Indian Journal of Radio & Space Physics
Vol. 50, June 2021, pp. 57-63

Implementation of Neural Networks in FPGA

Jayanthi B* & Lakshmi Sutha Kumar

Department of Electronics and Communication Engineering, National Institute of Technology Puducherry, Karaikal 609 609, India

Received: 12 February 2021; Accepted: 5 March 2021

Artificial Intelligence (AI) refers to the recreation of human intelligence in machines that have been designed to think like humans and mimic their actions. AI has been used in many fields such as image processing, health care, education, and marketing. Machine Learning (ML) has been the sub-division of AI, and deep learning has been the subdivision of ML. Artificial Neural Network has been the most predominantly used deep learning technique. While implementing the ANN technique, knowing whether the implementation could have been done in hardware or software becomes necessary, which is essential to achieve the expected performance. This paper gives a survey on the available methods in which the ANN architecture has been implemented to achieve efficient output with minimal resources. It is vital to study and analyze various strategies for implementation and their functionality. This paper has also explained the advantages and disadvantages of different implementation techniques that allow selecting the most appropriate hardware and respective methodology for optimizing the hardware.

Keywords: Artificial intelligence, Artificial neural network, Convolutional neural network

1 Introduction

We are living in a world where automatic process is gaining importance day by day, the technological advancements have changed all the manual operations to automations. There are many automatic applications in our day to day life. Some of those applications in real life are the music recommender systems, Google maps, Uber etc. These are some of the amazing technological advancements in the field of automation which are powered by artificial intelligence. Artificial intelligence enables a machine to decide and act without any human intervention. There are some more important concepts like machine learning and deep learning. Machine learning is a subdivision in a larger family of artificial intelligence. It enables the machine itself to learn and improve the result without explicit programming. Then comes deep learning, a subdivision of machine learning which enables a computer model to filter the input data through layers to predict and classify information. The deep learning system functions as that of the human brain. Some of its architectures include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Recursive Neural Networks (RvNN).

Artificial Neural Network (ANN) is one the rapidly emerging and predominantly used methods for a wide range of applications that include banking and online transactions, transportation (driverless cars), electronics and digital media platforms, aerospace and defense, medical research tools (for reusing drugs), and high machinery industries. Hence, it is important to know and determine the ways in which ANN has been implemented.

ANN Architecture is implemented either using software or hardware. Software implementation has a lower implementation cost. The software libraries are readily available in many programming languages. It is easy to run those software libraries in available computer systems as they do not require any hardware updates. Therefore, it is easy to implement ANN Architecture in the software.

Even though the software implementation has many advantages, it also has some disadvantages[1]. Since the data processing is carried out in a pseudo parallel way, the actions implemented are not efficient and quick. The software implementation has a low conversion rate, high power consumption and low physical protection[2]. Therefore, the hardware implementation is preferred which has the advantages of high speed, reliability and

*Corresponding author (E-mail: jayanthib1995@yahoo.com)

security. This paper surveys the selection of appropriate hardware, the commonly used neural networks and various optimization techniques to implement image applications in FPGA.

2 Materials and Methods

2.1 Hardware Selection

The first and foremost step in implementing ANN architecture is the selection of appropriate hardware based on the applications. ANN applications are capable of handling large number of data sets and also solve complex problems. The size and complexity of the system used for ANN applications and the requirements of resources are very high. The system also demands parallel processing of the data. In[3], the most commonly used hardware for this purpose are explained, some of which are CPUs, GPUs, Servers, Clusters, FPGA and ASICs.

CPU: It has a low computational efficiency and cannot meet requirements of a real time application. Hence, it was not suited for parallel processing.

GPU: It does parallel processing, but it consumes a lot of power. Hence, GPU was also not suitable for on field implementation.

ASICs: ASICs are power efficient and can be used for implementing ANN architectures. But their major disadvantage is that they are specific to a particular application and are not reusable for different architectures. The development cycle of ASICs is also large, hence it was not desirable to use for implementing ANN architecture.

