Efficient Convolutional Neural Network Based on Parallel Pipelining

Pooja Samudre[1], Prashant Shende[2], Vishal Jaiswal[3]
[1]*Department of Electronics & Telecommunication,*
[1]*PG (M. Tech.) Electronics & Communication,*
[1]*D.M.I.E.T.R., Sawangi (Meghe), Wardha, India*
[1]pujasamudre23@gmail.com
[2]*Asst. Prof. Department of Electronics & Telecommunication,*
[2]*D.M.I.E.T.R., Sawangi (Meghe), Wardha, India*
[2]ssprashantt@gmail.com
[3]*Asst. Prof. Department of Electronics & Telecommunication,*
[3]*D.M.I.E.T.R., Sawangi (Meghe), Wardha, India*
[3]dr.vishal_jaiswal@rediffmail.com

Abstract- Artificial Neural Networks are computational devices which are emboldened by the human brain for solving the various computing problems. Currently, NN are widely applied to resolving problems in different areas such as: image processing, pattern recognition, robotics etc…. Based on the deep learning algorithms, there are many recent rapid growths of applications. A deep learning algorithm which is expanded from Artificial Neural Networks, and it is extensively used for picture categorization and identification. Still there is no Neural Network based computing technique, that is parallels pipelining technique has been signified in order to assist the existence of deep neural network in terms of accuracy. In this paper, Computing-based convolutional neural network system produced highly precise and productive system by using parallel pipelining. The proposed convolutional neural network is compared with previous convolutional neural network executed on an FPGA using a computing technique to optimize the time delay and power consumption of the system, with high accuracy as compared to previous conventional neural network implementations.

Keyword*-Convolutional Neural Network, Artificial Neural Network, Dataflow Architecture.*

I. INTRODUCTION

The most all-around computers are emanate formed on the basis of von-Neuman architecture is sequential in nature. To resolve exhausting and diverse problems the multiplier perceptrons are being widely applied successfully by training them in the best manner with highly approved algorithm known as error back-propagation algorithm [Haykim 1999].

General purpose processors are inefficient for CNN implementation because of computation design of CNN and it does not give required performance. Therefore, to enhance the performance of CNN many accelerators based FPGA, GPU and even ASIC designs have been introduced lately. FPGA based accelerators have gained more interest of researchers because they have benefits of better result like high energy performance, quick development round and ability of reconfiguration [6].

Demand for deep learning is growing nowadays. It is very important to execute deep neural networks on small devices proficiently. Many appliances like mobile, vehicle, etc. can be used through this system more efficient by deep learning acceleration. Especially only inference is executed on these devices, but also inference requires many numbers of multiple and accumulates working to proceed every input, but the inference algorithm comparatively easy and repetitive, this makes it best phenomenon for hardware acceleration [1].

As a result, much digital hardware design for deep neural network acceleration, in which chip implementation also involves architecture scheme. It is widely followed by a conventional binary representation of the number. It approximates computing along with stochastic computing (SC) and others have been referred as an economist or an energy conservative option to conventional binary implementations, but it is restricted to architecture level analyses than real hardware implementations [3], [2].

Machine learning has attained a significant change lately due to the growth of ANN. This method is biologically guided computational models and it can improve the performance substantially than the earlier type of artificial intelligence. CNN is the best form of ANN architecture, in which complex image driven pattern acknowledgement tasks are mainly fixed by CNN.

Prof. YannLeCun invented CNN in the 90s. A machine learning algorithm is a different form of ANN particularly designed for image analysis and other similar 2D recognizing handwritten digits initially. As time went by, many applications, like artificial vision, data analysis and so on are used successfully by CNN. As a result, CNN is the best application for image recognition categorization [3], [14].

In this demonstration, a proposed system is said to be best for efficiency and correctness is as to compare earlier SC-DNN designs because this system is a full deep neural network (DNN) inference system using a state-of-the-art pipelining. This SC-DNN execution gives better accuracy than earlier FPGA execution of SC-based DNN and exhibit the viability of parallel pipelining, and it is compared to previous FPGA implementation for low cost application. This system is claimed to be the best for proposed work.

