Accelerated Low Power AI System for Indian Sign Language Recognition

Jella Sandhya
Electronics & Communication Engineering
University College of Engineering Science & Technology Hyderabad (JNTUHCEST)
Hyderabad, Telangana
sanju.436@gmail.com

Dr. Anitha Sheela Kancharla
Electronics & Communication Engineering
University College of Engineering Science & Technology Hyderabad (JNTUHCEST)
Hyderabad, Telangana
kanithasheela@jntuh.ac.in[2]

***Abstract*: Deep Convolutional Neural Network (CNN) based methods have become more powerful for wide variety of applications particularly in Natural Language Processing and Computer vision. Nevertheless, the CNN-based methods are computational expensive and more resource-hungry, and hence are becoming difficult to implement on battery operated devices like smart phones, AR/VR glasses, Autonomous Robots etc. Also with the increasing complexity of deep learning models like ResNet-50, there is a growing demand for efficient hardware accelerators to handle the computational workload. In this paper, we present the design and implementation of a neural network accelerator tailored for ResNet-50 on the ZCU102 platform using Field-Programmable Gate Arrays (FPGAs) which offers and customizable solution to address this challenge. We systematically investigate the design choices and optimization strategies for deploying custom built ResNet-50 network trained for Indian Sign language translation of 76 gestures enacted and build in our labs for Doctor patient interface on FPGA-based accelerators. In order to enhance operational speed, we have employed various techniques, including parallelism and pipelining, leveraging Depthwise Separable Convolution. Furthermore, we have implemented hierarchical memory allocation for different offsets using threads. Additionally, we have utilized weight and data quantization to optimize operational speed while minimizing resource consumption, thus achieving low power consumption while maintaining acceptable levels of inference accuracy. We, evaluated our accelerated FPGA model against CPU interms of various performance metrics viz: frames per second (fps), Memory allocations, LUTs, DSPs and Block RAMs used. Our findings underscore the superiority of FPGA-based accelerators, as evidenced by achieving a frame rate of 2.7fps on the Xilinx Ultra Scale ZCU102 platform with int8 quantization, compared to 0.8fps for Single precision. In contrast, the CPU achieved a frame rate of 0.6fps. Notably, we observed a minimal accuracy variation of only 1.37% with int8 quantization, while no accuracy variation was observed for Single precision. Our implementation utilized 16 convolution threads and 4 FC threads operating at 200 MHz for single precision, whereas for int8, we employed 25 convolution threads and 16 FC threads operating at 250 MHz.**

Keywords: FPGA accelerator, Data quantization, optimization, Xilinx Ultra Scale ZCU102

I. INTRODUCTION:

Video and image classification is basic problem in computer vision. CNNs are led to great advances in image classification accurately [1] Russakovsky et.al. Accuracies are improving on ImageNet day by day with advances in CNN based networks as compared to traditional methods. Even though they achieve state-of-the-art performance, CNN-based methods require significantly more computations and memory resources. Due of this, the majority of CNN-based implementations require High end servers.

CNN models, like the 5-layer LeNet, are suitable for basic tasks like MNIST handwritten text recognition [18]. Because of their extreme complexity, state-of-the-art CNN models for large-scale image classification can only be kept in external memory. In this way, memory and bandwidth especially for embedded system becomes a significant barrier to CNN acceleration.

To address this issuewe focus on leveraging Field-Programmable Gate Arrays (FPGAs) to accelerate deep learning inference tasks [2, 3, 4, 5, 6, 7,8,9], specifically targeting the deployment of a quantized ResNet-50 neural network model trained on our custom built Indian Sign language database for Doctor patient interface [10] on the Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit.

We present the flow for single and int8precision data quantization. Results show that only a 1.37% accuracy loss is introduced with ResNet-50 model with 8 -bit weight and bias quantization as compared to using Single precision.

We begin by initializing a pre-trained ResNet-50 network and then quantizing [11] it using a deep learning quantizer with FPGA execution environment specifications. This step ensures that the network is optimized for deployment on FPGA hardware while maintaining acceptable inference accuracy. Subsequently, we calibrate the quantizer using ISL dataset to fine-tune the quantization parameters for optimal performance.