FPGA: FPGAs are known for their reconfigurable structure[4]. They provide a possibility to reprogram the network design many numbers of times. They are low power consuming, enable parallel processing, have a design flexibility as shown in paper[5], and the development life cycle for FPGA is small as given in paper[6].

GPU is a multi-core processor[7] that is used for the interference and training of the ANN architecture. Since it does multi-million operations per second, it was suited for training and validation of neural networks. GPU consumes a lot of power, it was bulky in size[8], and cannot be suited for low power embedded systems. GPU supports only limited data types that are either single or double precision floating point. It cannot handle other types like 8/16 bit fixed point or 1-2 bit binary data.

Based on the analysis of various hardware for ANN architecture implementation, it was concluded that FPGAs were suitable for implementing ANN architectures in hardware, most commonly used ANN architectures for image applications are discussed next.

2.2 ANN Architectures for Image Applications

Deep Learning uses large datasets to recognize the pattern within the input image and produces classification within the image. Most commonly used ANN architectures for image analysis is CNN and RNN[9]. The challenge in deep learning for image classification is the time taken to train the ANN. This drawback can be overcome by the use of CNN for image classification and CNN is one of the most frequently used architecture for image classification[4], CNN is a multilayer network for image classification, segmentation and object detection[4], whereas RNN is used to analyze the sequential input. CNN reduces the image into its key features and identifies the feature using the combined probabilities. RNN evaluates the section of an input speech signal or video signal and then analyses the other sections one by one. CNN is used for medical image processing, face detection whereas RNN is used for image description, video tagging and video analysis.

Figure 1 shows the basic CNN Architecture, it consists of convolution layers, pooling layers and fully connected layers. There is much architecture in CNN. LeNet-5 is the basic CNN architecture and ResNet-50 and Dense Net is the most recent architecture. The number of layers may with the application. It is essential to optimize the implementations of neural networks to achieve better performance in terms of power, area and execution time. The review of the optimization techniques for implementing the neural networks are provided next.

3 Results and Discussion

3.1 Optimization techniques

The optimizations may be either hardware or software, they were mainly used to get better values of the design metrics. The metrics needed to determine the performance of the FPGA vary with

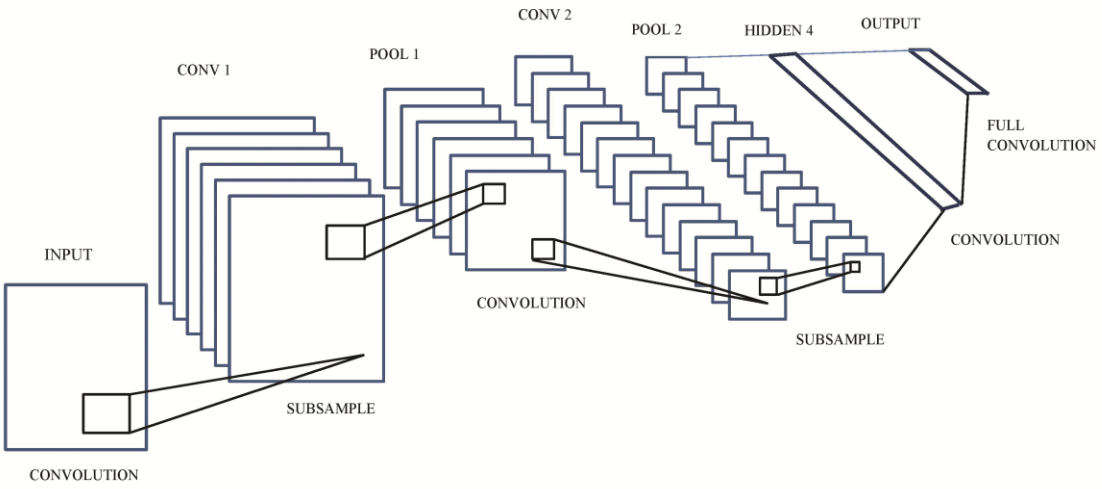

Fig. 1 — Block diagram of CNN architecture[9].

the applications[3]. The standard metric, used to find whether hardware optimization was done or not, was the Giga-Bit Operation per second (GOPS). The remaining metrics used for identifying hardware optimization were GOPS/W, GOPS/Slice, GOPS/DSP, speedup, accuracy, energy consumption and resources/area, and execution time.