We have executed and evaluated our proposed Convolutional Neural Network in VHSIC Hardware Description Language, and also evaluated the accuracy of the proposed system with previous Convolutional Neural Network using a parallel pipelining technique by applying some datasets as inputs. Our experimental results demonstrate that for CNN acceleration, the proposed SC Based-CNN is more delay-efficient in the compute array than the conventional SC while generating more accurate results, and can achieve high speed and lower power consumption compared with simple Convolutional Neural Network implementations of the same accuracy.

II. RELATED WORK

The proposed technique is SC-based deep neural network system. This system uses an input image, in which stochastic computing is used to recognize the input images, which gives same accuracy as conventional binary implementation and to obtain all this CNN is implemented on an FPGA [1]. Here the FPGA based implementations are mostly focused in recent times. There are many suggestions in the literature that shows a same objective.

A tale suggests that substantial CNN with FPGA is computed by using a stochastic based and scalable hardware architecture and circuit design. The purpose is to execute all elements of deep learning CNN, in which multi-dimensional convolution, activation and pooling layers are included [2]. However, the proposed work is also focusing on the different layers of a convolutional neural network in particular to train a network using pipelining.

The work proposes an analytical design system, In which rooine model is operated for optimum result of a CNN design, memory bandwidth using many development methods like a loop tilling and transformation are quantitatively analyzed, and that is computed its throughput and after this rooine model analyzes the system for optimum performance and minimum FPGA resource requirement. As per research, the implemented CNN accelerator on a VC707 FPGA board and compared it to earlier ways [8].

The work proposed in [3] proposes an efficient approximation scheme for hyperbolic tangent function. Regarding the highest number of permissible error like a design standard the approximation of the system depends upon mathematical analysis. In this process hardware implementation of the proposed approximation system is shown. It can be said that suggested system has positive better resulted than earlier architecture in terms of area and delay.

The other approximation scheme for hyperbolic tangent was proposed. The proposed approximation scheme of the system is based on a mathematical analysis considering highest permissible error as a design standard.

The proposed work in [7] evaluated a system ASIC-called a Tensor Processing Unit (TPU) -expanded in data centers since 2015 that stimulates the inference state of neural networks (NN). The TPU`s heart is a 65,536 8-bit MAC matrix multiplies unit that gives a maximum throughput of 92 traps/second (TOPS) and a large (28 MiB) software-managed on-chip memory. The proposed system has compared the TPU to as ever-class Intel Has good CPU and an NVIDIA K80 GPU, In which same data centers are contemporarily deployed. The proposed workload, which is written in the high-level TensorFlow framework, By using production NN applications (MLPs, CNNs, and LSTMs) that represent 95% of their datacenter's NN inference demand.

The work in [9] proposed for the operation on the compressed model which performs better implementation, this work is suggested for higher efficiency of an efficient inference engine that works on a compressed network model and stimulates which consequent in sparse matrix-vector multiplication with weight sharing. If it is compared with DaDianNao engine then EIE has 2.9×, 19× and 3×, better throughput, energy efficiency and area efficiency and area efficiency.

The author presented EIE, an energy-efficient engine optimized to operate on compressed deep neural networks. EIE is used to minimize energy requirements for computing a typical FC layer by 3,400 × compared with the GPU and it is obtained by leveraging sparsely in both the activation and the weights and having benefits of weight and quantization.

The presented work in [10] supports value based method for accelerating DNN in hardware and presented the Cnvlutin (CNV) DNN accelerator architecture. CNV is exhibited as conversion over the state-of-the-art DNN accelerator DaDianNao, the main concepts that runs CNV design carry wide working possibilities. The CNV design works as encouragement for extra exploration like combining CNV, which exploits other valuable properties of DNNs.

The proposed work in [11] addressed the two serious issues of SC-based CNNs, and it is done by introducing a novel SC multiply algorithm and its vector development, SC-MVM (matrix-vector multiplier), in this system SC multiple gives a flawless outcome and for this SC multiples needs a few cycles only. And this is significantly cheaper than the conventional SC based method. After the research results indicate that CNNs designed for MNIST and CIFAR-10 datasets and SC based CNN method and 40X~490X more accurate and more productive in computation as compared to conventional SC-based method. At the same time it gives lower area delay and takes less energy than bit width-optimized fixed-point implementations of the exact precision.

