World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

# FPGA implementation of AAD pooling unit and performance analysis

Rajamahanti Meher Kiran *, Toram Naga Jahnavi, Yarabati Mohana Rao, Yamala Sudheer and Udatha Prudhvi Naga Durgesh

*Electronics and Communication Engineering, GMRIT, Rajam, Andhra Pradesh, India.*

World Journal of Advanced Research and Reviews, 2022, 16(01), 697–704

Publication history: Received on 14 September 2022; revised on 19 October 2022; accepted on 22 October 2022

Article DOI: https://doi.org/10.30574/wjarr.2022.16.1.1085

## Abstract

Convolutional Neural Network (CNN) has been witnessing a massive growth for its various applications in different fields. It is a category of Neural Network or Deep learning that is being used in text detection, image classification, etc. It is also very effective for the classifications which are non-image like audio classifications, signal data classifications, etc. CNN is composed of convolutional layer, pooling layer and finally the fully connected layer. In this work, we mainly focus on pooling layer which impacts accuracy and speed of CNN. This work implements validation of CNN based CIFAR-10 classifier and Field Programmable Gate Array (FPGA) implementation of Absolute Average Deviation (AAD) Pooling unit. Pooling techniques like max pooling, average pooling, mixed pooling, min pooling, etc., are currently being used. This work uses AAD pooling in CNN to support that this pooling is having higher accuracy, i.e., 89% and less computational complexity. A benchmark CNN structure using TensorFlow is adopted to quantify the performance of the AAD pooling unit. Further, Register Transfer Level (RTL) coding is done in Verilog HDL and the testbench is developed. The FPGA implementation is carried out using the Xilinx Vivado tool.

**Keywords:** VLSI; FPGA; Convolutional Neural Network (CNN); Deep learning; Absolute Average Deviation (AAD) pooling

## 1. Introduction

The world is filled with lots of data in the form of pictures, texts, audio files, etc. Machine learning is being used for deriving meaning from all that data by classifying, detecting [1], [2] and tracking [3] the data. Convolutional Neural Networks (CNN) is most commonly used and is known for handling huge unstructured data. The data will be passed through each layer of CNN chronologically, i.e., input, convolutional layer, pooling layer, fully connected layer and then output [4]. Convolution and pooling layers keep on repeating until the input data is scaled down to the required size. The typical Convolutional Neural Network architecture is shown in Figure1 for pictorial representation.

Convolution and pooling are the two operations used to extract the features. Convolution is the stage where input feature representations are learned. There are several convolutional layers in the convolutional layer. In order to change the filters one pixel at a time, we utilize convolution with stride 1. The pooling, which acts as the second feature extractor, makes up the second component. Through down sampling, the resulting feature maps dimension is decreased.

Pooling layer is used to reduce the dimensions of feature maps. In general, mostly 2 × 2 sized filters and 3 × 3 sized filters are used in CNN. There are various pooling layers existing and are being used in CNN architectures. Max pooling, Average pooling, Min pooling are most commonly used.

* Corresponding author: Rajamahanti Meher Kiran
Electronics and Communication Engineering, GMRIT, Rajam, Andhra Pradesh, India.