To facilitate deployment on the ZCU102 platform, we configure the processor settings, specifying the bitstream and synthesis tool, essential for FPGA synthesis. To further optimize the hardware configuration for real-world deployment we employ optimization techniques tailored to the specific characteristics of the quantized ResNet-50 model. This optimization process aims at finding the optimum threads to be used both in Convolution and FC layer for performing parallel depth wise convolution operations to meet the throughput requirements in terms of frames per second(fps), ensuring efficient utilization of FPGA resources viz LUTs, DSPs, Block RAMS.

Finally, we utilize Xilinx Vivado as the synthesis tool and configure the hardware target interface for JTAG communication. Leveraging these tools and techniques, we build the optimized processor configuration, ready for deployment on the ZCU102 FPGA platform. We could achievean accuracy of 84.07% for ZCU102 with single

precisionwith a speed of 0.8 fps and an accuracy of 82.89% for ZCU102_int8 at 2.7fps.

II. BACKGROUND

A. A deep CNN model

A conventional Convolutional Neural Network (CNN) architecture comprises a cascade of filter channels, executed sequentially. The model's parameters, commonly referred to as "weights," are optimized during the training process. The initial layer of the CNN processes an input image and produces a set of global feature maps. Subsequent layers analyze these feature maps from preceding layers, generating more localized feature maps. Finally, a classifier layer, typically a Fully Connected (FC) layer coupled with a SoftMax layer, yields the probabilities associated with each category that the input image potentially belongs to. The CONV layer and FC layer represent two fundamental types of layers within the CNN framework. Following CONV layers, pooling layers are commonly incorporated to perform down-sampling of the feature maps, thereby reducing their dimensionality and mitigating sparsity. A typical CNN example is shown in Figure 1.

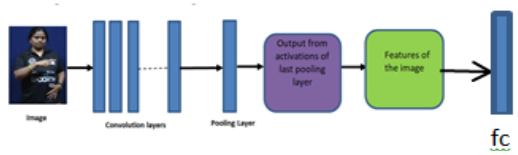


Fig. 1. Basic CNN structure from feature map

B. FPGA-BASED ACCELERATORS:

FPGA-based neural network accelerators are increasingly preferred over CPUs due to their superior efficiency [12]. FPGAs exploit parallelism to accelerate computations by mapping them onto parallel hardware, allowing multiple DNN structures to execute concurrently on the FPGA. These accelerators can achieve speedups of several orders of magnitude compared to baseline CPUs [9]. FPGAs offer designers the flexibility to implement only the necessary logic in hardware tailored to the target application. Architectures of FPGA-based DNN accelerators typically comprise a host computer and an FPGA component responsible for executing DNN algorithms.

FPGA-based DNN accelerators [13], can be broadly categorized into two types: accelerators tailored for specific applications such as speech recognition, object detection, and natural language processing, and accelerators designed for specific algorithms such as CNN and RNN. Additionally, there exist accelerator frameworks equipped with hardware templates. The design complexity of accelerators for the first two categories is relatively low. In this study, we utilize an accelerator for for Sign language recognition using pre-trained ResNet50 network and transfer learning.

III. DATASET

For the creation of an Indian Sign Language (ISL) recognition model, a dataset is constructed within the anechoic chamber, funded by UGC-MRP, situated in the ECE Department at JNTUHUCESTH. This dataset encompasses 76 unique ISL signs associated with doctor-patient interaction words and phrases. Each sign is depicted through 50 individual gesture videos performed by 10 individuals. Consequently, a total of 3800 videos have been collected and utilized to enable comprehensive training for optimal model performance.

No. of persons: 10

No. of videos per person: 5

Video details: frame width and height 1920×1080

Duration: 2 to 4sec, frame rate: 50 fps

IV. SYSTEM DESIGN

The image classification model ResNet50 will extract features from each frame of the ISL (Indian Sign Language) video for predicting the video labels. Video is converted into frames and mediapipe python framework is applied to extract knuckle points of the hands. These images are given to ResNet 50 pre trained model for training 76 gestures. The trained model is deployed onto ZCU102 board and tested for accuracies of each gesture of ISL.

