Low Power Pose Estimation Accelerator for Multiple Scenarios

Ze Jia

***Abstract*—Accelerators for pose estimation have a wide range of application scenarios at the edge. However, the complexity and variability of the edge environment and the limited power consumption restrict the application and deployment of conventional devices such as GPUs at the edge. In response, this paper proposes a low-power bit-pose estimation accelerator architecture suitable for application in multiple scenarios, and this design supports the mapping of different pose estimation networks to accelerators. In addition, we also develop an accompanying operator library to support new pose estimation networks. In order to cope with the resource- and power-scarce scenarios at the edge, the number of operators in the library can be flexibly configured before accelerator deployment, and the power consumption can be dynamically adjusted by dynamically adjusting the operating frequency of the accelerators in real applications. Finally, we deployed the proposed architecture on Virtex UltraScale+ VU9P and tested it using PVNet as well as MobileNetV2. The experimental results show that the proposed architecture can achieve performance similar to that of current state-of-the-art dedicated accelerators with lower power consumption and greater versatility while ensuring real-time performance and accuracy.**

***Index Terms*—Pose Estimation, CNN, FPGA.**

I. Introduction

Artificial Intelligence is a long sought after goal of mankind, and CNN as a way to achieve this goal[1] has been continuously improved and developed in recent years. New CNN architectures have been proposed by academia and industry to cope with the ever complex application scenarios, and CNNs have replaced most of the traditional algorithms in image processing by virtue of their high accuracy, adaptability and scalability. Pose estimation, as a branch of image processing, has also been influenced by CNN, such as: assisted driving[2], motion recognition[3], augmented reality[4], human-computer interaction[5], and other widely used fields of pose estimation have begun to replace the original mathematical algorithms with CNN. However, the premise of CNN high-precision results is the corresponding computational complexity, in order to achieve high-precision fast processing is bound to pay the corresponding price. This cost includes extremely high power consumption, a large number of computing units, high-frequency memory access, and so on. This makes how to achieve high efficiency and low power consumption to allow the deployment of attitude estimation networks at the edge an important research topic.

In practical applications of pose estimation, it is crucial to have the ability to perform accurate operations in complex and changing environments[6]. The complexity of the environment has led to the creation of many algorithms for different scenarios to meet the challenges of the pose estimation work, such as PVNet[7], which is good at dealing with occlusion situations, and MobileNetV2[8], a lightweight network. In this context, choosing a suitable vehicle for the bit-pose estimation network accelerator is crucial. Depending on their architectures, typical AI chips are now divided into three main categories[9]. The first one is a general-purpose AI chip optimised by hardware and software, such as GPU (Graphics Processing Unit), represented by NVIDIA, which has become the first choice for artificial neural network training due to its powerful data parallel processing capability and the high parallel demand of artificial neural networks[10]. However, when it comes to reasoning, GPUs are often difficult to deploy effectively at the edge due to their high power consumption. The second category is fully customised artificial intelligence chips such as ASIC (Application-specific Integrated Circuit), representative vendors are Google and Cambricon. ASIC chips can be customised and optimised at the hardware level to meet the needs of specific applications, with small size, low power consumption, high performance and low cost. The disadvantage of ASIC chips is that they cannot be dynamically configured, and there is no room for change once they are flowed on the chip, lacking the versatility and flexibility of GPU and FPGA chips. The third category is semi-customised AI chips such as FPGAs (Field-programmable Gate Array), with representative vendors such as Xilinx and Altera, which are mainly suitable for low-latency streaming computation-intensive tasks. FPGA chips have low power consumption, programmability and reconfigurability, allowing users to customise their designs and implement the latest neural network models. Neural network models.

