us the required assistance. We are very much grateful to **Dr. C. Gomathy,** Professor and Head, Department of Electronics and Communication Engineering for providing us with continuous encouragement that led us to where we are right now, satisfied, having gotten what we had worked for.

We take the extreme privilege in extending our hearty thanks and sincere gratitude to our Class In-Charge **Mr. P. Rathinakumar**, and Class Counsellor **Mr. P. Glaret Subin**, for their valuable suggestions, support and encouragement towards the completion of the project with perfection.

We would also extend our whole hearted thanks to **Dr. A. Shirly Edward** and **Dr.J.Anita Chirstaline** for periodically reviewing our project. We are very much thankful to our guide **Mr. A. Dinesh Babu**, for his guidance and valuable suggestions in completing this project a successful one amidst the COVID 19 lockdown.

We would like to thank all our friends, who gave us the necessary distractions from our project. Finally yet importantly, we would like to thank each other for the mutual concern and interest towards the completion of the project.

**TABLEOFCONTENTS**

|  |  |
| --- | --- |
| **ACKNOWLEDGEMENT** | **3** |
| **LIST OF FIGURES** | **8** |
| **LIST OF TABLES** | **10** |
| **LIST OF ABBREVIATIONS** | **11** |
| **ABSTRACT** | **13** |

## INDEX

|  |  |  |
| --- | --- | --- |
| **CH NO** | **CHAPTER NAME** | **PAGE**  **NO** |
| **1** | **INTRODUCTION**   * 1. Introduction   2. Working of Deep Learning      1. Artificial Intelligence      2. Machine Learning         1. Types of Machine Learning            1. Supervised Learning            2. Unsupervised Learning            3. Reinforcement Learning      3. Deep Learning      4. Neural Networks | 14  14  14  15  15  15  15  16  16  17  17 |

|  |  |  |
| --- | --- | --- |
|  | 1.2.4.1 Image Processing with Neural Network   * 1. Stochastic Computing   2. Chapter Organisation | 18  18  19 |
| **2** | **LITERATURE REVIEW**   * 1. Literature Survey      1. Arash Ardakani et al., VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing.      2. Zhe Li et al., Towards Budget-Driven Hardware Optimization for Deep Convolutional Neural Networks using Stochastic Computing.      3. Bingzhe Li et al., An FPGA Implementation of a Restricted Boltzmann Machine Classiﬁer Using Stochastic Bit Streams.      4. B.MuraliKrishna et al., FPGA based Pseudo Random Sequence Generator using XOR/XNOR for Communication Cryptography and VLSI Testing Applications.      5. Peng Li et al., A Stochastic Reconﬁgurable Architecture for Fault-Tolerant Computation with Sequential Logic.      6. Peng Li et al., Computation on Stochastic Bit Streams Digital Image Processing Case Studies.      7. Hao Chen et al., Stochastic Computational Models for   Accurate Reliability Evaluation of Logic Circuits. | 20 |
|  | 20 |
|  | 20 |
|  | 20 |
|  | 21 |
|  | 21 |
|  | 21 |
|  | 22 |
|  | 22 |

|  |  |  |
| --- | --- | --- |
|  | * + 1. Shang-Ping Pan Zhao Fang L et al., FPGA Realization of   Activation Function for Neural Network.   * + 1. Karen Simonyan et al., Very Deep Convolutional Networks for Large-Scale Image Recognition. | 23  24 |
| **3** | **STOCHASTIC BASED NEURAL NETWORK**   * 1. Types of Neural Networks      1. Convolutional Neural Network (CNN)      2. Recurrent Neural Networks (RNN)      3. Feedforward Neural Network      4. Radial Basis Function Neural Network      5. Multilayer Perceptron      6. Restricted Boltzmann Machines (RBM)      7. Deep Belief Networks   2. Stochastic Number Format      1. Stochastic Implementation of Basic Arithmetic Operations         1. Complementary Operation         2. Multiplication Operation         3. Addition Operation         4. Subtraction Operation         5. Division Operation   3. Activation Function      1. Linear Activation Function | 25 |
|  | 25 |
|  | 25 |
|  | 26 |
|  | 27 |
|  | 27 |
|  | 28 |
|  | 28 |
|  | 30 |
|  | 30 |
|  | 31 |
|  | 31 |
|  | 31 |
|  | 32 |
|  | 33 |
|  | 33 |
|  | 34 |

|  |  |  |
| --- | --- | --- |
|  | * + 1. Sigmoid / Logistic Activation Function     2. Tanh / Hyperbolic Tangent Activation Function     3. Relu (Rectified Linear Unit) Activation Function     4. Leaky Relu Activation Function     5. Parametric Relu Activation Function     6. Softmax Activation Function     7. Swish Activation Function   3.4 Matlab Software | 36 |
| 36 |
| 37 |
| 38 |
| 39 |
| 39 |
| 39 |
| 40 |
| **4** | **IMPLEMENTATION** | 42 |
|  | 4.1 Flow Chart | 42 |
|  | 4.2 Neuron Implementation | 42 |
|  | 4.3 Training the Neuron | 44 |
|  | 4.4 Activation Function | 45 |
|  | 4.5 Testing the Neuron | 45 |
|  | 4.6 Binary Implementation | 46 |
|  | 4.7 Stochastic Implementation | 47 |
| **5** | **RESULTS AND DISCUSSIONS** | 49 |
|  | 5.1 Binary Process | 49 |
|  | 5.2 Stochastic Process | 50 |

### REFERENCES 54

**APPENDICES**

|  |  |  |
| --- | --- | --- |
| **NO** | **LIST OF FIGURES** | **PAGE**  **NO** |
| 1.1  1.2  1.3  1.4  2.1  2.2  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  3.10  3.11  3.12  3.13  3.14  3.15  3.16  3.17  3.18  3.19  3.20  3.21  3.22 | Structure of Deep Learning Types of Machine Learning Structure of Single Neuron  Lfsr-Based Stochastic Number Generator Three Basic Boolean Logic  Sigmoid Equation  Different Types of Filters for Convolution Image Convolution Functions  Cnn  Feed Forward Neural Nerwork  Radial Basis Function Neural Network Multilayer Preceptron  Restricted Boltzmann Machines Function Of Rbm  Deep Belief Networks Stochastic Sesl Stochastic Not Gate  Stochastic Multiplication Operatio in (A) Unipolar For (B) Bipolar Format (C) Uesl Format (D) Sesl Format  Stochastic Addition Using (A)Mux (B) Or Gate Implementation of Subtraction Operation in Unipolar Format. Implementation of Subtraction Operation in Bipolar Format. Implementation of Division in Unipolar Format Implementation of Division in Uesl Format  Implementation of Division in Sesl Format Activation Function  Linear Activation Function  Sigmoid / Logistic Activation Function Graph  Tanh / Hyperbolic Tangent Activation Function Graph | 14  15  17  19  23  24  25  26  26  27  28  28  29  29  30  31  31  32  33  33  33  34  34  34  35  36  37  37 |

|  |  |  |
| --- | --- | --- |
| 3.23  3.24  3.25  3.26  3.27  3.28  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  4.10  4.11 | Relu (Rectified Linear Unit) Activation Function  Leaky Relu Activation Function Parametric Relu Activation Function Softmax Activation Function  Swish Activation Function Matlab  Error Correction Formula Neuron Code  Weight Matrix Testing Code  Sigmoid Multiplication Sigmoid Activation Code Saving the Training Part Testing Code  Input for The Neuron Lfsr Code  Stochastic Stream | 38  39  39  40  40  43  43  44  44  45  45  45  46  46  47  48  48 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE**  **NO** | **TABLE NAME** | **PAGE**  **NO** |
| 5.1 | Binary Process | 49 |
| 5.2 | Stochastic Process | 50 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ACRONYM** | **ABBREVIATIONS** |
| AI | Artificial Intelligence |
| RBM | Restricted Boltzmann Network |
| SOC | System on Chips |
| DBN | Deep Belief Network |
| DNN | Deep Neural Network |
| SC | Stochastic Computing |
| ANN | Artificial Neural Network |
| LFSR | Linear Feedback Shift Register |
| B2S | Binary to Stochastic |
| B2IS | Binary to Integer Stochastic |

**ABSTRACT**

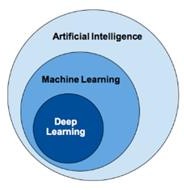
Deep learning techniques have gained prominence in the research world in recent years, but deep learning algorithms have a high computational cost, making them difficult to use for several applications. Calculations can be done faster in DNN and they are highly accurate. The deep neural networks involve huge manipulation of data which forms the building blocks with fundamental arithmetic operations. But it is a challenging task to perform such operations in DNN with less hardware requirements. In this paper, a study has been made on how the hardware is optimized in deep neural network using stochastic computing.