3.2 Hardware optimizations

Hardware optimizations were done either by optimizing the precision of data or by choosing the appropriate resources or based on the memory used.

3.2.1 Number system optimizations

The precision of data was changed either to floating point or to fixed point for saving the area of utilization in the FPGA[10]. Changing the numeric representation or reducing the arithmetic precision[3], helped to reduce the off chip memory bandwidth demand, reduced the on chip memory requirement, and reduces the complexity of the arithmetic units and thereby reduces the power consumption of the system.

In, FPGA based implementation of a deep neural network for the handwritten digit recognition and the needed recognition problems was shown, which needs millions of weights and arithmetic operations for producing one output[11]. In order to use only on-chip memory for weight storage, the weights are represented in 3 bits, while the internal signals employ the precision of 8 bits.

Table 1 — Resource utilization for digit recognition with different weight precision[11]

Resource	FF	LUT	BRAM	DSP
3-bit without DSP	130237	124862	323	0
3-bit with DSP	130802	121173	323	900
8-bit fixed point	136677	213593	750.5	900

FF: Flip Flop; LUT: Look up table; BRAM: Block Random Access Memory; DSP: Digital signal processor.

A retrain based fixed-point weight optimization technique was employed to reduce the performance gap with floating-point algorithms. The FPGA resource utilization for Xilinx ZC706 evaluation board is shown in Table I, where the designs with and without DSP slices were compared. Using DSP slices does not reduce the demand of FFs (flip-flops) and LUTs much. A comparison of resource utilization for the implementation with 8-bit fixed-point weights and 3-bit representation was given. It was identified that 8-bit representation consumes almost all of LUTs and entire DSP slices on the FPGA chip because 8-bit weights need hardware multipliers for the implementation. The 8-bit representation for weights cannot be implemented on this FPGA because the BRAM capacity was not sufficient.

3.2.2 Resource optimization

In resource optimization, main importance was given for the Multiply and Accumulate (MAC). Multiplication is the main operation in ANN and these MAC units are replaced by fast and lower resource consuming multiplication units. It was

also replaced by a Wallace tree multiplier[5]. The resources were further reduced by replacing the MAC units with a Modified BOOTH Encoding (MBE) Multiplier and Wallace tree Based Adders[12].

The optimization was made further effectively by using a FPGA based coprocessor design. The use of FPGA based coprocessor design makes the computer and the coprocessor to communicate with each other through bus architecture. In[6], training of the CNN was done by using a coprocessor to configure for new network structure without changing the design but changing only the contents in the block memory.

Another method was the hardware and software co-design in which the SoC module supported the parameter reconfiguration. In[8], several network mapping methods were proposed to reduce the area and improve the efficiency of the system. The accuracy of the system proposed in[8] is 98% which occupied less than 60% of the FPGA resources. The processing time was reduced to 1.128s. It was concluded that this method has improved the system performance and use lesser resources with a lesser delay.

The other commonly implemented optimizations are parallelism and data multiplexing[3]. There are three ways in which parallelism can be implemented in the CNN architecture. They are parallelism between different channels, parallelism between different kernels, and using the parallel convolution kernel[4].

Data multiplexing is divided into intra-layer and inter-layer multiplexing. Intra-layer multiplexing the architecture is computed layer by layer this increases the data multiplexing and parallelism at the particular layer. Inter-layer multiplexing reduces the off chip access of data. This can be done by storing the intermediate results in the off chip storage blocks. By using the pipeline architecture, the execution time and memory access time can be reduced. This is because the data in the pipeline is used instead of getting the data from the off chip memory every-time the data is needed[4].