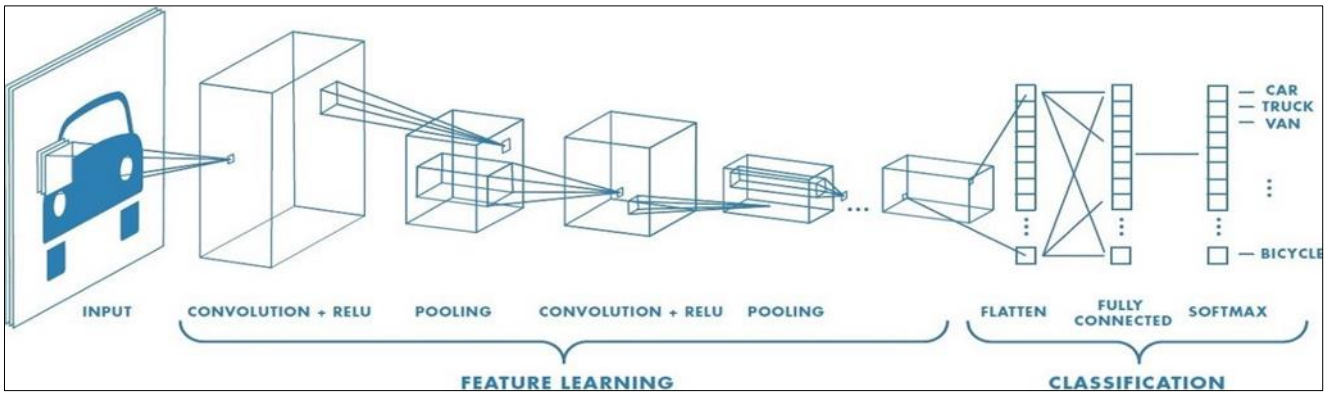


**Figure 1** Typical Convolutional Neural Network (CNN) Architecture

For achieving better accuracy of CNN architecture, extendibility, efficiency and equivalence are the parameters having high importance. To serve this purpose, a pooling method was introduced which normalizes average pooling layer with the help of Zernike moments [5]. It was also proven that this pooling method has the capability to achieve higher accuracy than that of AlphaMEX and Stochastic in every possible case. But this pooling technique has some limitations which was reflected in case of moments in higher order resulting in overfitting of the data. Also, it was having inefficient implementation which can be improved further.

Max pooling [6] is ideal for implementation of the hardware and makes use of the non-pooled data's maximum value. By reducing the offset errors of the values which are assessed from the convolutional layer, and by preserving more texture information, it achieves higher accuracy when compared with average pooling. The limitation of the max pooling is the accuracy, because only the highest value of activation is considered and remaining values are ignored. Also, max pooling always tends to choose brighter pixels. The features that are produced after max pooling are substantial, making overfitting simple to produce. The network resulting, however, has a limited capacity for generalization.

Average pooling [7] is another technique which uses average value for the available patches of the feature map. Translation invariance is added in a minor way, which means most pooled units output values are not considerably affected when the image is translated by a minor amount. In contrast to Max Pooling, which extracts more apparent features, i.e., like edges, feature extraction is smooth in case of average pooling.

Min pooling is another kind of most used pooling technique where the minimum value in an area serves as a representation for the overview of the features in that region. Due to min pooling's tendency to choose darker pixels, it is typically utilized only in images with light backgrounds. Min pooling is also comparatively having less accuracy than other pooling techniques [8]. In contrast in the case of Max pooling, Min pooling only considers least value of activation and remaining all values are not considered, which results in less accuracy and loss of features.

The main objective of this work is to increase the performance of CNN. Generalizing the pooling functions is one of the methods in which max pooling and average pooling is combined and formed a mixed pooling layer [9]. This improvement in performance costs in slight increase in the complexity in computations.

In PPA analysis, ‘A’ represents accuracy. In existing deep neural networks, artificial turned out to be a requirement which results in the decrement of accuracy in recognition. In order to eliminate this requirement, Spatial Pyramid Pooling [10] stands out to be the solution for visual representation.

Computational complexity is another hurdle for CNN that has to be overcome. Pruning stands out to be the solution for this problem but partially, because of its irregularities, accelerators are unable to use the resources of sparsity. In order to get rid of this problem, a sparse wise dataflow [11] can be used for skipping the cycles of MAC which are of weight zero. This results in the avoidance of unwanted computations and minimizes the energy requirements.

The standard pooling methods like max pooling, min pooling and average pooling are having some backdrops. In general, max pooling and average pooling results in loss of the features. Most especially if the input feature is small, then it will result in some serious loss of features. So, in order to overcome this issue, right mix of these pooling layers which result in generalized pooling [12] will be the solution.

The Absolute Average Deviation (AAD) pooling method [13], is one such kind where the absolute average is calculated for the deviations of adjacent pixel values. This pooling strategy attains higher accuracy than existing traditional pooling methods. Also, in case of classification problems, separability can be achieved and that too with comparatively higher precision. This pooling technique calculates the deviation between vertical pixel values and finds the average value of all the obtained deviation values.

In order to show that Absolute Average Deviation (AAD) pooling is having higher accuracy and more advantageous than other existing pooling techniques, Validation of CNN can be done and it results that AAD pooling is attained with higher accuracy.

## 2. Methodology

### 2.1. Absolute Average Deviation (AAD) Pooling

Just like the placement of other pooling methods, AAD pooling is also placed right after the convolution stage. After passing the input data through convolution and pooling stages, it will be followed through fully connected layer for classification which is the final layer in CNN and it is shown in Figure2. The sequence of convolution and pooling layers keeps on repeating until the feature map obtains required down sampled reduced dimensional values.