# CHAPTER 1 INTRODUCTION

### INTRODUCTION

Artiﬁcial neural networks usually require a very large number of computation nodes and can be implemented either in software or directly in hardware, such as FPGAs. Software-based implementations running on general purpose processors are the main approach for implementing large Artiﬁcial neural networks. With growing interest in neural networks, researchers look for new methods to achieve low-power and area sufficient neural networks rather than being restricted to the slow, nonportable, software-based approaches. A solution to the limitations of the software-based approaches is an FPGA- based implementation. However, this approach is limited by its high-power consumption and substantial resource requirements. The Restricted Boltzmann Machine is a type of artiﬁcial neural network that can solve difﬁcult problems. Like other machine learning models, RBM has two types of processes – learning and testing. During learning, the system is presented with many input examples and desired outputs to generate a suitable structure that learns a general rule to map inputs to outputs. In the testing process, the outputs for new inputs following the general rule obtained in the learning process. Implementing a fully parallel specially designed hardware version of a neural network in a single FPGA is expensive. To achieve this Stochastic Computing is used. Stochastic computing (SC) exploits the unique characteristics of computations using random streams of bits. This work shows that SC can be used to overcome FPGA hardware limitations. Stochastic bit streams can be encoded using either a unipolar or a bipolar format. The main motivation of using SC is the simplicity of the computational elements involved.

### WORKING OF DEEP LEARNING



**Figure 1.1 Structure of Deep Learning**

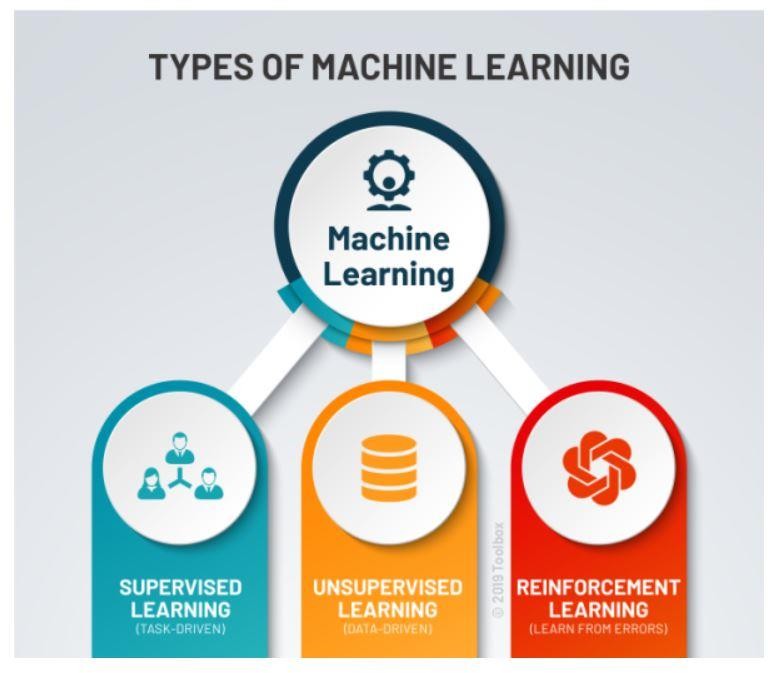
* + 1. **ARTIFICIAL INTELLIGENCE**

Artificial intelligence is a technique that enables a machine to mimic human behaviour. AI relates to human intelligence emulation on computers designed to think like humans and imitate their behaviour. The concept can also be applied to any computer that displays human mind-related characteristics such as thinking and problem-solving. The perfect function of artificial intelligence is its capacity to streamline and take action that has the greatest probability of achieving a particular objective.

### MACHINE LEARNING

Machine learning is a technique to achieve AI through algorithms, train with data. Machine Learning is defined as the study of computer programs that leverage algorithms and statistical models to learn through inference and patterns without being explicitly programmed. Machine Learning field has undergone significant developments in the last decade.

### TYPES OF MACHINE LEARNING



**Figure 1.2 Types of Machine Learning**

### SUPERVISED LEARNING

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labelled data. Even though the data needs to be labelled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances. In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labelled parameters required for the problem.

The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output. This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

### UNSUPERVISED LEARNING

Unsupervised machine learning holds the advantage of being able to work with unlabelled data. This means that human labour is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program. In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings. The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

### REINFORCEMENT LEARNING

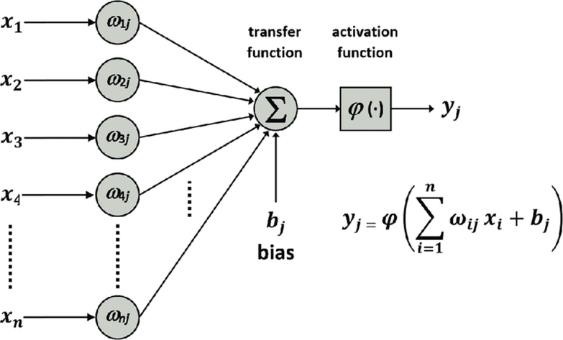
Reinforcement learning directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations

using a trial-and-error method. Favourable outputs are encouraged, or ‘reinforced’, and non- favourable outputs are discouraged or ‘punished’. Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favourable or not. In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favourable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result. In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

### DEEP LEARNING

Deep learning is independent, does not need any algorithm. Requires more volume of data. The field of artificial intelligence is essentially when machines can do tasks that typically require human intelligence. It encompasses machine learning, where machines can learn by experience and acquire skills without human involvement. Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. Similarly, to how we learn from experience, the deep learning algorithm would perform a task repeatedly, each time tweaking it a little to improve the outcome. We refer to ‘deep learning’ because the neural networks have various (deep) layers that enable learning. Just about any problem that requires “thought” to figure out is a problem deep learning can learn to solve.

### NEURAL NETWORKS



**Figure 1.3 Structure of Single Neuron**

A neural network is a sort of computer software, inspired by biological neurons. Similarly, a neural network is made up of cells that work together to produce a desired result, although each individual cell is only responsible for solving a small part of the problem.

### IMAGE PROCESSING WITH NEURAL NETWORKS

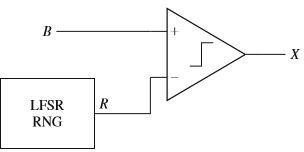
The image processing chain containing the five different tasks

* Pre-processing
* Data reduction
* Segmentation
* Object recognition
* Image understanding

### STOCHASTIC COMPUTING

Stochastic computing circuits are able to realize arithmetic functions with very few logic gates. This is achieved by encoding numerical values within the statistics of random binary sequences. Given a probability P(X), 0≤ PX ≤ 1, the corresponding stochastic number X is a sequence of random binary numbers X0,X1,... for which any Xj ∈{ 0,1}may equal 1 with probability PX. Stochastic numbers are usually produced using a uniform random number generator (RNG) based on a linear feedback shift register LFSR. Stochastic numbers have precision that improves with sequence length, a property called progressive precision. For instance, the value 3/7 can be precisely represented by a sequence of at least length 7, however the value 5/14 requires a sequence of at least 14 bits. Since the individual bits in the sequence are random, a much longer sequence length is required before the sequence’s actual average converges within the desired precision. A K-bit unsigned binary integer B is supplied, indicating a probability PX = B/2K. At each clock cycle, the LFSR-based RNG produces a new K-bit random number R. If R<B then the output for that clock cycle is X = 1, otherwise X = 0.

The below figure 3 shows stochastic number generator that uses a comparator to convert k-bit binary number B to stochastic number X with PX = B/2K.



### Figure 1.3 LFSR-Based Stochastic Number Generator

* 1. **CHAPTER ORGANISATION**

The rest of this project report is structured as follows. chapter 2 deals with the literature survey. Next, chapter 3 gives brief details about the and implementation. chapter

4 cover details of proposed system. The performance analysis based on the design summary and timing analysis is specified in chapter 5. Finally, chapter 6 concludes the report along with discussion about the future works.