3.2.3 Memory Optimizations

Even though FPGA is best suited for implementing ANN in hardware, the major disadvantage is the memory constraint. The internal memory of FPGA is very low. Even the latest FPGA has a memory of 30 MB but the CNN processing needs 100s of MBs, hence an external DDR was used. So, the optimization of memory[8] plays a major role in affecting the number of resources, processing speed and efficiency of the output. One more parameter was the power consumption of FPGA. The reasons for power consumption were due to the complex mathematical computations. Power could be reduced by reducing the number of operations.

The latency of the processing was increasing due to the frequent access of data from the DDR. The memory reduction process is broadly divided into on chip optimization and off chip optimizations. On-chip optimization includes various data reuse, loop optimization and data arrangement techniques. Off-chip optimization included weight compression, pruning and quantization techniques. These techniques were used to reduce the number of weights mostly during training.

On-chip Optimization – The primary information such as input image and all kernels were stored in external memory which must be loaded entirely into FPGA for every test. But, results computed in intermediate stages (the values calculated in the adjacent layers) were stored in on-chip buffer. This reduced the data access to external memory significantly. Thus, the data stored in on-chip buffers were used as the input for the next layers. Commonly 2 ping-pong buffers were used for each stage, where the former layer may write to or read from one buffer while the next layer reads data from the other buffer. Also, dual-port memory driver and memory blocks were used to transfer data like ping-pong to the buffers. Data reuse, loop optimization techniques, batch based computations and buffer bank are some of the computational techniques which were proposed to reduce the shortage of on-chip memory resources. They are explained next[14].

Figure 2 explains how the accelerator was designed for CNN architecture. The image was given as the input and the weights were fetched from the external memory and fed to different layers. The operations of different layers were done and the intermediate results were propagated to the adjacent layers and then the output was obtained after the fully connected layer. This is the overview of the hardware implementation of the CNN architecture as explained in[14].

Data reuse: This technique is mostly used for the intermediate data from convolution to pooling

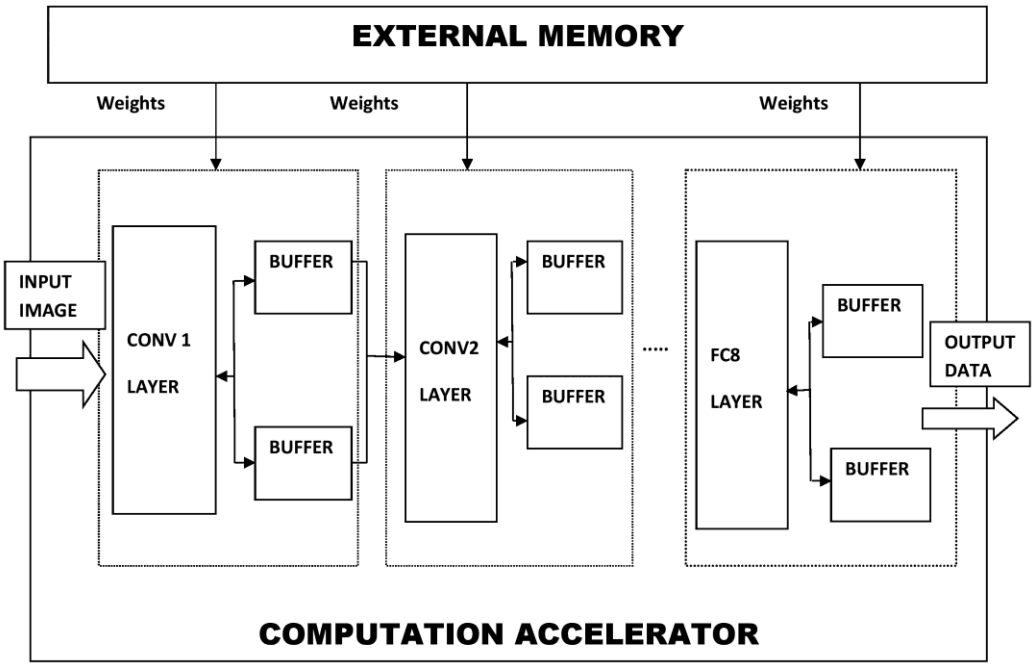

Fig. 2 — An overview of the accelerator architecture[13].