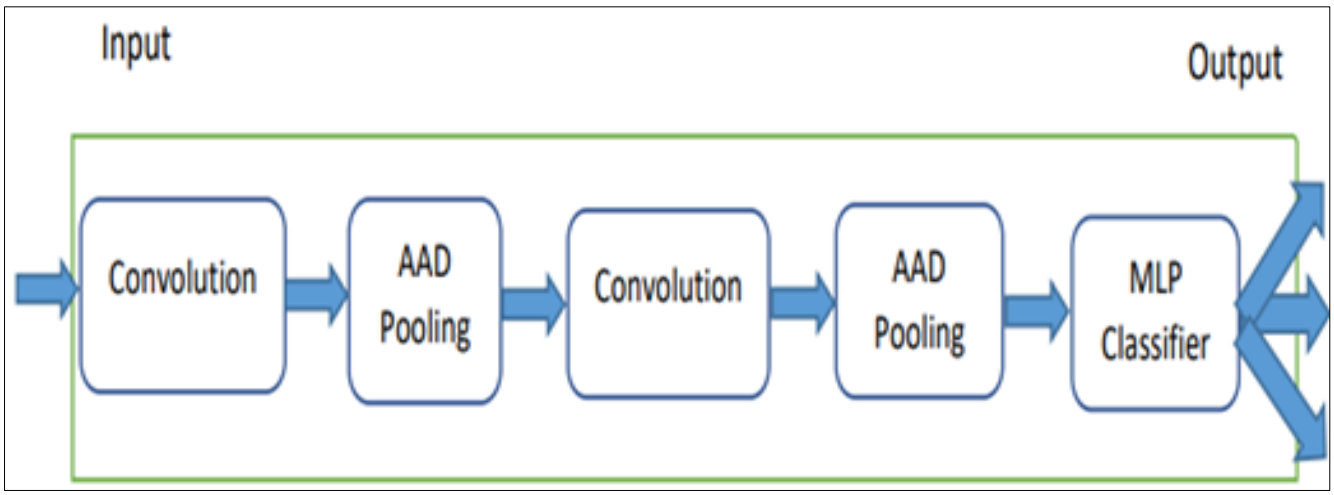


**Figure 2** Placement of AAD pooling in CNN

Most commonly used pooling layers are of 2 × 2 size with stride 1 and 2 × 2 size with stride 2. AAD pooling technique includes calculating deviations from the pixel values which should be absolute and then calculating averages of it. This pooling technique can be implemented in three ways. Those are by calculating deviations with horizontal pixel values, or by calculating deviations with vertical pixel values, or by calculating deviations with both horizontal and vertical pixel values.

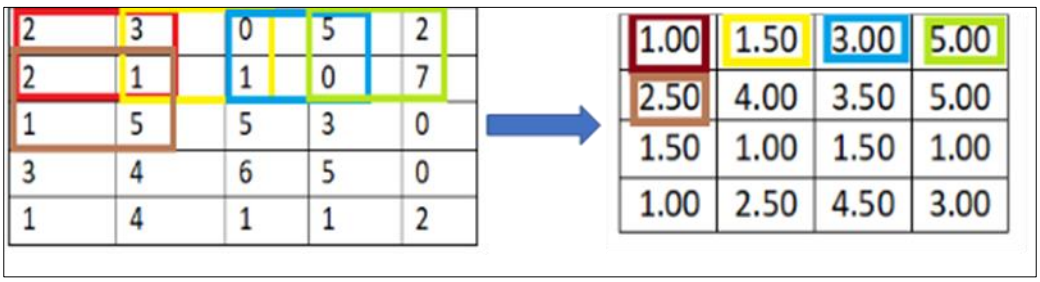


**Figure 3** Example of AAD pooling of size 2 x 2 with stride 1 with vertical deviation

As shown in Figure 3., AAD pooling of size 2 x 2 with stride 1 with vertical deviation can be done as follows:

- Step 1 Firstly, 2 x 2 matrix values will be considered based on the pooling size which is 2 x 2. If we consider Figure 3. as input matrix, then this step refers to 2, 3, 2 and 1 values.
- Step 2 Now as a second step, calculate vertical deviations of those matrix values. i.e., |2-2|= 0 and |3-1|=2.
- Step 3 Sum up the vertical deviations which are obtained in the previous step. i.e., 0+2=2.
- Step 4 As there are two computations involved, the average value of the obtained sum of deviations can be calculated by dividing it with 2. i.e., 2/2 =1.