Current mainstream bit-posture estimation accelerators usually limit the acceleration target to a specific network, which is difficult to achieve ideal results in complex edge environments. In this paper, we propose a low-power bit-posture estimation accelerator for multi-scenarios, aiming to address the shortcomings of current mainstream bit-posture estimation accelerators in terms of single application scenarios and high power consumption. We choose PVNet and MobileNetV2 as the validation networks for the accelerator, with the former maintaining excellent performance when the detection target receives severe occlusion, and the latter being a lightweight network. To facilitate the deployment of the network, we also perform quantisation operations on the parameters, an approach that can significantly improve the inference speed while the

Manuscript received April 10, 2025

Ze Jia, School of computer science and technology, Tiangong University, Tianjin, China.

accuracy is only slightly affected. Finally, we validate on the LineMod dataset, where PVNet achieves 84.53% accuracy and MobileNetV2 achieves 78.42% accuracy. Our main contributions are as follows:

1. By inputting a sequence of instructions to the controller to control the accelerator to perform the corresponding operations on the data, this control mode can be used to dynamically reconfigure the accelerator's operating network to cope with different operating scenarios by changing the instructions written in real time without changing the underlying FPGA configuration file.

2. In the development session of the accelerator, in order to ensure hardware adaptability under different resource conditions. We use the SpinalHDL development language to enhance the flexible configuration of FPGA parameters, so that the user does not need to focus on the specific implementation details, and can change the types of operators in the arithmetic library according to the different networks being accelerated. This architecture allows the accelerator to minimise resource consumption to cope with resource constraints at the edge, and facilitates additional expansion of the arithmetic library at a later stage.

3. The work intensity of the accelerator is controlled by dynamically adjusting the operating frequency of the system. This mode allows the edge device to dynamically adjust the operating frequency according to the different real-time requirements in different scenarios to achieve the purpose of reducing power consumption.

The rest of the paper is organised as follows: section II describes the related work. Section III analyses the bit-position estimation network and its mapping at the accelerator Section IV describes the overall architecture of the system, the development of the operator library, and the optimisations performed on the edge side. Section V presents the hardware implementation and experimental results. Section VI concludes.

II. Related Work

A complete pose estimation process is roughly divided into four steps: feature extraction, keypoint detection, keypoint matching and correlation, and pose estimation. Early feature extraction algorithms usually relied on classical algorithms such as SIFT[11] (Scale Invariant Feature Transform), SURF[12] (Speeded-Up Robust Features), and HOG[13] (Histogram of Orientation Gradients). These methods capture the important information in an image by means of features that are statistically derived manually. In the era of deep learning, CNNs have greatly improved the efficiency and accuracy of feature extraction by virtue of their ability to automatically learn hierarchical features from images. Traditional keypoint detection methods include Harris corner detection[14], FAST[15] (Features from Accelerated Segment Test), and so on. These methods rely on local changes in the image to detect keypoints. CNN can directly output the location heat map of keypoints through network training, which greatly improves the accuracy and robustness of detection. The traditional keypoint matching work is generally relied on violent matching, KD tree[16] and other ways to complete, CNN's powerful learning ability is far more than the traditional algorithms in this work efficiency. Only in the position estimation after obtaining the key point features, such as PnP[17] (Perspective-n-Point) and ICP[18] (Iterative Closest Point), the traditional algorithms can be temporarily higher than the deep learning algorithms in terms of algorithm complexity and accuracy. Based on this status quo, the current pose estimation network generally only contains the work related to feature point extraction and matching, such as PVNet algorithm's object block diagram is derived from the PnP algorithm after the network derives the feature points. It is also due to this work characteristic that the networks of pose estimation algorithms generally do not have a fully connected layer as in the case of object classification networks. The absence of a fully connected layer consumes a lot of computational resources if deployed at the edge, and makes attitude estimation networks more suitable for deployment at the edge using accelerators.