layer. Two types of data reuse methods were discussed in[15] such as spatial reuse and temporal reuse. In spatial reuse, in one clock cycle, an image pixel or a kernel read from on-chip buffers was used for multiple parallel computation units, and in temporal reuse, for multiple consecutive clock cycles, an image pixel or a kernel read from on-chip buffers was used. Also a novel data storage and data reuse scheme was discussed in[16], where all intermediate data was stored in BRAMs. Based on their design, memory usage was reduced more than 14 times for VGG16 implementation.

Loop optimization technique: In[15] the authors explored various loop optimization techniques to reduce memory consumption as well as to reduce the data movement. They have maximized the resource utilization in this way to achieve high performance. Loop Tiling and loop unrolling are the two main loop optimization techniques. In Loop tiling, the FPGA on-chip memory is limited to store all intermediate and input data. Therefore, it was necessary to use external DRAMs to store the weights and a portion of intermediate pixel values from CNN layers[16]. Loop tiling was proposed in[15] to divide data into multiple blocks to be fit in on-chip buffers. The loop tiling defined the lower bound on the required on-chip buffer size. In loop unrolling, unrolling along different loop dimensions will generate different implementation variants. Whether and to what extent two unrolled execution instances share data will affect the complexity of generated hardware, and eventually affect the number of unrolled copies and the hardware operation frequency. A step-by-step example of how to unroll convolutions to matrix multiplication and its architecture are shown in[16].

Batch-based computing: A batch-based computing technique can be used for Fully Connected layers (FC). Multiple input feature vectors were computed as a batch in parallel. The batch size was the number of input feature vectors. In[13], it was shown that, the numbers of Arithmetic Operations (AOs) were increased by batch size within the same memory access. AO included both multiplication and addition operations. Therefore, the memory bandwidth was reduced. In order to keep the maximum throughput, batch size should be more than the number of clock cycles required to read weights. Although this technique required larger on chip memory.

Buffer bank: Data buffering techniques can be used to hide the memory access latency and interconnected, so that the computation time could overlap with the data transfer overhead from the device DRAM to FPGAs BRAM. Two DMAs

usually used to establish communication between input/output buffers and DRAM to read/write simultaneously[15]. The input buffer could be split to data buffer and weight buffer to be filled one by one which was technically called ping-pong buffers.

Off-chip Optimization - Off-chip techniques were to balance the computation through put and memory bandwidth to keep the peak performance. Data compression, data arrangement and data quantization were generally utilized techniques to reduce and optimize data size and its arrangement on memories.

Data compression: The efficiency of Neural network model could be improved by reducing memory access through compression[17,18]. The most common deep compression techniques were pruning, trained quantization and variable encoding that reduced the memory requirement without affecting the system accuracy[17]. All general steps of data compression are shown as block diagram in Fig. 2. The pruning method removed redundant connections by learning the connectivity using standard network training. Then, all the connections with weights below a threshold (absolute value) were removed from the network. Finally, the network was re-trained to learn the final weights for the remaining sparse connections. This technique reduced the number of parameters by 9x and 13x for AlexNet and VGG-16 model[17], respectively.

The second block in Fig. 3 was meant for quantization and weight sharing. In this process, the weights from pruning were quantized, and multiple connections shared the same weight. Therefore, only the effective weights and the indices of the weights need to be stored and each parameter could be represented using very less number of bits[17]. After the first two stages, all the weights were represented with a fixed number of the bit width. Variable encoding could reduce memory usage by further compressing the model since the weights distribution was non-uniform. In this method, fewer bits were used to represent the more frequently appearing weights, and more bits were used to represent the less frequently appearing weights[17]. This technique was offline since Huffman coding did not require training. Table II shows various neural networks and their accuracy comparisons. It was clear from Table II that compression could save storage significantly with no loss of accuracy[17].