# CHAPTER 2 LITERATURE REVIEW

### LITERATURE SURVEY

* + 1. **Arash Ardakani et al., VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing.**

In this paper they have proposed an efﬁcient implementation of a Deep Neural Network based on integral stochastic computing. Virtex7 FPGA is used resulting in 45% and 62% average reductions in area and latency. They produce long latencies. In order to reduce the long latency integral stochastic computing is used. They show that integral stochastic computing results in up to 21% reduction in energy consumption compared to the binary radix implementation at the same misclassiﬁcation rate. They have also implemented using quasi- synchronous implementation which yields 33% reduction in energy consumption comparing with the binary radix implementation without any compromise on performance.

They main idea is to achieve low power, area efﬁcient and to produce less latency in the system. The simulation and implementation result shows that the proposed design reduces the area occupation by 66% and the latency by 84%. They also showed that the proposed design consumes 21% less energy than its binary radix counterpart. They also consumed 33% of the energy with respective to the binary radix implementation by using quasi-synchronous implementation method without reducing the efficiency of the system.

### Zhe Li et al., Towards Budget-Driven Hardware Optimization for Deep Convolutional Neural Networks using Stochastic Computing.

This paper introduced hardware implementation for Deep Convolutional Neural Network using stochastic computing. This paper proposed an automatic design allocation algorithm driven by budget requirement considering overall accuracy performance. An algorithm is proposed that given design budgets, re-structuring the Stochastic computing based Deep Convolutional Neural Network can achieve optimized hardware with high accuracy in performance. Each stochastic computing-based neuron in DCNN is analysed and Experimental results shows that, with restricted design the optimized SC-based implementation is achieved with the lower rate.

### Bingzhe Li et al., An FPGA Implementation of a Restricted Boltzmann Machine Classiﬁer Using Stochastic Bit Streams.

They proposed that Artiﬁcial neural networks requires a very large number of computation nodes. This can be implemented using software and hardware also (FPGA). Software based approaches are not suitable for real time applications, but they support a large number of nodes. But in FPGA based implementation they can significantly speedup the computation time. However, resources are limited in an FPGA it restricts the maximum number of computation nodes in hardware. To overcome this, the stochastic bit streams to implement the Restricted Boltzmann Machine (RBM) handwritten and digit recognition application completely on an FPGA. By doing this it saves a large number of hardware resources and making the FPGA based implementation feasible.

### B.Murali Krishna et al., FPGA based Pseudo Random Sequence Generator using XOR/XNOR for Communication Cryptography and VLSI Testing Applications.

In this paper, the proposed system is used to generate a pseudo random sequence generator using xor and xnor gates. Random numbers are generated using LFSR. The proposed method presents a linear feedback shift register (LFSR) which generates an arbitrary number based on XOR, XNOR gates. Multiplexer is attached to generate a random value at user defined state in runtime. Hardware complexity and power consumption is reduced by replacing the multiplexer with tristate buffers. The proposed LFSR method is designed using XNOR does not require any seed value, whereas XOR based LFSR in conventional method requires ( initial value) seed value, to generate random numbers. Result analysis indicates that proposed LFSR with and without seed value (initial value) gives a better performance, low power consumption and improves more randomness in runtime. One of the applications in lfsr is cryptography. LFSR is used for hiding the data i.e it can encrypt and decrypt with random key appends in an image.

### Peng Li et al., A Stochastic Reconﬁgurable Architecture for Fault-Tolerant Computation with Sequential Logic.

In this paper, they have discussed how to convert binary into stochastic stream (i.e) B2S converter. By using a linear feedback shift register (LFSR) and a comparator, we can convert a value from its binary radix encoding to its stochastic encoding. By comparing the value of lfsr and comparator stochastic bit stream is generated. The concept of stochastic is to make sure that

there is less hardware used and the area. It also consumes less energy, while achieving the same performance in terms of processing time and fault-tolerance. They also describe a reconﬁgurable architecture to perform computation on stochastic bit streams using sequential logic like FSM based implementation.

### Peng Li et al., Computation on Stochastic Bit Streams Digital Image Processing Case Studies.

The reliability of integrated circuits as transistor sizes continue to shrink to nanoscale dimensions is a signiﬁcant challenge for the industry. In this paper, they have introduced new stochastic computational elements based on finite state machines for the task of digital image processing. Then they have presented ﬁve digital image processing algorithms as case studies of practical applications of the technique. They also compared the error tolerance, hardware area, and latency of stochastic implementations to those of conventional implementations using binary radix encoding. They found an analysis of a particular function, namely the stochastic linear gain function.

The experimental results show that stochastic implementations are remarkably tolerant to soft errors and have very low hardware cost. stochastic implementations introduce some variance. Accordingly, this technique can be applied in those applications that mandate high reliability but do not require high precision and can tolerate some variance. For this type of applications, they represent a value by a 1024-bit stochastic bit stream, and it shows accurate results.

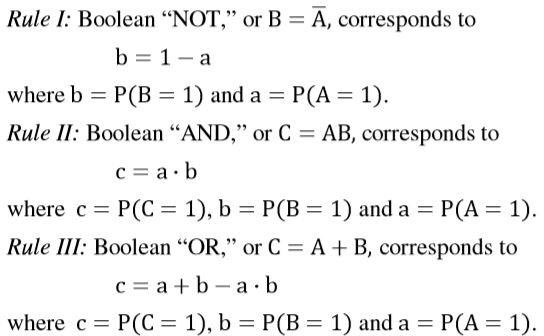
The five types of algorithms are image edge detection, median ﬁlter-based noise reduction, image contrast stretching, frame difference-based image segmentation, and KDE-based image segmentation.

### Hao Chen et al., Stochastic Computational Models for Accurate Reliability Evaluation of Logic Circuits.

This paper presents a model based on stochastic computation, in which probabilities are encoded in the statistics of random binary bit streams, for the reliability evaluation of logic circuits. This accurately determines the reliability of a circuit with its precision only limited by the random fluctuations inherent in the representation of random binary bit streams. Their simulation results demonstrate the accuracy and scalability of the stochastic computational models. The reliability analysis of logic circuits has been based on the probabilistic treatment of

signals in combinational logic networks. The signal probability of an input or output of a logic gate is defined as the probability that the signal is a logical 1. A logic function provides a transform from its input to output probabilities. The reliability of an output is obtained as the probability of the output if the output is expected to be a logical 1, or 0.

There are 3 rules of Boolean logic can be mapped to an arithmetic equation of signal probabilities.



### Figure 2.1 Three Basic Boolean logic

These are the basic rules of Boolean to form any arithmetic equations. This paper is all about the stochastic gates such as AND,OR and NOT. These are the basic gates to form any circuits.

### Shang-Ping Pan Zhao Fang L et al., FPGA Realization of Activation Function for Neural Network.

In this paper they have presented the FPGA implementation of neuron block units based on a sigmoid activation function for artificial neural networks applications. They have also simulated the sigmoid function in Modelsim XE II. The sigmoid non-linearity takes a real valued number its range is between 0 and 1. In particular, large negative numbers become 0 and large positive numbers become 1.

To evaluate the hardware implementation of the neuron block units, the design has been simulated and synthesized using Synopsys design compiler logic synthesis tool. The proposed ANN consists of sigmoid activation function with floating-point arithmetic is realized using FPGA devices. The below figure shows the sigmoid equation.



### Figure 2.2 Sigmoid Equation

* + 1. **Karen Simonyan et al., VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION.**

In this paper they have investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Their main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small 3 × 3 convolution filters. In this they have evaluated very deep convolutional networks up to 19 weight layers for largescale image classification. Their models was made to show a wide range of tasks and datasets, outperforming more complex recognition pipelines built around less deep image representations.