A number of accelerators have been proposed for bit-position estimation, A. Sohrabizadeh et al[19]. constructed a set of bit-position estimation network with MobileNetV2 as the backbone on FPGA and optimised the system end-to-end. This strategy undoubtedly strengthens the precision and accuracy of the computation, but at the same time it inevitably increases the dependence of the edge system on high-performance servers and the increase of the power consumption of the whole system.Xiang Wang et al[20]. proposed a MobileNetV2+LightPose processing method based on FPGA. This strategy achieves high frame rate bit-pose estimation through extreme compression of the neural network, and at the same time, the compression of the neural network makes its AP only 0.546, which is difficult to meet the requirements of practical applications.Fan H et al[21]. proposed a MobileNetV2+SSDLite-based approach to reduce the number of network parameters and practiced with the ZC102 platform, and the design is indeed effective! This design does effectively reduce the computational pressure of the accelerator, but the partial quantisation still makes the performance of this scheme unsatisfactory in terms of frame rate. While several previous acceleration schemes are limited to accelerating a fixed network, VICTORIA HEEKYUNG KIM et al[22]. proposed a dynamically reconfigurable CNN accelerator that can change the network by replacing the internal configuration file of the FPGA, which undoubtedly improves the flexibility of the system and allows the edge device to change the neural network at any time to cope with different task scenarios. However, the dynamic reconfiguration method by replacing the BIT file requires each set of networks to generate a corresponding set of configuration files in advance, which is undoubtedly complicated in practical applications.

It is a challenge to efficiently accelerate the pose estimation network to achieve real-time frame rates and lower power consumption to cope with complex pose estimation scenarios in real applications. A common approach is to quantise the network to reduce the number of parameters and computational complexity of the network. For example, Miyama M et al[23]. in their design of an accelerator for semantic segmentation accelerators quantised the network parameters in 3-bit quantisation, which significantly improves the inference efficiency of the network, but at the same time the impact on the network accuracy makes it difficult to be applied in real-world scenarios where accuracy is required. It is known from

Vanhoucke et al[24]. that 8-bit quantisation can significantly speed up the inference process, and Gysel et al[25]. show that data with 8-bit fixed-point representation is basically comparable to data with 32-bit floating-point representation in terms of accuracy. The approach taken in this work is the 8-bits quantisation approach, but unlike the work [20] in where the network was trained quantitatively, we chose to quantise the parameters of the trained network, an approach that is more flexible and more conducive to shortening the deployment time of the bit-pose estimation network in practical applications. Another commonly used model compression method is to prune or compress the network itself, which can effectively reduce the amount of computation during network inference. This approach is mainly applied to the fully-connected layer of the CNN, and the pruning operation on the convolutional layer is often not effective, and since the pose estimation network generally does not have a fully-connected layer, this technique is not used in this work.。

In this paper, a low-power bit-pose estimation accelerator for multi-scenario applications is proposed. and is deployed using the Virtex UltraScale+ VU9P FPGA platform. In the accelerator development phase, we use the SpinalHDL development language, a choice that greatly simplifies the development process and allows for significant code flexibility. For model compression, we chose 8-bit quantisation, which significantly reduces the computational effort with little impact on accuracy. In order to enable dynamic switching of multiple networks on our accelerator, our architecture uses instruction-controlled data flow to compute in individual operators, which enables any network supported by the operator library to be deployed on the accelerator without changes to the accelerator itself. This agile design not only enables accelerators to make network changes on a real-world basis at the edge where scenarios are complex, but also shortens the workflow from the new network to the accelerator deployment.

III. NETWORK MODEL ANALYSIS

In this work, two networks, PVNet and MobileNetV2, are used to validate the functionality and agility of the accelerator. As a pose estimation network, PVNet maintains good performance even when the target is heavily truncated or interfered with.7 The main body of the network is divided into four parts: an initial convolutional layer for extracting details in the space, a rich feature information captured through the ResNet18 backbone network, a directional map convolutional layer to obtain directional information at key points, and a confidence convolutional layer to obtain the heat of each key point. Power map. In Table 1, we analyse the computational resource distribution of PVNet.MobileNetV2 is a lightweight network, where we remove the fully-connected layer behind the network and introduce a pixel voting mechanism to make it applicable to pose estimation.The main feature of MobileNetV2 is the inverted residual structure, which is a bottleneck layer that maps the input high-dimensional features into a lowdimensional space and then then recovered to high dimensions through an extension layer. This design significantly reduces the computational effort and preserves feature information through jump connections. In Table 2 we analyse the computational resource distribution of MobileNetV2. By analysing the ratio of computational resources of the two bit-pose estimation networks, it can be seen that the main computational resources of the network are allocated in the feature extraction and feature enhancement phases, in which convolutional computation is dominant, for this reason, this work has made data reuse and optimization strategies for high-channel convolutional computation of the accelerator, which will be mentioned in the following sections. After the network computation, the feature points are converted into the pose block diagram of the object by the PnP algorithm, which is outside the scope of this work, so the final output of the accelerator is the feature point diagram.