III. METHODOLOGY

Most of the work in the literature has focused on various techniques for data flow acceleration of CNNs. Here the suggested architecture for the data flow acceleration of CNNs are presented, by explaining how every layer is implemented and detailing how a complete network is constructed.

A typical Convolutional Neural Network Figure 1 consists of millions of neurons where they are organized in several layers. The beginning layer is Convolution layer and the last few layers, are Fully Connected. The Fully Connected Layer also named as Classifier. The layers between convolution layer and fully connected layer are called hidden layers. The main purpose of the convolution layer is to extract image features, then drive them into the hidden layers of computing, and extract the results through the output layer. Layers among hidden layers, usually, such as pooling layers (max, average etc), are sub-sampling layers, are partially connected, while the output layers are fully connected. Between the hidden layers often there are activation functions that help to keep valuable information for next layers.

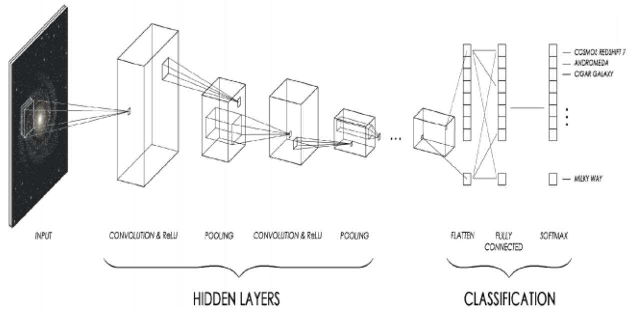

Figure 1: Architecture of CNN

Mathematical function framed as a model of biological neurons, a neural network is known as artificial neurons. In artificial neural network artificial neurons are fundamental units. The artificial neuron takes one or more input and sums them to generate an output.

A. Neuron Implementation

A neural network is able to take and represent complex input/output relationships as it is a robust data-modeling tool. It highly interrelates ingredient computational units. The model of the nervous systems of human exalted them so they are called neuron. Each computational unit (see Figure 2) consists a set of input connections that get signals from other computational units and a bias adjustment, a set of weights for each input connection and bias adjustment and a transfer function that transforms the sum of the weighted inputs and bias to decide the value of the output from the computational unit. The linear combination of all signals from each connection (Xi) times the value of the connection weight between node j and connection i(Wji) is the sum value of the computational unit (node j). As we consider a literature interchangeably the term artificial neuron is used with: node, unit, processing element or even computational unit. When modeling the framework objects, the term neuron is considered in order to maintain the analogy to the network structures. As per necessity, the neuron is applied as input to the beginning layer of the network, known as an input layer by using the terms convolution.

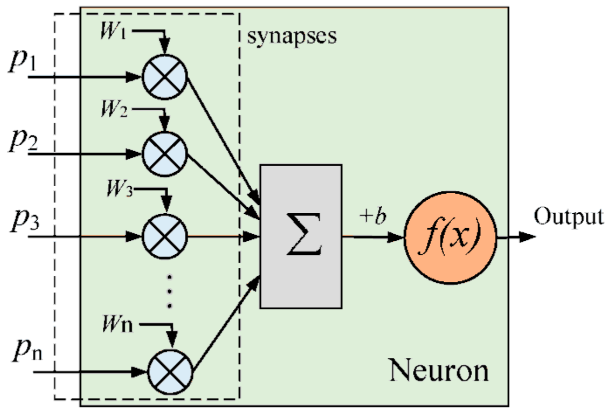

Figure 2: Basic Neuron Module

Figure 2 shows the common mathematical module of a neuron. According to the above, here we consider the first net input as ith input and the output (yj) as a jth output, and then first net input to the ith unit of the next layer from the jth node can be written as:

$$\text{Net } j = \Sigma_i X_k W_{ik} \text{ …………(1)}$$

Where: Xk: is the input to the node. Wjk : is the weight associated with each input to the node from input k to node j. This sum-of-products calculation is an important in the network simulations. The speed at which this calculation can be performed usually determines the performance of any given network simulation because there is often a very large number of interconnects in a network. To calculate an activation function , first the net input is calculated once. The determination of the output of the function is as follows:

$$yj = f(\text{net } j) \text{ …………………(2)}$$

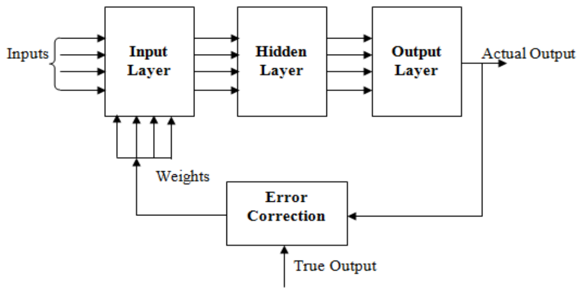

Figure 3: Overview of Convolution Neural Network

A pipeline is used to improve the performance. In this case, the neural network is divided into two phases, and every phase can process a various set of input signals at the same time.

The proposed work is first to develop a neuron in the neural network, this neuron can perform basic operations like a simple ALU. The neuron will then be expanded into the input layer for taking inputs into the system. This layer will contain multiple neurons for better performance.

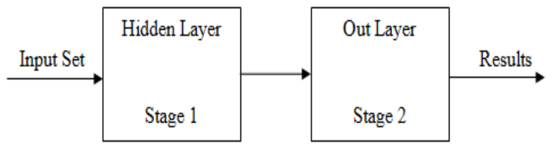

Figure 4: The two stages of the network

The hidden layer will receive the set of input values when it will finish the process the results from the hidden layer will send to the output layer. At the same time, the hidden layer will receive a new input set and both layers will operate with different values. When both layers will finish their tasks the hidden layers send the results to the output layer again and will accept different input set. At the same time, the output layer will show the results and will receive the result from the hidden layer. Through the use of this technique, the neural net can work with two different sets of input values and a better performance can obtain.

The same process that was applied to the neural network can be applied to each neuron. A neuron can be divided into two stages and can process two different sets of values in each stage. The neuron can be divided as follows:

The first stage will be constituted by the set of multipliers, and the second stage will be constituted by the adder and the circuit of the transfer function, as it can be seen in below figure.

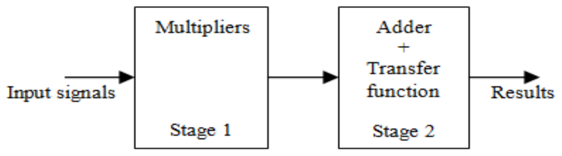

Figure 5: A Pipeline Nature

Through the use of pipeline neurons in a two stages pipeline network, a four-stage pipeline circuit can be obtained. The synchronization of all the circuits with pipeline can be done by the use of clock signals. Using the four stages pipeline network, for various sets of input signals can be proceed in this circuit simultaneous.

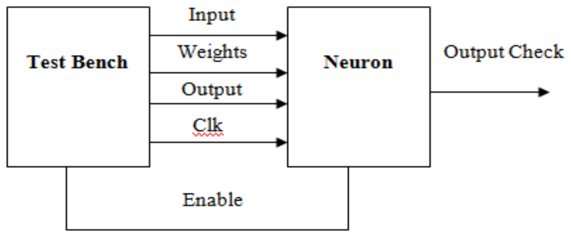

Figure 6: Testing Structure of Neuron

B. Testbench

When describing digital circuit, there is a need to test the accuracy of implementation with a testbench. A test bench is commonly a non-synthesizable VHDL file which reiteratively appeals an order of regulated inputs of a circuit and analyzes its concrete output against the proposed output. If a difference is found, an error is shown in the VHDL simulator`s log which can then be advised to operate direct a designer search for the issue in the circuit`s RTL presentation [3].

The pipelining technique based design of the implementation of a convolutional neural network is used to reduce the time delay and power consumption.

C. Pipelining

The more efficient way to improve the overall processing performance of a convolutional neural network is pipelining. Various instructions can be executed at the same time in this computing technology in order to improve the performance of CNN. Pipelining is transparent to the programmer, by overlapping the execution process of instructions it exploits parallelism at the instruction level. It is analogous to an assembly line where workers perform a special task and pass the partially completed product to the next worker[3].