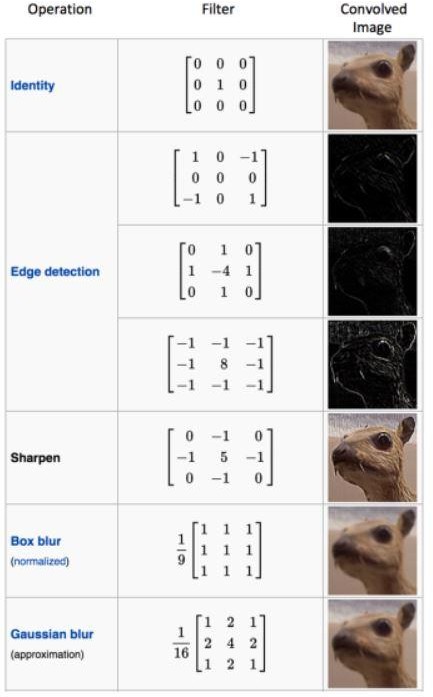
# CHAPTER 3

**STOCHASTIC BASED NEURAL NETWORK**

### TYPES OF NEURAL NETWORKS

* + 1. **CONVOLUTIONAL NEURAL NETWORK (CNN)**

Convolutional Neural Networks (CNN) is one of the variants of neural networks used heavily in the field of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here it simply means that instead of using the normal activation functions defined above, convolution and pooling functions are used as activation functions. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

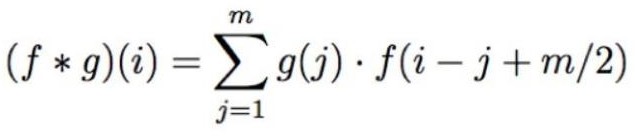


### Figure 3.1 Different Types of Filters For Convolution Image

The above example shows various convolution image after applying different types of

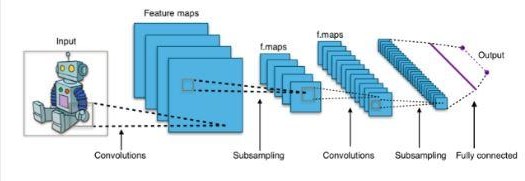
filters.

A convolution of two functions f and g is defined as which is nothing but dot product of the input function and a kernel function.



### Figure 3.2 Convolution Functions

In case of Image processing, it is easier to visualize a kernel as sliding over an entire image and thus changing the value of each pixel in the process. Pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden- layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.



### Figure 3.3 CNN

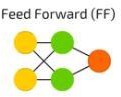
There are 2 main types of pooling commonly known as max and min pooling. Max pooling is based on picking up the maximum value from the selected region and min pooling is based on picking up the minimum value from the selected region. Therefore, a Convolutional Neural Network or CNN is basically a deep neural network which consists of hidden layers having convolution and pooling functions.

### RECURRENT NEURAL NETWORKS (RNN)

Recurrent Neural Networks are a very important variant of neural networks heavily used in Natural Language Processing. In a general neural network, an input is processed through a number of layers and an output is produced, with an assumption that two successive inputs are independent of each other. This assumption is however not true in a number of real-life scenarios. For instance, if one wants to predict the price of a stock at a given time or wants to predict the next word in a sequence it is imperative that dependence on previous observations is considered. RNN has shown to be hugely successful in natural language processing especially with their variant LSTM, which are able to look back longer than RNN.

### FEEDFORWARD NEURAL NETWORK

This is one of the simplest types of artificial neural networks. In a feedforward neural network, the data passes through the different input nodes till it reaches the output node. In other words, data moves in only one direction from the first tier onwards until it reaches the output node. This is also known as a front propagated wave which is usually achieved by using a classifying activation function. Unlike in more complex types of neural networks, there is no backpropagation and data move in one direction only. A feedforward neural network may have a single layer, or it may have hidden layers. In a feedforward neural network, the sum of the products of the inputs and their weights are calculated. This is then fed to the output. Here is an example of a single layer feedforward neural network.



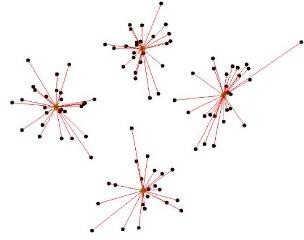
### Figure 3.4 Feed Forward Neural Nerwork

Where yellow colour indicates input cell, green colour indicate hidden cell and orange colour is output cell. Feedforward neural networks are used in technologies like face recognition and computer vision. This is because the target classes in these applications are hard to classify. A simple feedforward neural network is equipped to deal with data which contains a lot of noise. It is also simple to maintain.

### RADIAL BASIS FUNCTION NEURAL NETWORK

A radial basis function considers the distance of any point relative to the centre. Such neural networks have two layers. In the inner layer, the features are combined with the radial basis function.

Then the output of these features is considered when calculating the same output in the next time-step.

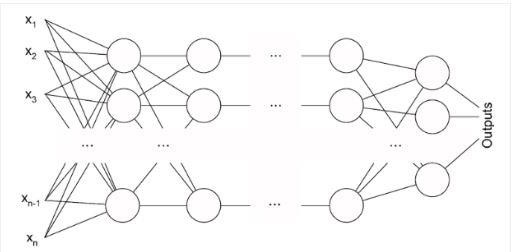


### Figure 3.5 Radial Basis Function Neural Network

The radial basis function neural network is applied extensively in power restoration systems. In recent decades, power systems have become bigger and more complex. This increases the risk of a blackout. This neural network is used in the power restoration systems in order to restore power in the shortest possible time.

### MULTILAYER PERCEPTRON

A multilayer perceptron has three or more layers. It is used to classify data that cannot be separated linearly. It is a type of artificial neural network that is fully connected. This is because every single node in a layer is connected to each node in the following layer. A multilayer perceptron uses a nonlinear activation function.



### Figure 3.6 Multilayer Preceptron

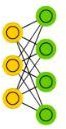
This type of neural network is applied extensively in speech recognition and machine translation technologies.

### RESTRICTED BOLTZMANN MACHINES (RBM)

Restricted Boltzmann Machines are a two-layered artificial neural network with generative capabilities. They have the ability to learn a probability distribution over its set of input. It is used for dimensionality reduction, classification, regression, collaborative filtering,

feature learning, and topic modelling. RBMs are a special class of Boltzmann Machines and they are restricted in terms of the connections between the visible and the hidden units.

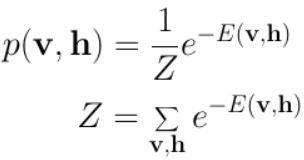
This makes it easy to implement them when compared to Boltzmann Machines.



### Figure 3.7 Restricted Boltzmann Machines

They are a two-layered neural network one is the visible layer and the other one is the hidden layer and these two layers are connected by a fully bipartite graph i.e. means that every node in the visible layer is connected to every node in the hidden layer but no two nodes in the same group are connected to each other.

Where Yellow Colour Is Backed Input Cell and Green Colour Is Hidden Cell Multiple RBMs can also be stacked and can be fine-tuned through the process of gradient descent and back-propagation. Such a network is called a Deep Belief Network. the first step in training an RBM with multiple inputs. The inputs are multiplied by the weights and then added to the bias. Weights will be a matrix with the number of input nodes as the number of rows and the number of hidden nodes as the number of columns. The first hidden node will receive the vector multiplication of the inputs multiplied by the first column of weights before the corresponding bias term is added to it. Restricted Boltzmann Machines are probabilistic. As opposed to assigning discrete values the model assigns probabilities. At each point in time the RBM is in a certain state. The state refers to the values of neurons in the visible and hidden layers v and h. The probability that a certain state of v and h can be observed is given by the following joint distribution.

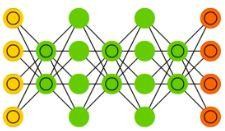


### Figure 3.8 Function Of Rbm

Here Z is called the ‘partition function’ that is the summation over all possible pairs of visible and hidden vectors.

### DEEP BELIEF NETWORKS

Deep belief networks are algorithms that use probabilities and unsupervised learning to produce outputs. They are composed of binary latent variables, and they contain both undirected layers and directed layers. Each layer in deep belief networks learns the entire input. In convolutional neural networks, the first layers only filter inputs for basic features, such as edges, and the later layers recombine all the simple patterns found by the previous layers. Deep belief networks, on the other hand, work globally and regulate each layer in order.



### Figure 3.9 Deep Belief Networks

The network is like a stack of Restricted Boltzmann Machines (RBMs), where the nodes in each layer are connected to all the nodes in the previous and subsequent layer. The connections in the top layers are undirected and associative memory is formed from the connections between them. The connections in the lower levels are directed. The nodes in the hidden layer fulfil two roles they act as a hidden layer to nodes that precede it and as visible layers to nodes that succeed it. These nodes identify the correlations in the data. Applications of Deep Belief Networks are Image recognition, Video recognition and Motion-capture data etc.