These steps can be repeated sequentially for obtaining remaining matrix values of pooling output.

This work mainly focusses on the hardware implementation of AAD pooling unit and the method which we focused on is calculating the deviations with vertical pixel values and then finding averages of those values which gives the resultant down sampled output of this pooling unit. In order to show that this pooling technique achieves higher accuracy than other existing traditional pooling techniques, validation of CNN is performed in Python.

The workflow of this project can be explained as shown in Figure4. As a first step, System specifications (shortly termed as spec) are analyzed. Based on the specifications and flow of data, architecture is designed based on the required operations that are to be done to get the final pooling unit output. After making sure that the architecture is working properly, behavioral modelling is the next step followed. Now, Register Transfer Level (RTL) code for the designed hardware architecture is developed using Verilog Hardware Description Language and testbench is developed using Verilog HDL for obtaining simulation and synthesis results.

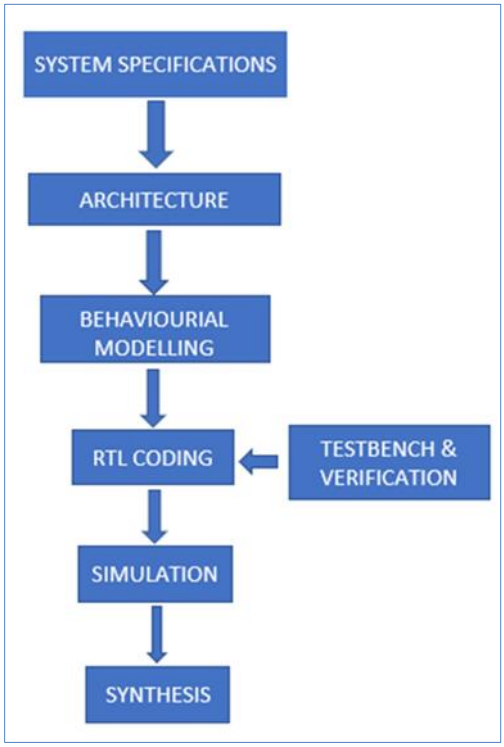


**Figure 4** Workflow of the project

### 2.2. Validation of CNN Classifier

After understanding the functionality of AAD pooling technique, Python code is written for AAD pooling unit. Python code for an existing benchmark CNN structure is adopted, where the structure is consisting of MAX pooling layers. When executed the code with CIFAR-10 dataset [14,15], it resulted in 89% of accuracy. Now, MAX pooling part in Python code is replaced with the AAD pooling method and executed again with CIFAR-10 dataset and observed that the accuracy improved and is higher than 89% accuracy.

### 2.3. Dataset

CIFAR-10 dataset is a frequently dataset consisting of collection of 10 different classes images. It is mostly used to train computer vision algorithms and this dataset is one of the most popular datasets for machine learning algorithms. 60,000

color images of size 32x32 in 10 different classifications make up the CIFAR-10 dataset. The ten categories include trucks, frogs, horses, deer, dogs, cats, birds, cats, ships, and cars. Each class has 6,000 photos total. 10000 images for testing and 50,000 images for training are available in this dataset. Object recognition algorithms in computers frequently pick up new skills through practice. A collection of photos called CIFAR-10 can be used to train a computer to recognize items. The low-resolution images in CIFAR-10 make it possible for researchers to swiftly test out various methods to determine what works.

### 2.4. Hardware Implementation

The hardware architecture for AAD with two inputs has been developed. Three stages—a division stage, an absolute stage, and a subtraction stage—make up the circuit. To obtain the difference between the two inputs, the subtraction operation is applied to the two inputs. Two paths lead to the output. The comparator circuit receives the first one. To get the absolute value of the result of the subtraction, utilize the comparator. The comparator circuit evaluates its input against a "0" threshold value. The comparator results a positive number if the input is positive. If the input applied is negative, then the comparator results in negative one.

As shown in fig 5., Subtraction Absolute (SA) block consists of a subtractor, buffer, and a comparator with threshold at zero. The key component of the Absolute Average Deviation (AAD) pooling approach is the Subtraction-Absolute (SA) module. The results are divided by M after the outputs from the SA modules have been added together to determine the total value of deviation. Depending on the filters applied, different feature maps were produced after convolutional layer computations. The AAD approach achieves a higher level of classification accuracy while providing accurate representation of data. Verilog Hardware Description Language is used to implement every hardware module. The comparator's job is to supply the same sign as the input. A buffer receives the second branch of the subtraction output. For a multiplication operation, this buffer is utilized to synchronize the comparator output and the buffer output.