D. Convolution neural network

CNN is able to use less number of parameters to capture translational invariance. It can be implemented by weights replication over frequency and time. The implementation of a convolutional neural network is the substitution method apart from using a deep neural network. It can be said that CNN provides better efficiency compared to a deep neural network. Thus, CNN improves the limitation occurs in a deep neural network from where it requires large network size and a large number of training samples if the size is adequate. Furthermore, deep neural network caused the input topology to be ignored [3] [12].

Due to the representation of the input in a fixed order, this situation has occurred thus, the performance of networks is not affected. CNN local correlation for modeling provides advantageous towards other fields and spectral representation of speech shows high correlation [13]. The layers of CNN are fully connected at the top and it may contain one or more convolutional layer.

E. The Back-Propagation Learning

In proposing system we have presented a neuron implementation using a very suitable algorithm, that is back-propagation algorithm. The back-propagation algorithm is one of the most practical algorithms of CNN training. A back propagation neural network follows by a feed-forward topology, supervised learning and back propagation learning algorithm. This algorithm is powerful for training the network as it’s a general purpose learning algorithm. There are several back propagations in the neural network. Relatively simple form of optimization which are known as gradient descent, modifications of BP are conjugate gradient and Newton's methods is the basis of the back propagation algorithm. Since back propagation is based on a relatively simple form of optimization known as gradient descent, modifications of BP are conjugate gradient and Newton's methods. Back propagation algorithm has two primary virtues, first one is; it is simple and easy to understand, another one is; it works for a wide range of problems. Due to its characteristics "basic" back propagation is still the most widely used. Basic back propagation consists following steps based on figure 3:

Step1. Initialize weights and offsets. Adjust all weights and node offsets to small random values.

Step2. Present input and desired output. The desired output is 1. The input should be new on each trial or samples from a training set.

Step3. Calculate actual outputs. Use the sigmoid nonlinearity formulas to calculate outputs.

Step4. Adapt weights.

Step5. Repeat by going to step 2.

F. VHDL Implementation

To implement proposed system, VHDL language is used. In VHDL language, it is possible to implement and simulate a digital system which forms a high level of distraction and with important facilities of modular design, as VHDL is a hardware description language which simplifies the development of complex systems [16]. Features of VHDL are used to allow electrical characteristics of circuit behavior (such as rise and fall times of signal, delays through gates, and functional operation) to describe precisely. In large circuits the resulting VHDL simulation models are then used as building blocks in order to obtain simulation. The individual modules which are obtained by dividing back propagation algorithm logic have been implemented in VHDL using behavioral modeling. The different modules are: synapse, neuron, error generator at the input, error generated at the output, weight update unit and a weight transfer unit. At the input layer all the inputs are multiplied by the corresponding weights in order to get actual output and the weighted inputs are summed together.

Finally, to pass on the value of updated weights to the actual weights a weight transfer unit is used. A final entity having structural modeling in which all the entities are port mapped that has been formulated. The results constitute simulation of VHDL codes of different layers (i.e. Input layer hidden layers and output layer) in a Model-Sim simulator. The simulation waveform of the structural layers shows that the neural network is learning and the output of the each layer is approaching the target. VHDL is used because, this code is valid to program in FPGA.

In this system, the suggested architecture for optimizing the performance of CNNs, by explaining the way of implementation of every layer and the detail construction of a complete network of pipeline training. A design is constructed from various independent modules, in order to reduce time delays and to optimize power consumption. This feature is implemented to operate the execution of CNNs on FPGAs with this technique. In addition to this a standard architecture offers flawless optimizations regarding every particular layer of the network.

IV. RESULT & SIMULATION

Here in proposed work, first of all single neuron has implemented shown below in figure 7, which is subjected as an input to the input layer. Before passing signal to the input layer, signal gives to test bench. The activation function is passed through an activation function to produce the output of the neuron.