Data arrangement: The performance of the accelerator was based on the arrangement of data inside the DDR and BRAM. There are two different methodologies for arranging data for all Convolution and FC layers. For the Convolution layer data arrangement, the tiles/data which was used consecutively were stored at nearby memory locations to increase burst size.

Data Quantization: In paper[19], they have used 8-bit data, and 4-bit weight quantization techniques to reduce required memory and only 0.4% loss of accuracy was reported as compared to 32-bit full precision implementations. The networks could be 2-bit network or 1 bit network. Some of the software

Table 2 — Networks and their accuracy comparison[17]

Network	Original		Compressed		Ratio
	Size	Accuracy	Size	Accuracy	
AlexNet	240MB	80.4%	6.9MB	80.3%	35X
VGGNet	550MB	88.7%	11.3MB	89.1%	49X
GoogleNet	28MB	88.9%	2.8MB	88.9%	10X

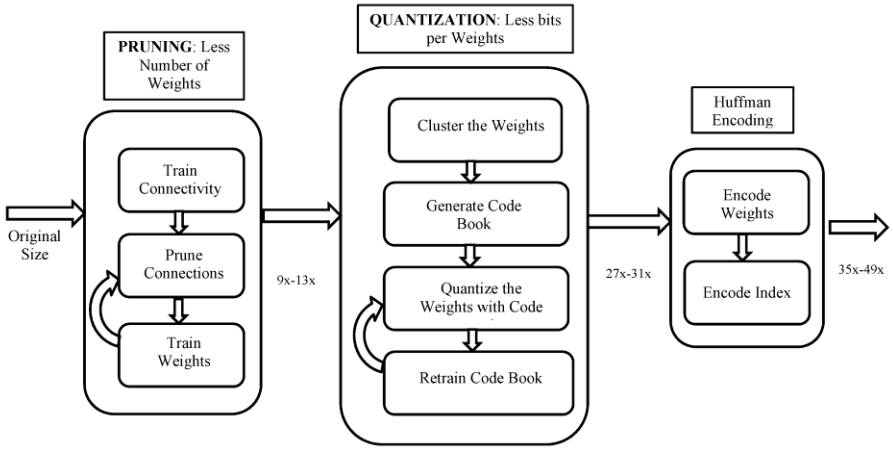

Fig. 3 — The three-stage compression pipeline: pruning, quantization and Huffman coding[18].

optimization were analyzed in[3], and they are explained next.

3.4 Software Optimizations

As CNN has a number chained loops, loop un-rolling was used to speed up the process and in turn reduce the execution time. Pruning is a method of disconnecting the neurons from the layers. This could reduce the number weight that has to be stored in the main memory. NSGAII (Estimator Distribution Algorithm) is the best network to be implemented in the hardware. It is possible to determine the network that can be implemented with more efficiency by using NSGAII. The other software optimization techniques generally used were interactive stencil loops, residual learning and matrix multiplication. Coppersmith and Winograd multiplication algorithm was implemented to improve the way of implementing the matrix multiplication and gives good results[3].

Software optimizations are made by integrating two layers in the CNN Architecture. In[20], the convolution layer and the batch normalization layers were integrated together. The convolution kernel was binarized and merged with the batch normalization layer. This design was tested in an object localizations system that was run on MATLAB. To solve the object localization task, the error was 1.6% and the accuracy decreased from 74 to 67% which is also a tolerable value.

4 Conclusion

This paper has compared the available hardware and software implementation methods of ANN architecture for image processing applications. The advantages of using FPGA for hardware implementations are that FPGA is effective in terms of power and re-configurability. It has been concluded from the survey that FPGA is best suited for hardware implementation. It has been found from the literature that most commonly used ANN architectures for image analysis were CNN and RNN. CNN architecture is being recommended for image classification as it requires less supervision and fewer hyper parameters. Even though FPGAs are the best for hardware implementation, they have resource and memory constraints. In order to overcome these limitations, some of the hardware and software optimization techniques are reviewed. These methods give an insight of optimizing the hardware and software based on the specific applications.