Figure 2(a) illustrates the system architecture specifically tailored for the ZCU102 platform. Given the non-utilization of the Heterogeneous Processing System (HPS) in this particular design, only the FPGA segment is depicted. The DDR4 memory module is directly linked to the FPGA portion. Operating at a clock frequency of 250MHz, the CNN accelerator(ResNet50) encounters frequency constraints primarily imposed by its adder tree configuration. To facilitate the seamless transfer of weights and input images from flash memory to DDR4 external memory, a Vitis AI softcore microprocessor is instantiated. The integration of an external memory interface IP alongside a Modular Scatter-Gather Direct Memory Access (mSG-DMA) IP is employed to effectively bridge the buffers in CNN accelerator [14] and the FPGA memory.

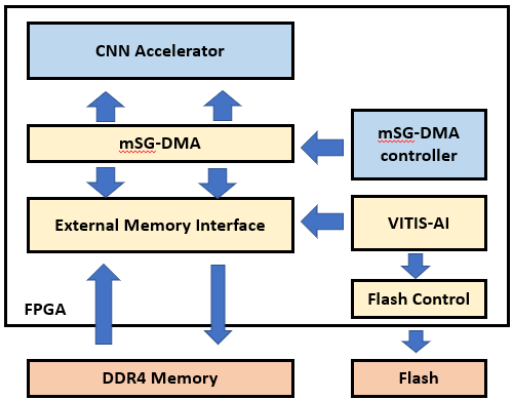


Fig. 2. (a) FPGA System Architecture

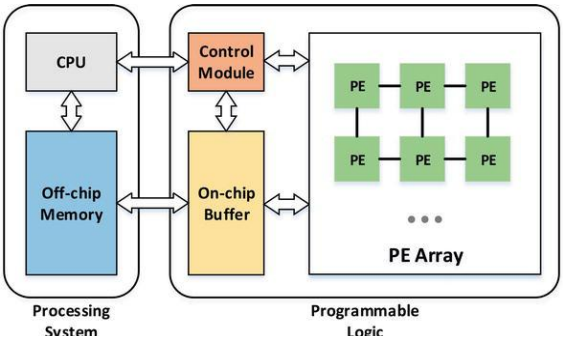


Fig2 (b) Architecture of CNN accelerator based on embedded FPGA platform

Fig 2(b) shows how acceleration on FPGA is performed. It is done under two heads:

A. Data Quantization:

Accelerating large CNN models on embedded FPGA platforms necessitates data quantization, a critical aspect where achieving a concise representation with minimal loss in accuracy is paramount. Utilizing fixed-point arithmetic units on FPGAs proves advantageous due to their lower resource requirements compared to floating-point implementations. Contemporary CNN accelerators leverage fixed-point arithmetic [3, 15, 16, 2] for enhanced acceleration while maintaining inference accuracy. Additionally, reducing precision from 32-bit to 8-bit for both data and weights yields significant reductions in memory footprint and computational resources.

For instance, Chen et al. demonstrated that the area and power consumption of a 16-bit multiplier are 0.164× and 0.136× respectively, compared to a 32-bit multiplier, using 65nm fabrication technology. Previous studies predominantly adopted the 16-bit quantization strategy [17, 15, 3, 4]. In [3], Chen et al. observed that employing 16-bit numbers instead of 32-bit ones resulted in a mere 0.26% increase in error rate on the MNIST dataset. Similarly, in [4], 16-bit numbers were employed during inference while 32-bit numbers were used during training, showcasing a 0.01% reduction in accuracy on the MNIST dataset.

With this understanding in our work, we have worked withfloating single (16-bit) and fixed int8 for weight and bias representation during inferenceto reduce computational complexity and compared the resultant recognition accuracies. However, while training we have used 64 bit precision on GPU based computer since training will be done offline and hence there is less constraint on speed or computational resources.

B. Optimization of processor configuration:

The total number of multiplications to be used for performing Standard Convolution operation is $\boldsymbol{C \times M_k^2 \times M_D^2 \times L}$,where$M_k \times M_k$ is image size, C is number of channels, and output size will be $M_p \times M_p$.