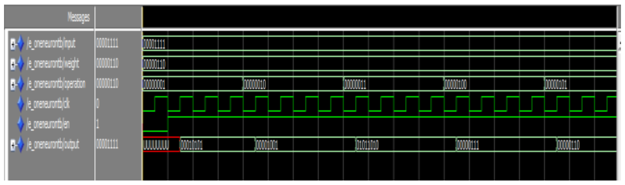

Figure 7: Simulation Waveform of Single Neuron

Likewise, the workflow for the artificial neural network totally depends on the input layer as input layer is the very beginning of the NN. The input layer of a neuron is framed of artificial input neurons shown below in figure 8, and for further processing; it brings the initial data into the system with subsequent layers of artificial neurons. Hence the output waveform of the input neurons is shown below.

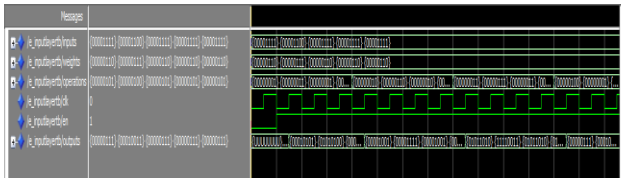

Figure 8: Simulation Waveform of Input Neurons

Output layer is an important layer of the artificial neural network as the overall workflow of the network depends on output layer; it is a final layer of the network. The output layer of the network is framed of artificial input neurons, learnable weight and biases. Each neuron receives several inputs, takes a weighted sum over them and passes it through an activation function and responds with and output as shown in below figure 9.

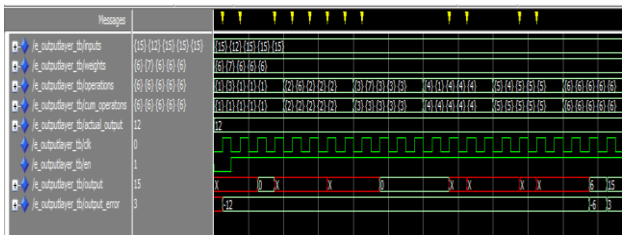

Figure 9: Simulation Waveform of Output Layer

The resulting waveform of the convolutional neural network has shown in below figure 10 and figure 11. The waveform which has been shown in figure 10 is the resulting waveform of CNN before it trained and the figure 11 shows the resulting waveform of CNN after training in Parallel pipeline technique. After comparing both the waveform the speed of the network has been improved by 29%, so the delay of network has been reduced.

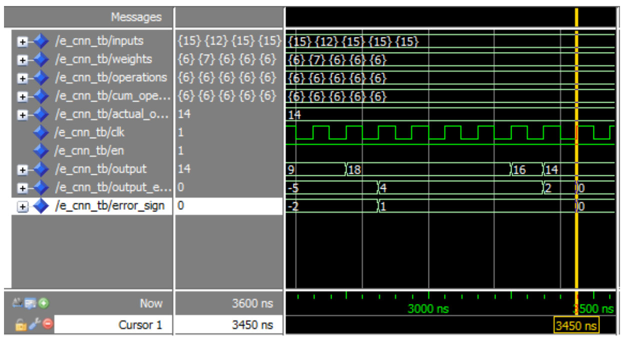

Figure 10: Resulting Waveform of CNN

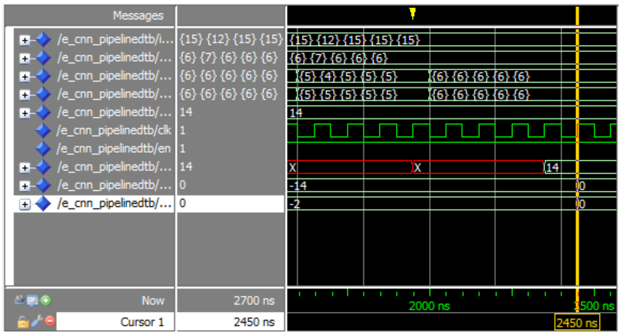

Figure 11: Resulting Waveform of Trained CNN

V. CONCLUSION

The proposed work to implement a convolutional neural network with the approach based on Parallel pipelining technique, as it improves the overall performance of a neural network by training the system with a high-level pipeline between the different network layers and numbers of artificial input neurons have been applied to the input layer of the network. The work is focused on time delay and power consumption in order to optimize the overall performance of a neural network. After training the proposed system by parallel pipeline technique the speed of the proposed convolutional neural network has been improved by 29% faster than that of the previous Convolutional neural network. So the delay of the proposed system has been reduced and the network had become more efficient.