### STOCHASTIC NUMBER FORMAT

Initially there were two formats of stochastic numbers, namely unipolar and bipolar formats. In these formats the numerical representation was limited to [0 to 1] and [-1 to +1] respectively. Bipolar format of a stochastic number is deduced from its unipolar format by making a change of variable. The relationship between bipolar and unipolar format of a stochastic number is given as



where p\* is the value of the stochastic number in bipolar format and p is the value of number in unipolar format.

Later on, two new formats of stochastic numbers were proposed called unsigned extended stochastic logic (UESL) and signed extended stochastic logic (SESL). Where the numerical representation was extended to [0 to +∞] and [-∞ to +∞] respectively. In UESL, the stochastic numbers are represented by the ratio of two numbers in unipolar format while as in SESL these are represented by ratio of two numbers in bipolar format. Stochastic numbers in UESL are given as

### x = p/q

where p and q are the stochastic numbers in unipolar format. While the stochastic numbers in SESL are given as

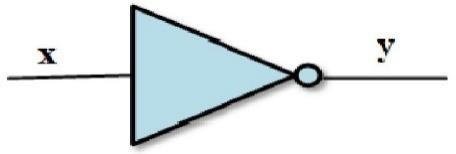


where p\* and q\* are the stochastic numbers in bipolar format. Advantage of UESL and SESL over unipolar and bipolar formats is that they are more robust to noise and hence are more error tolerant.

### STOCHASTIC IMPLEMENTATION OF BASIC ARITHMETIC OPERATIONS

* + - 1. **COMPLEMENTARY OPERATION**

It is implemented in unipolar and bipolar format using a simple NOT gate.

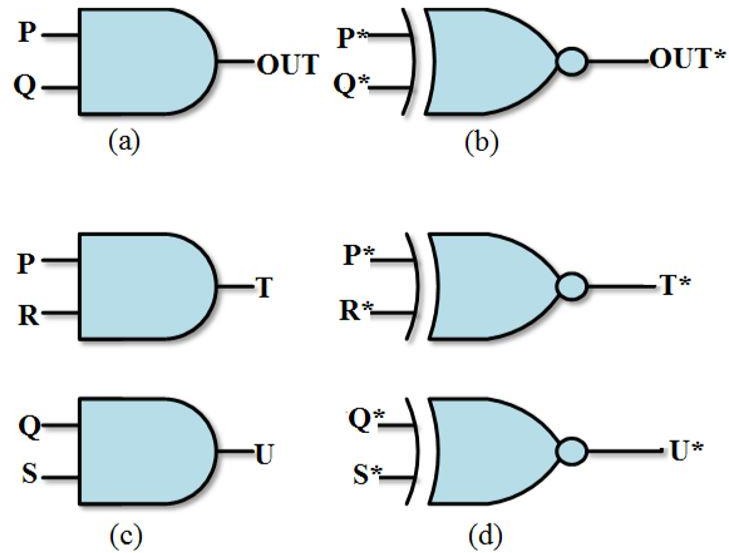


### Figure 3.4 Stochastic Not Gate

* + - 1. **MULTIPLICATION OPERATION**

It is achieved by using a single AND gate in unipolar format, a single XNOR gate in bipolar format, two AND gates in UESL, two XNOR gates in SESL. 4 - bit stochastic multiplier in which linear feedback shift register was used as a random number generator and its results

are highly accurate.



### Figure 3.5 Stochastic Multiplication Operatio in (A) Unipolar Format (B) Bipolar Format

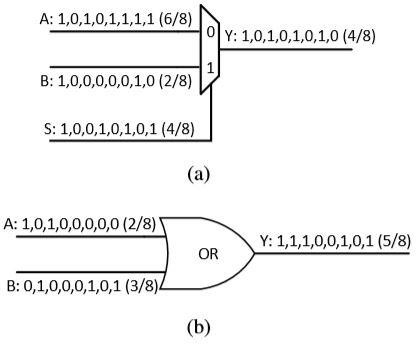
**(C) UESL Format (D) SESL Format**

* + - 1. **ADDITION OPERATION**

Additions in SC are usually performed by using either scaled adders or OR gates. The scaled adder uses a multiplexer (MUX) to perform addition. The output of a MUX Y is given by

Y = A·S + B·(1−S)

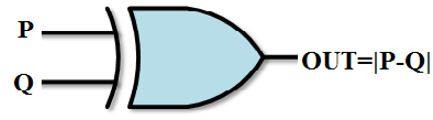
This 2-input scaled adder ensures that its output is in the legitimate range of each encoding format by scaling it down by factor of 2. Therefore, L-input addition can be performed by using a tree of multiple 2-input MUXs.



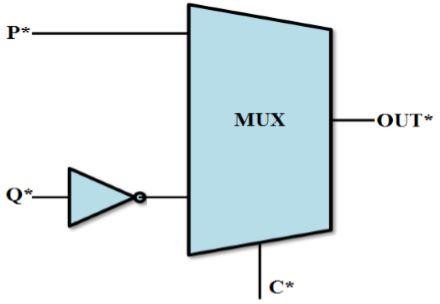
### Figure 3.6 Stochastic Addition Using (a)mux (b) or gate

* + - 1. **SUBTRACTION OPERATION**

In unipolar format, subtraction operation is achieved by single XOR gate where the inputs to this gate must be correlated. However, the output is absolute difference and not the actual difference.



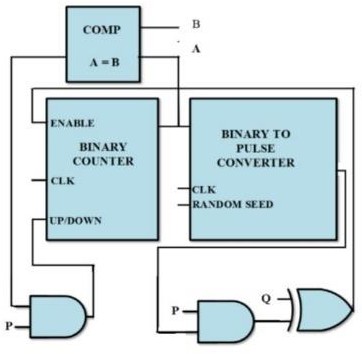
### Figure 3.7 Implementation of Subtraction operation in unipolar format



**Figure 3.8 Implementation of Subtraction operation in bipolar format**

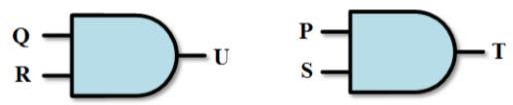
* + - 1. **DIVISION OPERATION**

In unipolar and bipolar format, stochastic divider consists of Up/Down counter, binary to pulse converter, comparator, AND gate and XOR gate.

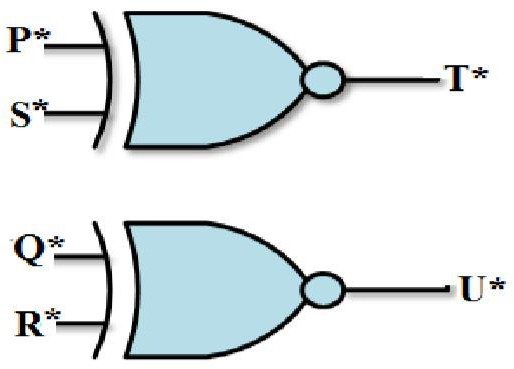


### Figure 3.9 Implementation of division in unipolar format

Therefore, the hardware required for division operation in these formats is much more than in UESL and SESL, where it is achieved by only two AND gates and two XNOR gates, respectively.



### Figure 3.10 Implementation of division in UESL format



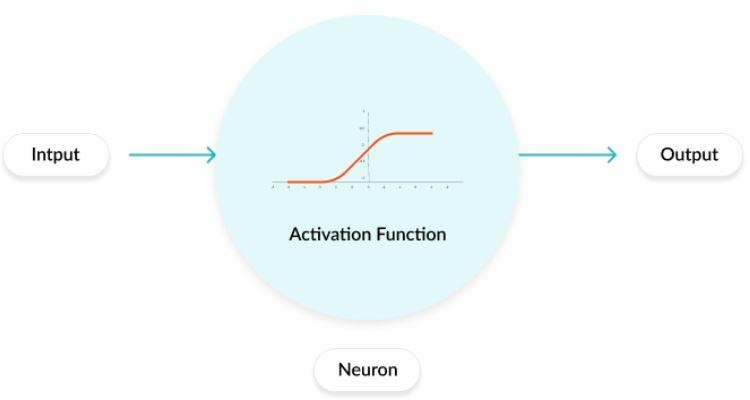
**Figure 3.11 Implementation of division in SESL format**

* 1. **ACTIVATION FUNCTION**

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network and determines whether it should be activated or not, based on whether each neuron’s input is relevant for the model’s prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1. In a neural network, numeric data points, called inputs, are fed into the neurons in the input layer. Each neuron has a weight and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer. The

activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.