In our FPGA processing we have used depth wise separable convolution.

Depth wise separable convolution is performed using two processes:

1. Depth wise convolutions
2. Point wise convolutions

Depth wise convolution: Convolution will be applied to single channel unlike standard convolution.So filter or kernel size will be of size $M_k \times M_k \times 1$. If there are C channels in input data then C such filters are required and hence the output will be of size$M_D \times M_D \times C$. For depth wise convolution total no. of multiplications required are $C \times M_k^2 \times M_p^2$

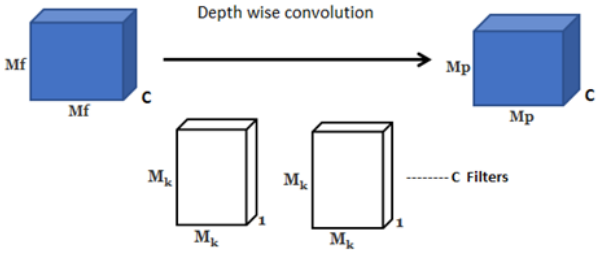


Fig. 3. Depthwise convolution operation

Point wise convolution: 1×1 convolution operation is applied on C channels. Size of the filter is 1×1× C. Using L such filters the output becomes $M_D \times M_D \times L$. Total number of multiplications required for point wise convolution is $C \times M_p^2 \times L$

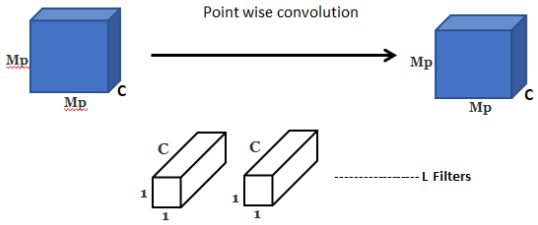


Fig. 4. Point wise convolution operation

Hence the total numberof multiplicationsrequired for Depth wise separable convolution = depth wise +pointwise multiplications

$$= C \times M_k^2 \times M_p^2 + C \times M_p^2 \times L = C \times M_p^2 (M_k^2 + L)$$

From above equations compared to standard convolution depth wise convolution operations are less by a ratio of (1/L) + $(1/M_k^2)$.

The number of LUTS, BRAMS and DSPs required will be increasing very heavily which in turn will increase the power consumption if the convolution is applied on a single channel as mentioned abovein depth wise convolution. Hence to have a tradeoff between the speed of operations and power consumption we have proposed to perform the depth wise separable Convolutions using multiple threads so that a bunch of convolutions will be performed serially and remaining convolutions are done in parallel.

V. IMPLEMENTATION DETAILS

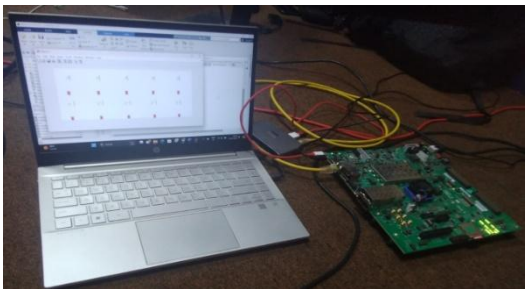


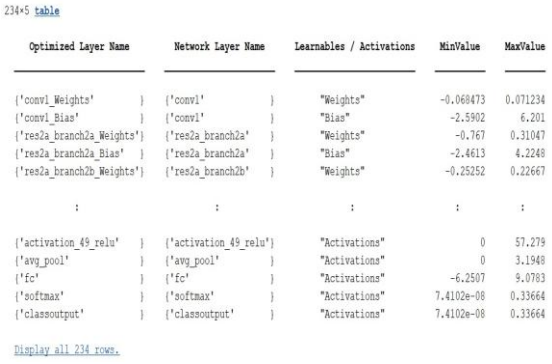
Fig. 5. . Zcu102 Development board image

A. Quantization of Deep learning Network:

Quantization is a process to represent a continuous value (floating point) expressed as a real number as an integer multiple, which is a quantum of the smaller unit.