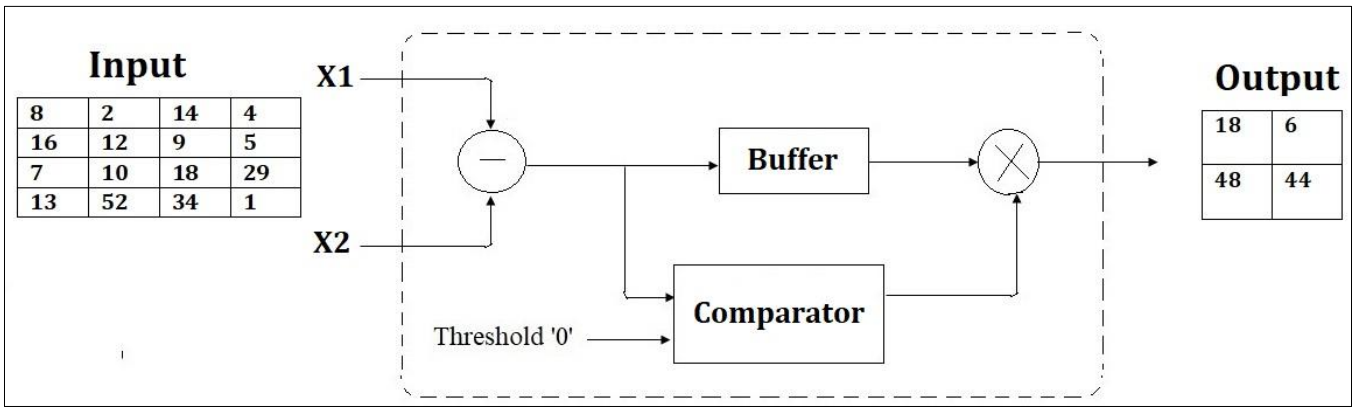


**Figure 5** Subtraction - Absolute (SA) block (by considering AAD pooling layer of size 2 x 2 with stride 2)

The absolute deviation is calculated by multiplying the comparator and buffer outputs. If the outcome of the comparison is positive, the output of the comparator is also positive, and the outcome of the multiplication is also positive. If the comparator's output is negative, the result of the multiplication is positive even though the result of the difference is negative. As a result, this stage generates the absolute deviation, and the output is obtained by dividing the result by two.

A sliding window-based approach is implemented which depicts the AAD approach architecture based on sliding windows with certain stride. The size of the sliding window used over the data is based on the pooling size. The stride determines how far the window moves. Most commonly, stride 2 is used in order to keep the hardware implementation simple and to avoid unwanted pixel values resulting less complexity. The input is used to drive the subtractor, which calculates the difference between pixels. The adder's output is saved in a register. In order to obtain the final sum of deviation, the subtractor output is added to the previously saved data in the register. As a final stage, the output of the accumulator is divided by M. Where M=N(N-1), here N represents the size of the pooling layer. The whole AAD pooling unit hardware architecture is shown in Figure6.