Neural networks use non-linear activation functions, which can help the network learn complex data, compute, and learn almost any function representing a question, and provide accurate predictions.



### Figure 3.12 Activation Function

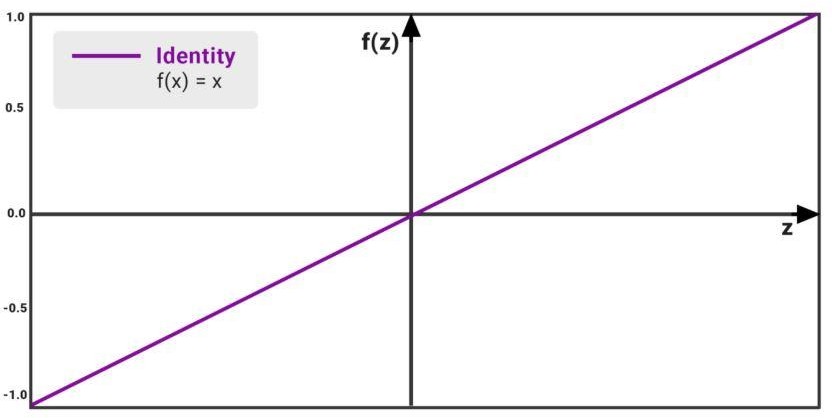
The are 7 types of activation function. They are

* Linear activation function
* Sigmoid / Logistic activation function
* TanH / Hyperbolic Tangent activation function
* ReLU (Rectified Linear Unit) activation function
* Leaky ReLU activation function
* Parametric ReLU activation function
* Softmax activation function
* Swish activation function

### LINEAR ACTIVATION FUNCTION

The value of f(z) increases proportionally with the value of z. The input value is the weighted sum of the weights and biases of the neurons in a layer. The linear function solves the issue of a binary step function where it reports only a value of 0 and 1. The output of the function is not confined between any range; that is, the value of f(z) can go from which not necessarily a problem as we can proceed into the next or final layer by taking the max value of

the neurons that have fired after.



### Figure 3.13 Linear Activation Function

* + 1. **SIGMOID / LOGISTIC ACTIVATION FUNCTION**

It has Smooth gradient, preventing “jumps” in output values. Output values bound between 0 and 1, normalizing the output of each neuron. Clear predictions for X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions. we see that the value of f(z) increases but at a very slow rate. The mathematical reason is that as z (on the x-axis) increases, the value of e exponent -z becomes infinitesimally small and the value of f(z) become equals to 1 at some point.



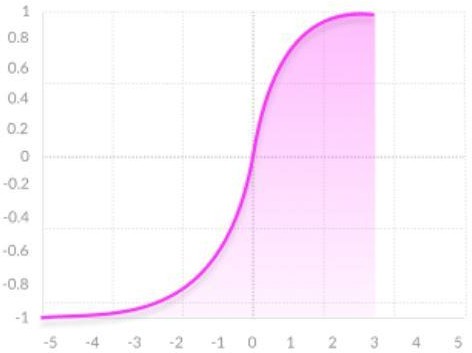
### Figure 3.14 Sigmoid / Logistic Activation Function Graph

It has Vanishing gradient which is very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further or being too slow to reach an accurate prediction. The Outputs is not zero centred. It is Computationally expensive. The output ranges from 0 to 1.

### TANH / HYPERBOLIC TANGENT ACTIVATION FUNCTION

The Tanh function is a modified or scaled up version of the sigmoid function. What we saw in Sigmoid was that the value of f(z) is bounded between 0 and 1; however, in the case of Tanh the values are bounded between -1 and 1.The output ranges from -1 To 1. This is neat in the sense, we are able to get values of different signs, which helps us in establishing which scores to consider in the next layer and which to ignore. But, this function still has the vanishing gradient problem that was seen in the sigmoid function. The model slows down exponentially beyond the

+2 and -2 range. The change in gradient is very small except within this narrow range. the gradient or the derivative of the Tanh function is steeper as compared to the sigmoid function.

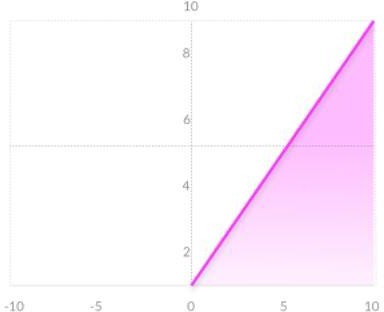


### Figure 3.15 Tanh / Hyperbolic Tangent Activation Function Graph

* + 1. **RELU (RECTIFIED LINEAR UNIT) ACTIVATION FUNCTION**

The Rectified Linear Unit or ReLU for short would be considered the most commonly used activation function in deep learning models. The function simply outputs the value of 0 if it receives any negative input, but for any positive value z, it returns that value back like a linear function. So, it can be written as f(z)=max(0,z). It should be noted that the ReLU function is still non-linear, so we are able to backpropagate the errors and have multiple layers of neurons. This function was quickly adopted, as ReLU took care of several problems faced by the Sigmoid and the Tanh.

The ReLU function has a derivative of 0 over half of its range which spans across all the negative numbers.

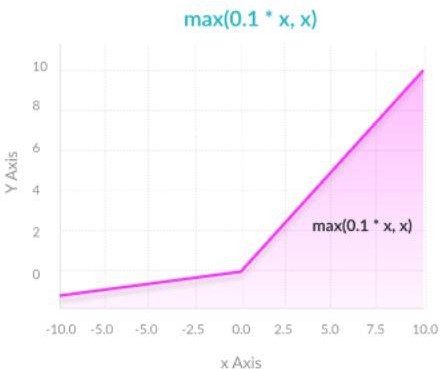


### Figure 3.16 Relu (Rectified Linear Unit) Activation Function

For positive inputs, the derivative is 1. So, we have rectified the ‘vanishing’ gradient problem. The function suffers from the dying ReLU problem. For activations correspondent to values of z< 0, the gradient will be 0 because of which the weights will not get adjusted during the gradient descent in backpropagation. That means, such neurons will stop responding to variations in error/input, so the output network becomes passive due to added sparsity.

### LEAKY RELU ACTIVATION FUNCTION

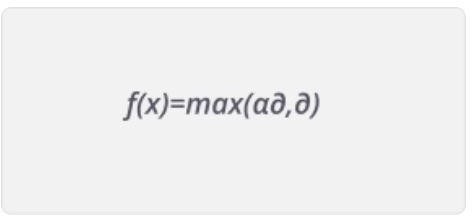
It Prevents dying ReLU problem this variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values. Otherwise it is just like ReLU. The Results are not consistent in leaky ReLU and it does not provide consistent predictions for negative input values.



### Figure 3.17 Leaky Relu Activation Function

* + 1. **PARAMETRIC RELU ACTIVATION FUNCTION**

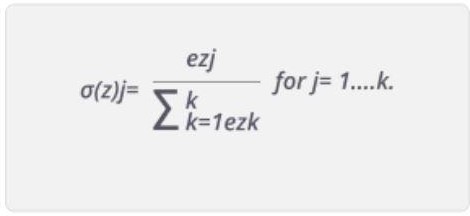
It Allows the negative slope to be learned, unlike leaky ReLU, this function provides the slope of the negative part of the function as an argument. It is, therefore, possible to perform backpropagation and learn the most appropriate value of α. It does not have anything great, its Otherwise like ReLU. It May perform differently for different problems.