References

1 Tikhonov E E, Chebanov K A, & Burlyaeva V A, *2019 International Multi-Conference on Industrial Engineering and Modern Technologies(IEEE, Vladivostok, Russia)*, ISBN:978-1-7281-0061-6,2019, p.1.

2 Hu H, Huang J, Xing J, & Wang W, *2008 Second InternationalSymposium on Intelligent Information Technology Application(IEEE, Shanghai, China)*, I SBN: 978-0-7695-3497-8,2008, p. 259.

3 Dias M A, & Ferreira D A P, *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IEEE, Rio de Janeiro, Brazil)*, ISBN: 978-1-7281-3510-6, 2019, p. 95.

4 Ding R, Tian X, Bai G, Su G, & Wu, X Ru, *2019 IEEE 13th International Conference on ASIC (IEEE,Chongqing, China)*, ISBN:978-1-7281-0735-6, 2019, p.1.

5 Farrukh FUD, Xie T, Zhang C, & Wang Z, *2018 IEEE Conference on Integrated Circuits, technology and Applications(IEEE, Beijing, China)*, ISBN:978-1-5386-6551-0, 2018, p.88.

6 Clere S R, Sethumadhavan S, & Varghese K, *2018 21st Euromicro Conference on Digital System Design(IEEE, Prague, Czech Republic)*, ISBN:978-1-5386-7377-5, 2018, p.381.

7 Highlander T, & Rodriguez A, *2019 British Machine Vision Conference*,arXiv preprint arXiv:1601.0681,2016.

8 Zhang N, Shi H, Chen L, Lin T & Shao X, *2019 IEEE International Conference on Signal, Information and Data Processing(IEEE, Chongqing, China)*, ISBN: 978-1-7281-2345-5,2019, p.1.

9 https://missinglink.ai/guides/neural-network-concepts/cnn vs-rnn-neural-network-right.

10 Shah N, Chaudhari P & Varghese K, *IEEE Trans Neural Netw Learn Syst*, (2018) 5922.

11 Park J & Sung W, *2016 International Conference on Acoustics, Speech, and Signal Processing (IEEE, Shangai, China)*, ISBN: 978-1-4799-9988-0, 2016, p. 1011.

12 Farrukh FUD, Jiang Y, Zhang Z, Wang Z, Wang Z, & Jiang H, *IEEE Open J Circuits Syst*, 1 (2020) 76.

13 LiH , FanX, JiaoL, CaoW, Zhou W, & Wang L, *2016 26th International Conference on Field Programmable Logic and Applications(IEEE, Lausanne, Switzerland)*, ISBN:978-2-8399-1844-2, 2016, p. 1.

14 Shahshahani M, Goswami P, & Bhatia D, *2018 IEEE 13th Dallas Circuits and Systems Conference (IEEE, Dallas, TX, USA)*, ISBN:978-1-5386-9262-2, 2018, p. 1.

15 Ma Y, Cao Y, Vrudhula S & Seo J.-s., *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, p. 45.

16 Wang J, Lin J & Wang Z, *IEEE Trans Circuits Syst I Regul Pap IEEE T CIRCUITS-I*, 65 (2018) 6.

17 HanS, MaoH & Dally W L, *2016 International Conference on Learning Representations*, arXiv preprint arXiv: 1510.00149, 2016.

18 Chen Y, Krishna T, Emer J S, & Sze V, *IEEE J Solid-State Circuits*, 52 (2017) 127.

19 Qiu J, Wang J, Yao S, Guo K, Li B, Zhou E, Yu J, Tang T, Xu N, Song S, Wang Y, & Yang H, *2016 International Symposium on Field-Programmable Gate Arrays*, 2016, p.26.

20 Sledevic T, *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)(Vilnius, Lithuania)*, ISBN: 978-1-7281-2499-5, 2019, p.1.