This technique is the most basic and easy way to use trained models to make inferences quickly, reducing operation costs, reduce calculation loads and reduce memory consumption. However, the accuracies may suffer if the quantization step size is very big.We have used int8 and single precision for both computations and memory accesses with lower precision data and compared the performance.

Fig5shows the minimum and maximum values used by the network for weights, biases and activations while using int8.



234×5 table

Optimized Layer Name	Network Layer Name	Learnables / Activations	MinValue	MaxValue
{'conv1_Weights'}	{'conv1'}	"Weights"	-0.068473	0.071234
{'conv1_Bias'}	{'conv1'}	"Bias"	-2.5902	6.201
{'res2a_branch2a_Weights'}	{'res2a_branch2a'}	"Weights"	-0.767	0.31047
{'res2a_branch2a_Bias'}	{'res2a_branch2a'}	"Bias"	-2.4613	4.2248
{'res2a_branch2b_Weights'}	{'res2a_branch2b'}	"Weights"	-0.25252	0.22667
:	:	:	:	:
{'activation_49_relu'}	{'activation_49_relu'}	"Activations"	0	57.279
{'avg_pool'}	{'avg_pool'}	"Activations"	0	3.1948
{'fc'}	{'fc'}	"Activations"	-6.2507	9.0783
{'softmax'}	{'softmax'}	"Activations"	7.4102e-08	0.33664
{'classoutput'}	{'classoutput'}	"Activations"	7.4102e-08	0.33664

Display all 234 rows.

Fig. 6. Network quantization weights for int8

B. Deep learning Network optimization for acceleration and low power

After quantizing the network with different quantization, it is optimized for increasing the throughput by changing the Convolution and FC layer threads. dlprocessor optimization tool of Matlab : Deep learning toolbox is used to estimate the resource allocation for a given fps.

C. Bit stream generation, deployment and verification:

After optimizing the processor configuration for deep learning network the following steps are performed for generating the bitstream:

1. Loading the optimized Deep learning processor model and generation of HDL code generation for the model that is the testbench/Device under test(DUT)
2. Compilation of the model ‘testbench’
3. Generating HDL for ‘testbench/DUT’ and running HDL checks. Model complilation and PIR creation will be completed.
4. Generating RTL code and IP Core
5. Applying HDL optimization on the model ‘testbench’
6. LUTMapToRAM is set to ‘on’ to map lookup tables to block RAM in hardware
7. Create Project and build FPGA Bitstream.
8. Once the bitstream is generated the network is deployed on to the FPGA with weights, biases being loaded using JTAG.
9. The deployed network is tested with various inputs and accuracies are noted.

VI. SYSTEM EVALUATION

The following Tables 1and 2 shows the resource utilization of Deep learning Processor for both Single and int8 quantization.

TABLE I. ZCU102_SINGLE DEEP LEARNING PROCESSOR RESOURCE UTILIZATION

	DSPs	**BlockRAMLUTs(*CLB/ALUT*)**	
Available	2520	912	274080
Total	390(16%)	586(65%)	249893(92%)
Ref.Design3(1%)		78(9%)	35000(13%)
DLProcessor387(16%)		508(56%)	214893(79%)

TABLE II. ZCU102_INT8 DEEP LEARNING PROCESSOR RESOURCE UTILIZATION

	DSPs	**BlockRAMLUTs(*CLB/ALUT*)**	
Available	2520	912	274080
Total	502(20%)	536(59%)	190099 (70%)
Ref.Design3(1%)		78 (9%)	3500 (13%)
DLProcessor499(20%)		458(51%)	155099(57%)

A. Simulation results after training Resnet50 with custom made ISL database:

Fig7.Shows the training progress curve in terms of Training and Validation accuracies/losses while training the Resnet 50 with ISL gestures. Table1 shows accuracies of the Deep learning network during simulation and after deployment on to hardware using int8 and Single precision. Resource utilization along with performance is given in Table2. for the same.