Table.1 PVNet network capacity

B	L	INPUT SIZE	operate	FLOPs (M)
#1	1	320×320×1	CONV	1006.08
	2	160x160x64	MAXPOOL	0
	3	80x80x64	residual block	147.456
	4	80x80x64	residual block	147.456
	5	40x40x128	residual block	36.864
	6	20x20x256	residual block	9.216
#2	7	10x10x512	CONV,DownSimple	188.743
	8	10x10x256	CONV,DownSimple	47.185
	9	10x10x128	CONV,DownSimple	9,437
#3	10	10x10x64	CONV,UpSimple	6.968
#4	11	10x10x64	CONV,UpSimple	3.484
Total				1602.889

Table.2 MobileNetV2 network capacity

B	L	INPUT SIZE	operate	FLOPs (M)
#1	1	320x320x1	CONV	294.912
#2	2	160x160x32	Inverted Residual	442.368
	3	160x160x16	Inverted Residual	106.168
	4	80x80x24	Inverted Residual	23.311
	5	80x80x24	Inverted Residual	11.640
	6	40x40x32	Inverted Residual	7.258
	7	40x40x32	Inverted Residual	7.258
	8	40x40x32	Inverted Residual	7.368
	9	20x20x64	Inverted Residual	7.147
	10	20x20x64	Inverted Residual	7.147
	11	20x20x64	Inverted Residual	7.147
	12	20x20x64	Inverted Residual	10.721
	13	20x20x96	Inverted Residual	16.108
	14	20x20x96	Inverted Residual	16.108
	15	20x20x96	Inverted Residual	11.289
	16	10x10x160	Inverted Residual	10.373
	17	10x10x160	Inverted Residual	10.373
	18	10x10x160	Inverted Residual	16.065
	19	10x10x320	CONV	41.472
#3	20	10x10x1280	CONV,UpSimple	59.904
	21	20x20x256	CONV,UpSimple	47.185
	22	40x40x128	CONV	18.874
#4	23	40x40x64	CONV,UpSimple	6.968
#5	24	40x40x64	CONV,UpSimple	3.484
Total				1297.308

IV. ACCELERATOR DESIGN

A. Overall structure

In order to enable networks to dynamically switch among accelerators in real time, we adopt a collaborative architecture, and the overall system architecture is shown in Fig. 1. The compiler is responsible for generating the corresponding control instructions according to the network to be accelerated and does not participate in the acceleration

process of the network. The host computer is responsible for collaborating with the accelerator, sending the instructions, weights, and data required by the accelerator to the accelerator, and reading the data in the DDR after receiving the completion signal from the accelerator and carrying out the corresponding post-processing, such as the PnP algorithm, etc. All the data control inside the accelerator is carried out by the DDR. The data control inside the accelerator is all taken care of by the instruction decoder, while the instructions controlling the accelerator to perform the computation are generated by the compiler before the accelerator is deployed, and then written into the instruction registers of the accelerator by the host computer through the AXI-Lite bus. Similarly, the image data required for the accelerator computation and large data such as weights are written by the host computer via the AXI bus to the DDR to be processed. When the decoder detects the start instruction or the completion signal of the previous instruction in the instruction register, it starts to decode the next instruction and calls the corresponding module in the arithmetic library to complete the corresponding calculations. If the next instruction is the completion instruction, the accelerator sends a completion interrupt signal to the host computer, which is a complete acceleration process.