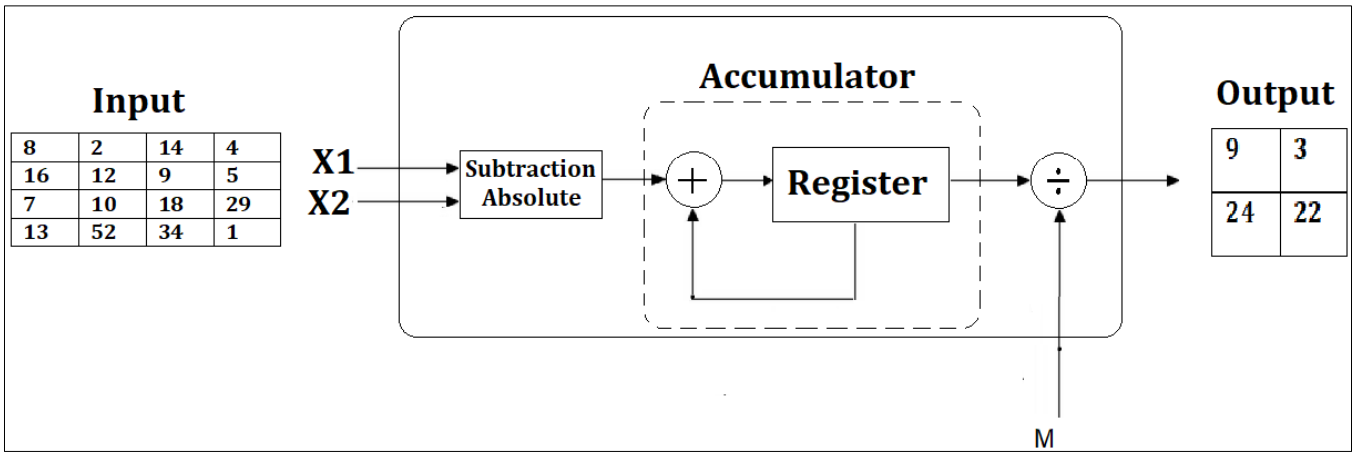


**Figure 6** Hardware Architecture of AAD pooling of size 2 x 2 with stride 2

## 3. Results and discussion

After implementing AAD pooling in python using TensorFlow on CIFAR-10 dataset and comparing its accuracy with existing traditional pooling techniques, i.e., max pooling and average pooling, we observed that AAD pooling is comparatively having higher accuracy, i.e., 89% and is shown in table 1.

**Table 1** Comparison of AAD pooling with other pooling techniques

| Pooling | dataset | Accuracy |
|---|---|---|
| Average pooling | CIFAR 10 | 76 |
| Max pooling | CIFAR 10 | 77 |
| AAD pooling | CIFAR 10 | 89 |

Hardware architecture as shown in fig 5 is implemented in Xilinx Vivado using Verilog Hardware Description Language (HDL). Simulation results of the architecture is shown in Figure 7 where x1 and x2 are 8 bit binary inputs applied, AAD_out is the final output of pooling unit and diff, buf_out, comp_out, tot_dev are intermediate outputs of subtractor, comparator and SA (Subtraction Absolute) blocks respectively.

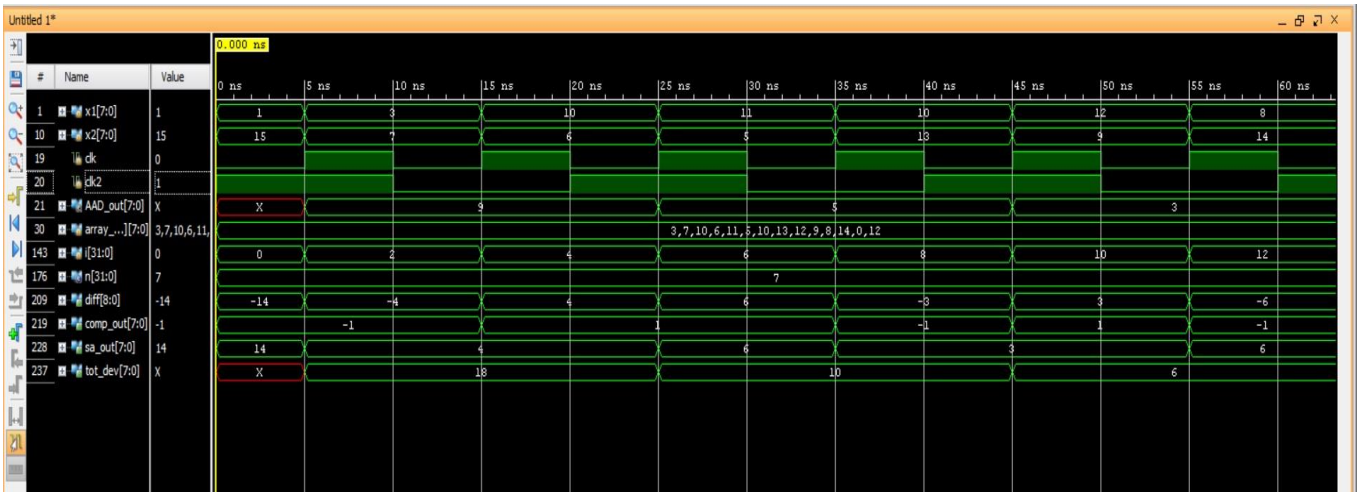


**Figure 7** Simulation result of AAD pooling

Table 2 shows the post implementation results describing the availability and usage of Look Up Tables (LUT), Flip flops (FF), Input Output (IO) and Buffer (BUFG).