### Figure 3.18 Parametric Relu Activation Function

* + 1. **SOFTMAX ACTIVATION FUNCTION**

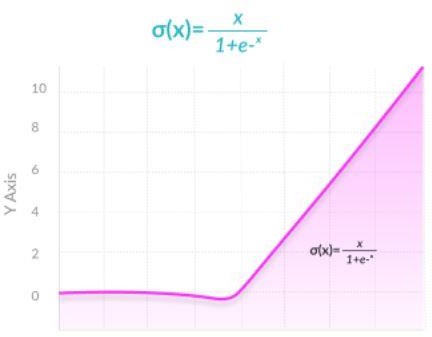
It can handle multiple classes. Only one class in other activation functions normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class. It is Useful for output neurons. Typically, Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.



### Figure 3.19 Softmax Activation Function

* + 1. **SWISH ACTIVATION FUNCTION**

Swish is a new, self-gated activation function discovered by researchers at Google. According to their paper, it performs better than ReLU with a similar level of computational efficiency. In experiments on ImageNet with identical models running ReLU and Swish, the new function achieved top -1 classification accuracy 0.6-0.9% higher. Swish is a smooth, non- monotonic function that consistently matches or outperforms ReLU on deep networks applied to a variety of challenging domains such as Image classification and Machine translation. It is unbounded above and bounded below & it is the non-monotonic attribute that actually creates the difference.



### Figure 3.20 Swish Activation Function

* 1. **MATLAB SOFTWARE**

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages. Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds

graphical multi-domain simulation and model-based design for dynamic and embedded systems.



### Figure 3.21 MATLAB

MATLAB can be used for

* Math and computation
* Algorithm development
* Modelling, simulation, and prototyping
* Data analysis, exploration, and visualization
* Scientific and engineering graphics
* Application development, including Graphical User Interface building The MATLAB system consists of five main parts
* The MATLAB language
* The MATLAB working environment
* Handle Graphics
* The MATLAB mathematical function library
* The MATLAB Application Program Interface (API)

# CHAPTER 4 IMPLEMENTATION

We have decided to compare normal 4 bit binary number with the 4 bit stochastic stream and by giving this as input to the neuron with added weight and bias. We have also implemented sigmoid activation function which gives the output ranging from -1 to 1. After training the neuron with the inputs and the required outputs we have tested with the respective binary numbers such that the trained numbers are evolved as output for the respectively trained inputs.

### FLOW CHART

Neuron Implement

Training

Activation Function



Testing

* 1. **NEURON IMPLEMENTATION**

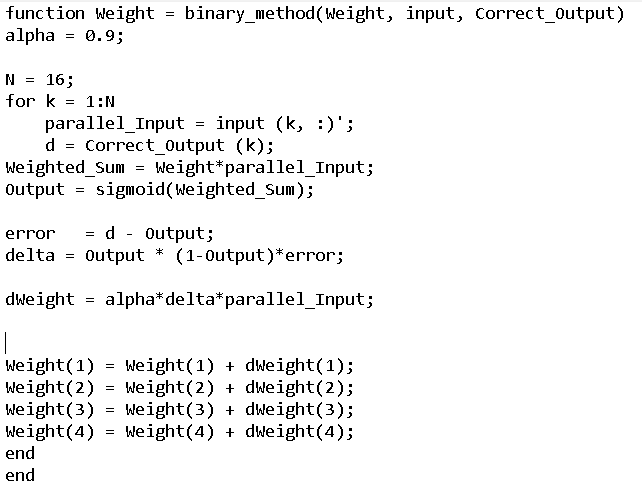
Input of the neuron function is implemented by following weights on that particular neuron and input part followed by the correct output .the weights on that neuron is constant such that all the all the four input part of that neuron is implemented with that constant weight and it is given as 0.9.

The inputs are given as parallel input as the neuron can take only one bit at a time so as we are giving 4 bit input one bit parallel at a time.



### Figure 4.1 Error Correction Formula

If there is any error while testing the neuron the weights on that neuron can be updated using error detection formula such that the previous constant weights can be updated with the new weights till it reaches the correct output with the help of activation function which helps the neuron to adjust the output according to the given input on that neuron. The error rate of that neuron is actually implemented using OR GATE by adding the previous weights with the new weights.



### Figure 4.2 Neuron Code

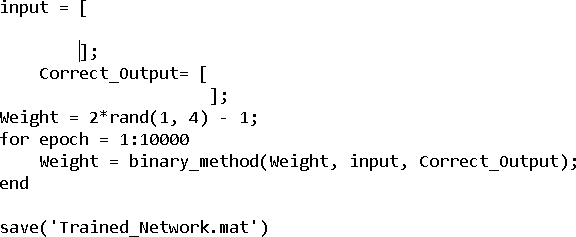
* 1. **TRAINING THE NEURON**

While training the neuron for the network the input which we want to give and the output for that given input are given in a sequence such that while training the neuron that input will correctly match with the given output.



### Figure 4.3 Weight Matrix

The input of the weights while training the neuron is given in a matrix format since there are four input elements present in that neuron and after giving the weight the neuron is tested for about n number of times till it reaches the approximate output. One epoch implies each sample in the training dataset has provided an opportunity to change the parameters of the internal model. It is said to be epoch which me one complete cycle of training the neuron for batch testing all testing samples move simultaneously in one epoch via the learning algorithm until weights are changed. Traditionally the number of epochs is high, often hundreds or thousands, which enables the learning algorithm to operate until the model error has been adequately minimized.



### Figure 4.4 Testing Code

* 1. **ACTIVATION FUNCTION**

To get the correct output for that particular neuron which we use the sigmoid activation function must be multiplied with the weighted sum of that neuron.



### Figure 4.5 Sigmoid Multiplication

This particular function is called finally when the neuron gets updated with weights and bias and the sigmoid activation function is diploid on the whole neuron which acts with the help of some exponential term which makes derives the training part of the neuron with the help of epoch to get the correct trained output.



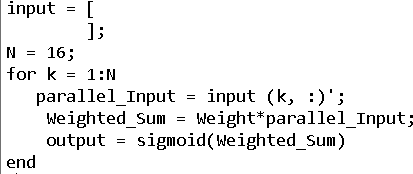
### Figure 4.6 Sigmoid Activation Code

* 1. **TESTING THE NEURON**



**Figure 4.7 Saving the Training Part**

Once the neuron is completely trained with n number of epoch then save the code as trained neuron and then test the trained neuron code in the testing platform. As while we are giving the input as 16 bit while training the neuron for testing also we have to give the input as 16 bit input such that it matches the input of the training part to the input of the testing part.

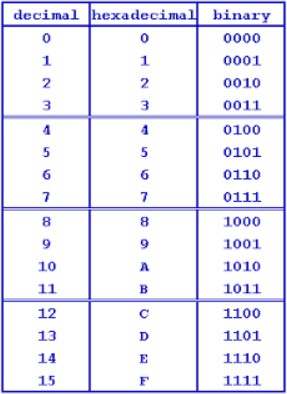


### Figure 4.8 Testing Code

After successful training and testing of the neuron save it and run the code on the output box u can find the output for the particular neuron with the help of input you have trained.

### BINARY IMPLEMENTATION

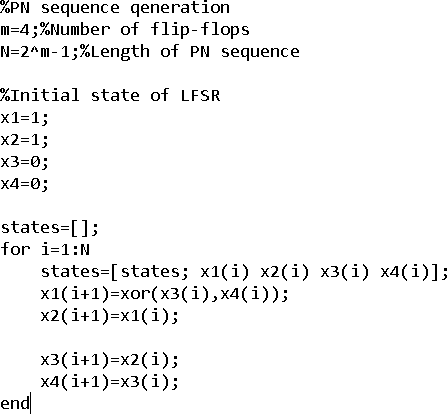
The input for the neuron on is 4 bit binary number [for example number 9 is converted as 1 0 0 1 in the binary number format] and given to the input part of the neuron.



### Figure 4.9 Input for the Neuron

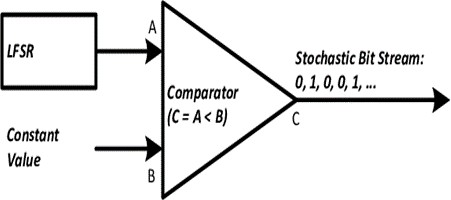
* 1. **STOCHASTIC IMPLEMENTATION**

As we all know that stochastic is the process of counting the number of ones in a bit stream. To create that bit stream we have used LFSR pseudo random generator to generate random numbers.



### Figure 4.10 LFSR Code

and with that code we have given this as code as input to the comparator and on the other end a constant number is set in such a way that the random number which is being generated by 4 bit LFSR gets compared with that constant number and produces the output.