REFERENCES

[1] Daewoo Kim, Mansureh S. Moghaddam Hossein Moradian, Hyeonuk Sim, Jongeun Lee, Kiyoung Choi, *"FPGA Implementation of Convolutional Neural Network Based on Stochastic Computing"*, 978-1-5386-2656-6/17/$31.00©2017 IEEE.

[2] Mohammed Alaward, and Mingjie Lin, *"Stochastic-Based Multi-Stage Streaming Realization of Deep Convolutional Neural Network"*,978-1-5090-5404-6/17/$31.00©2017 IEEE.

[3] Pooja Samudre, Prashant Shende, Vishal Jaiswal, "Optimizing Performance of Convolutional Neural Network Using Computing Teqchnique". In IEEE 5th International conference for convergence in technology,Pune, March 2019.

[4] Babak Zamanlooy, and Mitra Mirhassani, *"Efficient VLSI Implementation of Neural Networks With Hyperbolic Tangent Activation Fuction"*, 1063-8210©2013 IEEE.

[5] A. Krizhevsky *et al.*, *"Imagenet classification with deep convolutional neural networks,"* in Advances in Neural Information Processing Systems 25, F. Pereira et al., Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[6] F. N. Iandola *et al.*, *"Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size,"* CoRR, vol. abs/1602.07360, 2016.

[7] Chen, Yu-Hsin and Krishna, Tushar and Emer, Joel and Sze, Vivienne, *"Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks,"* in ISSCC 2016, 2016, pp. 262–263.

[8] N. P. Jouppi et al., *"In-Datacenter Performance Analysis of a Tensor Processing Unit,"* ArXiv e-prints, Apr. 2017.

[9] C. Zhang et al., *"Optimizing fpga-based accelerator design for deep convolutional neural networks,"* in FPGA *'15*. New York, NY, USA: ACM, 2015, pp. 161–170.

[10] S. Han *et al.*, *"Eie: Efficient inference engine on compressed deep neural network,"* in ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 243–254.

[11] J. Albericio *et al.*, *"Cnvlutin: Ineffectual-neuron-free deep neural network computing,"* in Proceedings of the 43rd International Symposium on Computer Architecture, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016.

[12] H. Sim and J. Lee, *"A new stochastic computing multiplier with application to deep convolutional neural networks,"* in 54th Annual ACM/IEEE Design Automation Conference (DAC '17), Jun. 2017, pp. 29:1–29:6.

[13] LeCun, Y. and Y. Bengio, 1995. *"Convolutional Networks for Images, Speech, and Time Series"*. The Handbook of Brain Theory and Neural Networks, 3361(10): 1-14.

[14] LeCun, Y., F.J. Huang and L. Bottou, 2004. *"Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting"*. In the Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp: 1-8.

[15] Introduction to convolution neural network (online) https://en.wikipedia.org/wiki/Yann_LeCun.

[16] Back-propagation learning in CNN and VHDL Implementation (online) https://www.ijser.org

Pooja H. Samudre received her B.E.(Electronics) degree in 2016 from D.M.I.E.T.R., Wardha. She is currently pursuing her M.E.(Electronics and Communication) degree from D.M.I.E.T.R., Wardha.

.

Prashant Shende received his B.E.(Instrumentation) degree in 2002 from Govt. College of Engineering, Chandrapur, MBA(HR) degree from CAMS in 2007 and M.Tech(VLSI) degree in 2013 from PCE, Nagpur. Presently he is working as Assistant Professor in department of Electronics and Telecommunication Engineering, D.M.I.E.T.R., Wardha. His Area of Interest is VLSI and Embedded System.

Vishal Jaiswal received his B.E.(Electronics and Telecommunication) degree in 2009 from BNCE, Pusad, M.Tech(Electronics) degree from BDCE, Wardha in 2012. Presently he is working as Assistant Professor in department of Electronics and Telecommunication Engineering, D.M.I.E.T.R., Wardha. His Area of Interest is VLSI.