**Table 2** Post implementation results

| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 45 | 53200 | 0.08 |
| FF | 8 | 106400 | 0.01 |
| IO | 26 | 200 | 13.00 |
| BUFG | 1 | 32 | 3.13 |

## 4. Conclusion

This work showed the importance of pooling layer in Convolutional Neural Networks which is a deciding factor for attaining better accuracy of the CNN. Functionality of the Absolute Average Pooling layer is explained in detail along with its hardware architecture and the blocks used in it. This AAD pooling is implemented using Verilog HDL and observed the simulation waveform and utilization of various resources with minimum utilization percentage which symbolises that the architecture of AAD pooling has less complexity, utilises less area and also consumes less power when compared with other traditional pooling layers. AAD pooling is also validated on CIFAR-10 dataset and observed 89% accuracy which is greater than existing pooling layers like max pooling and average pooling. Hence the existing pooling layers can be replaced with AAD pooling layer in the CNN architectures to achieve better classification or recognition and lot more applications with higher accuracy.

## Compliance with ethical standards

*Acknowledgement*

This project is done in VLSI lab of Department of Electronics and Communication Engineering of GMR Institute of Technology. We are grateful to our project guide Dr. L. Govinda Rao, Associate Professor in GMR Institute of Technology and to our college management for providing us the required software tools for this project.

*Disclosure of conflict of interest*

Hereby, the authors declare no conflicts of interest.

## References

[1] Huimin Qian, Jiawei Xu and Jun Zhou. Object Detection Using Deep Convolutional Neural Networks. IEEE Access, 2018 Chinese Automation Congress (CAC).

[2] Xiaoming Xi, Zhilou Yu, Zhaolei Zhan, Yilong Yin and Cuihuan Tian. Multi-Task Cost-Sensitive-Convolutional Neural Network for Car Detection. IEEE Access, Vol 7, 2019.

[3] Madoka Asai, Gan Chen, Isao Takami. Neural network trajectory tracking of tracked mobile robot. 16th International Multi-Conference on Systems, Signals & Devices (SSD), 2019.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.

[5] Thomas Theodoridis, Kostas Loumponias, Nicholas Vretos and Petros Daras. Zernike Pooling: Generalizing Average Pooling Using Zernike Moments. IEEE Access, Vol 9, July 2021.

[6] Bin Wang, Yu Liu, WenHua Xiao, Zhihui Xiong, Maojun Zhang. Positive and negative max pooling for image classification. IEEE International Conference on Consumer Electronics (ICCE), 2013.

[7] Luna Sun, Zhenxue Chen, Q. M. Jonathan Wu, Hongjian Zhao, Weikai He, Xinghe Yan. AMPNet: Average- and Max-Pool Networks for Salient Object Detection. IEEE Transactions on Circuits and Systems for Video Technology, Vol 31, 2021.

[8] Tuba Pala, Ugur Guvenc, Hamdi Tolga Kahraman, Ibrahim Yucedag, Yusuf Sonmez. Comparison of Pooling Methods for Handwritten Digit Recognition Problem. International Conference on Artificial Intelligence and Data Processing (IDAP), 2018.

[9] Chen-Yu Lee, Patrick Gallagher, and Zhuowen Tu. Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 40, April 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.

[11] Chaoyang Zhu, Kejie Huang, Shuyuan Yang, Ziqi Zhu, Hejia Zhang and Haibin Shen. An Efficient Hardware Accelerator for Structured Sparse Convolutional Neural Networks on FPGAs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020.

[12] Kuankuan Hao, Shukuan Lin, Jianzhong Qiao, and Yue Tu. A Generalized Pooling for Brain Tumor Segmentation. IEEE Access, Vol 9, 2021.

[13] Kasem Khalil, Omar Eldash, Ashok Kumar and Magdy Bayoumi. Designing Novel AAD Pooling in Hardware for a Convolutional Neural Network Accelerator. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol 30, No.3, March 2022.

[14] Raveen Doon, Tarun Kumar Rawat, Shweta Gautam. Cifar-10 Classification using Deep Convolutional Neural Network. IEEE Punecon Conference Paper, 2018.

[15] Xinche Zhang. The AlexNet, LeNet-5 and VGG NET applied to CIFAR-10. 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE), 2021.