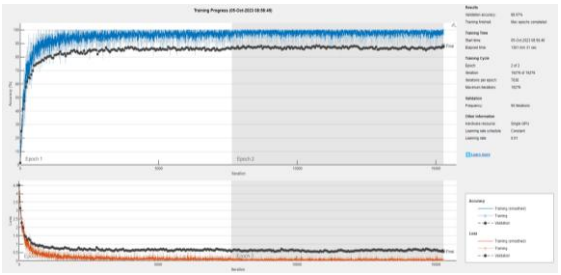


Fig. 7. Training and Validation Accuracy and loss graphs of Resnet50

TABLE III. COMPARISON OF ACCURACIES FOR SIMULATION AND HARDWARE IMPLEMENTATION USING INT8 AND SINGLE PRECISION

S.No.	***ISL GESTURE***	***Simulation Accuracy (RESNET 50) %***	***FPGA Accuracy ZCU102_single (RESNET 50) %***	***FPGA Accuracy ZCU102_int8 (RESNET 50) %***
1	0	100	100	100
2	1	100	100	100
3	2	100	100	90
4	3	100	100	90
5	4	100	100	100
6	5	100	100	90
7	6	100	100	100
8	7	100	100	70
9	8	100	100	100
10	9	100	100	100
11	A	100	100	100
12	Accident	50	50	50
13	All the best	100	100	100
14	Allergies	100	100	50

15	Asthma	100	100	90
16	B	70	70	100
17	Blood pressure	50	50	100
18	Breathe	100	100	100
19	Bye	100	100	100
20	C	60	60	60
21	Cancer	100	100	100
22	D	100	100	100
23	Diabetes	100	100	100
24	Doctor	90	90	90
25	E	100	100	100
26	Ecg	100	100	100
27	Emergency	60	60	80
28	Excuse me	100	100	90
29	F	100	100	100
30	Fever	60	60	100
31	G	90	90	90
32	Good afternoon	0	0	60
33	Good evening	100	100	100
34	Good morning	50	50	50
35	Good night	100	100	100
36	H	100	100	100
37	Headache	50	50	0
38	Health insurance	90	90	80
39	Heart attack	90	90	90
40	Hello	0	0	30
41	Hospital	30	30	70
42	How are u	100	100	60
43	I	0	0	0
44	I am fine	100	100	100
45	J	100	100	60
46	K	100	100	100
47	L	100	100	100
48	M	50	50	50
49	Medicine	80	80	60
50	My name is	100	100	100
51	N	100	100	70
52	Nice to meet u	60	60	50
53	No	50	50	80
54	O	100	100	100
55	Operation	100	100	100
56	P	60	60	50
57	Please	100	100	70
58	Q	0	0	0
59	R	100	100	70
60	S	100	100	100
61	Sorry	100	100	100
62	Stomach ache	100	100	100
63	T	100	100	20
64	Thank you	100	100	100
65	Thermometer	100	100	100
66	U	100	100	100
67	V	50	50	100
68	Virus	100	100	100
69	Vomit	100	100	100
70	W	100	100	60
71	Welcome	100	100	100
72	What is ur name	0	0	100
73	X	100	100	100
74	Y	100	100	100
75	Yes	100	100	80
76	Z	100	100	100

TABLE IV. COMPARISON OF RESOURCE UTILIZATION AND PERFORMANCE INTERMS OF ACCURACIES, FPS AND LATENCY

Parameters	Software testing	ZCU102_single	ZCU102_int8
Accuracy	84.07%	84.07%	82.89%
FPS	1 FPS	0.8 FPS	2.7 FPS
Latency in cycles			
LUT's	-	214893	155099
DSP	-	387	499
RAM	-	508	458

VII. CONCLUSION:

In this paper we have designed and deployed an optimized custom ResNet-50 network by using transfer learning for ISL recognition for doctor patient interface gestures on edge computing platform ZCU102. We have shown that FPGAs offer a highly flexible and customizable solution to address the increasing computational demands of complex neural networks while maintaining efficient resource utilization.

Techniques to improve performance such as parallelism, memory hierarchy, and pipelining have been leveraged to achieve significant improvements in throughput and latency compared to CPU implementations.