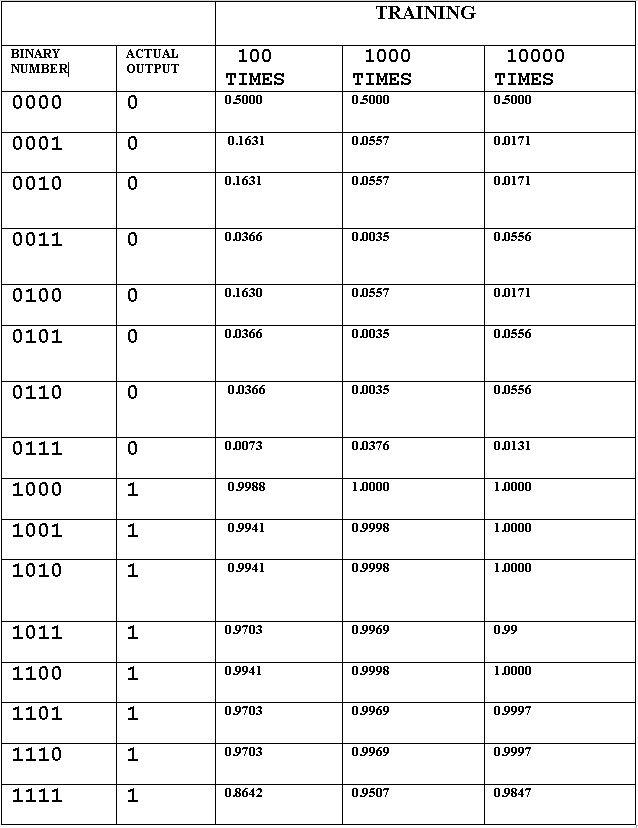
### Figure 4.11 Stochastic Stream

The comparator is given with the operation a<b in which the LFSR provides bit stream from 0000 to 1111 in binary form and the comparator is set with 1001 so if lfsr input becomes less than constant value then input the comparator output becomes zero and if the lfsr input becomes greater than constant value then input the comparator output becomes one and the output of the stochastic is trained with the neuron and tested with the input to get the correct output.

# CHAPTER 5 RESULTS AND DISCUSSIONS

As we train and test the code in the MATLAB platform we attain the results for both binary and stochastic process.

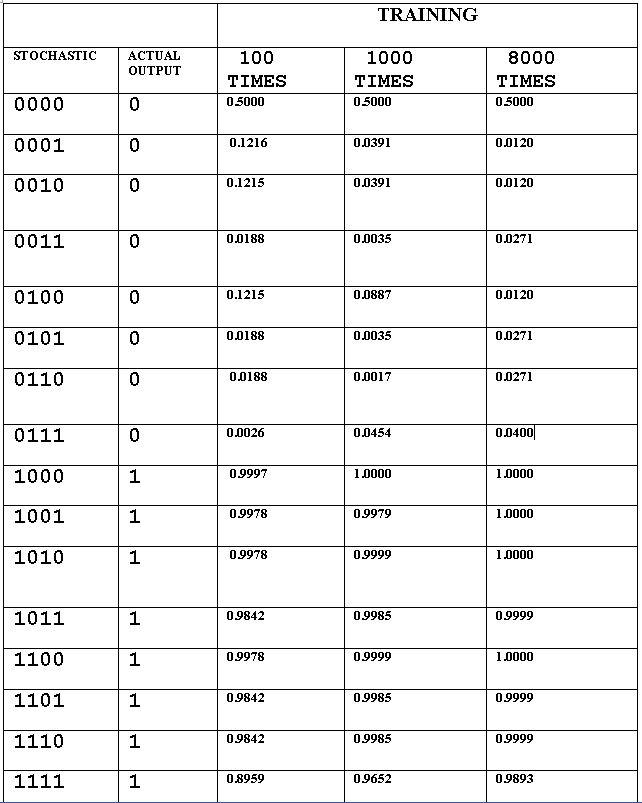
### Binary Process



**Table 5.1 Binary Process**

We train the neuron (epoch) for 100 times to obtain the output but as we train to get the actual output but as we can see the output for 100 times it doesn’t match with the actual output so again by training the neuron for 1000 times and comparing it with the actual output still we didn’t match with the actual output so by again training the neuron for about 10000 times we approximately get the output as the actual one.

### Stochastic Process



**Table 5.2 Stochastic process**

As we train the neuron (epoch) with the stochastic input for 100 times to obtain the actual output but as we can see the output for 100 times it doesn’t match with the actual output so again by training the neuron for 1000 times and comparing it with the actual output still we didn’t match with the actual output so by again training the neuron for about 8000 times we approximately get the output as the actual one. This shows that the normal binary number output which we got is trained for 10000 times but in stochastic the normal binary output which we got is actually trained for only 8000 times which implies that the stochastic is more faster than the normal binary bit stream.

# CHAPTER 6 CONCLUSION

### CONCLUSION

By simulating the code between binary and stochastic we could able to reduce the OR GATE operation in the hardware by 2000 times less than the binary value. When it comes to stochastic it’s not only counting the number of ones in bit stream it is really fast when compared to the normal binary bit stream. The lfsr used in the stochastic bit stream with the help of comparator helps the stochastic input to train faster with the neuron with help of epoch and get the output. The OR GATE operation is reduced 2000 times from 10000 to 8000 while training and testing to get the output and it is done in a way to add the weights with the newly trained weights.

### FUTURE SCOPE

In this project we have used a single neuron to train the inputs and to get that outputs but we can also train this inputs by giving it to the multi-layer neuron, feedforward neuron etc. so that with the help of many hidden layers and by adding that layers with weights and bias we can easily get the outputs in fast manner than that of single layer neuron, we can also get the power and area consumption by dumping the MATLAB code in fpga and find the percentage of consumption that the hard ware have actually consumed.

### REFERENCES

1. VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing Arash Ardakani, Student Member, IEEE, François Leduc-Primeau, Naoya Onizawa, Member, IEEE,Takahiro Hanyu, Senior Member, IEEE and Warren J. Gross, Senior Member, IEEE,2017
2. “Towards Budget-Driven Hardware Optimization for Deep Convolutional Neural Networks using Stochastic Computing” Zhe Li∗, Ji Li†, Ao Ren∗, Caiwen Ding∗,Jeffrey Draper‡†, Qinru Qiu∗,Bo Yuan§, Yanzhi Wang,2018
3. Zihao Yuan, Ji Li, Zhe Li, Caiwen Ding, Ao Ren, Bo Yuan, Qinru Qiu, Jeffrey Draper, and Yanzhi Wang. Softmax regression design for stochastic computing based deep convolutional neural networks. In Proceedings of the Great Lakes Symposium on VLSI, pages 467–470. ACM, 2017.
4. Y. Ji, F. Ran, C. Ma, and D. Lilja, “A hardware implementation of a radial basis function neural network using stochastic logic,” in Design, Automation Test in Europe Conference Exhibition (DATE),

March 2015

1. J. L. Rosselló, V. Canals, and A. Morro, “Hardware implementation of stochastic-based Neural Networks,” in The 2010 International Joint Conference on Neural Networks (IJCNN), July 2010
2. P.-S. Ting and J. Hayes, “Stochastic Logic Realization of Matrix Operations,” in 17th Euromicro Conf. on Digital System Design (DSD), Aug 2014
3. P. Li, W. Qian, and D. Lilja, “A stochastic reconfigurable architecture for fault-tolerant computation with sequential logic,” in IEEE 30th International Conference on Computer Design (ICCD), Sept 2012
4. B. Brown and H. Card, “Stochastic neural computation. I. Computational elements,” IEEE Trans. on Computers, vol. 50, no. 9, pp. 891–905, Sep 2001
5. ] B. Li, M. Najafi, and D. J. Lilja, “An FPGA implementation of a Restricted Boltzmann Machine classifier using stochastic bit streams,” in IEEE 26th Int. Conf. on Application-specific Systems,

Architectures and Processors (ASAP), July 2015

1. D. Ly and P. Chow, “A multi-FPGA architecture for stochastic Restricted Boltzmann Machines,” in Int. Conf. on Field Programmable Logic and Applications,2009