We could achieve a throughput of 2.7 fps as compared to 0.6fps on 12th Gen Intel(R) Core(TM) i9-12900K, 3200 Mhz processor with a difference in accuracy of only 1.37%. The resource utilizationis also as low as only 20% utilization of DSPs, 51% of BRAM and 57% of LUTs while running on Xilinx Zynq Ultra Scale+ MPSoc SOC ZCU 102 Evaluation Kit.

The developed model is deployable on embedded boards andfacilitates online and offline doctor-patient interactions, especially benefiting individuals with hearing impairments by overcoming communication barriers and enhancing accessibility to healthcare services.

VIII. FUTURE WORK

We intend to expand this research to encompass dynamic data gesture recognition tasks, focusing on interpreting video gestures. Our goal is to develop an optimized hardware accelerator with low power consumption tailored specifically for this application.

IX. DECLARATION:

A. Funding:

We acknowledge the support provided by Dr. Reddy Labs in providing us the training for performing the gestures for creating the ISL database. Additionally, we received funding from the UGC MRP scheme and AICTE RPS, which enabled us to establish the Anechoic Chamber, along with the GPU-based processing System and Audio video recording equipment.

B. Competing Interests:

No, I declare that the authors have no competing financial and/or non-financial interests as defined by **the journal**, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

REFERENCES:

[1] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115 (2015): 211-252.

[2] Zhang, Chen, et al. "Optimizing FPGA-based accelerator design for deep convolutional neural networks." Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays. 2015.

[3] Chen, Tianshi, et al. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning." ACM SIGARCH Computer Architecture News 42.1 (2014): 269-284.

[4] Chen, Yunji, et al. "Dadiannao: A machine-learning supercomputer." 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE, 2014.

[5] Liu, Daofu, et al. "Pudiannao: A polyvalent machine learning accelerator." ACM SIGARCH Computer Architecture News 43.1 (2015): 369-381..

[6] Du, Zidong, et al. "ShiDianNao: Shifting vision processing closer to the sensor." Proceedings of the 42nd annual international symposium on computer architecture. 2015.

[7] Chakradhar, Srimat, et al. "A dynamically configurable coprocessor for convolutional neural networks." Proceedings of the 37th annual international symposium on Computer architecture. 2010.

[8] Farabet, Clément, et al. "Neuflow: A runtime reconfigurable dataflow processor for vision." CVPR 2011 workshops. IEEE, 2011.

[9] Farabet, Clément, et al. "Cnp: An fpga-based processor for convolutional networks." 2009 International Conference on Field Programmable Logic and Applications. IEEE, 2009.

[10] Jella Sandhya, KANCHARLA ANITHASHEELA. Spatiotemporal Modeling for Dynamic Gesture Recognition in Video Streams, 08 March 2024, PREPRINT (Version 1)available atResearch Square [https://doi.org/10.21203/rs.3.rs-4019650/v1]

[11] Qiu, Jiantao, et al. "Going deeper with embedded FPGA platform for convolutional neural network." Proceedings of the 2016 ACM/SIGDA international symposium on field-programmable gate arrays. 2016

[12] Nurvitadhi, Eriko, et al. "Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC." 2016 International Conference on Field-Programmable Technology (FPT). IEEE, 2016.

[13] Dhilleswararao, Pudi, et al. "Efficient hardware architectures for accelerating deep neural networks: Survey." IEEE access 10 (2022): 131788-131828.

[14] Bai, Lin, Yiming Zhao, and Xinming Huang. "A CNN accelerator on FPGA using depthwise separable convolution." IEEE Transactions on Circuits and Systems II: Express Briefs 65.10 (2018): 1415-1419.

[15] Farabet, Clément, et al. "Large-scale FPGA-based convolutional networks." Scaling up machine learning: parallel and distributed approaches 13.3 (2011): 399-419.

[16] Sankaradas, Murugan, et al. "A massively parallel coprocessor for convolutional neural networks." 2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors. IEEE, 2009.

[17] Larkin, Daniel, Andrew Kinane, and Noel O'Connor. "Towards hardware acceleration of neuroevolution for multimedia processing applications on mobile devices." Neural Information Processing: 13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006. Proceedings, Part III 13. Springer Berlin Heidelberg, 2006.

[18] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324