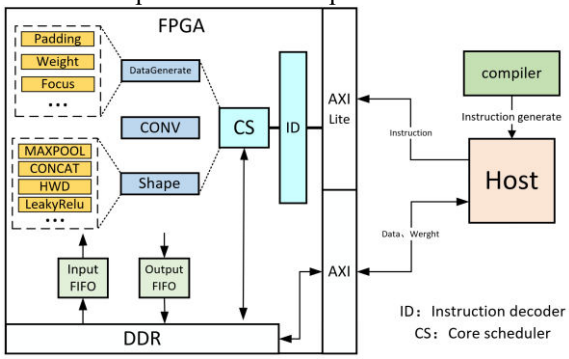

Fig.1 Overall System Architecture

In this work, the host computer can also achieve dynamic clock output by accessing the DRP through configuration registers using the AXI-Lite interface, with the control module shown in Figure 2. This design allows the accelerators deployed at the edge to increase the operating frequency in real time according to the intensity of the work to achieve higher real-time performance, and also reduce the operating frequency to reduce power consumption. In practical tests, under the premise of meeting the timing convergence, this accelerator module supports an operating frequency of up to 225MHz, and the power consumption under different operating intensities is discussed in detail in Section V.

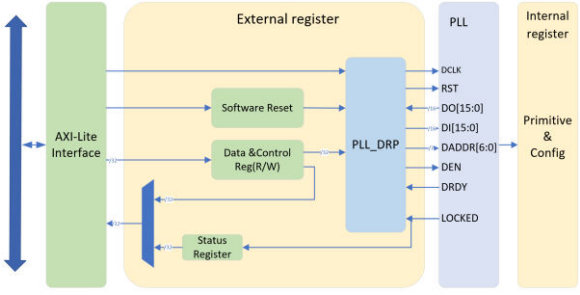

Fig.2 Clock Control Module

B. Instruction set design

In this work, the accelerator works by controlling the data flow through the decoder and the size and storage address of the data to be computed and the type of computation to be performed are stored in each instruction. In this architecture, the design of the instruction set is particularly important. The use of a complex instruction set will increase the user's control over the details of the accelerator's operation, but it will also increase the complexity of the decoding and will not be conducive to subsequent development. In this design, our instruction format is shown in Figure 3. The length of a single instruction is 40 bits, where 38 and 39 bits are instruction categories (green) and 0 to 37 bits are instruction details (blue and orange). The instructions are classified based on the category of the instruction, and the instructions in this work are classified into four categories: state control, DMA read/write, parameter setting, and operator selection. After the decoder recognises the category part of the instruction, it performs the corresponding work according to the different values. The state control instructions mainly include the start and stop of the accelerator, the start of the operator after the data loading is finished, and the status return instruction which is used by the host computer to check the current progress of the accelerator.The first 2bits of the DMA read/write instructions are used to differentiate between read/write functions and read/write channels, because the inputs of many operators in the accelerator consist of two inputs (e.g., CONV), and so they contain two write instructions and four read instructions. instructions. The first bit of the operator type selection instruction is used to distinguish whether the selection is for a CONV operator or an operator in Shape. This instruction is used to configure the parameters required for the operator to be run, and the first 4 bits of this instruction are used to activate the corresponding registers to receive the parameters. Some of the parameter setting instructions are shown in Fig. 3, such as the convolution input and output sizes, the pre-convolution Padding parameter settings, and the settings of the operator parameters in Shape. When designing the instructions, we reserve some expansion space for each type of instruction for subsequent development of new operators and other work.

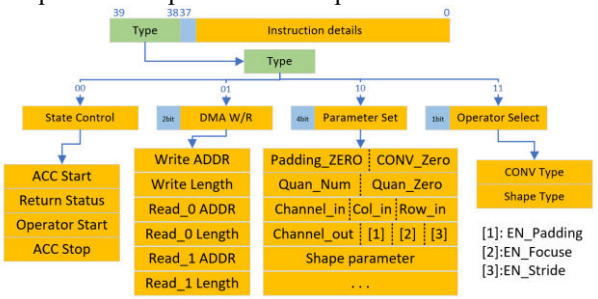

Fig.3 Instruction Format Design

C. Arithmetic library development

In this work, the operators in the operator library are mainly classified into two categories based on their functions in the network: the CONV module and the Shape module. The CONV module is responsible for the convolutional computation in the network. During the design of the accelerator, we integrate the data preparation module before convolution and the BN (batch normalisation) and activation

operations after convolution in the CONV module, which greatly reduces the bandwidth pressure and time loss caused by the data mobilisation between the DDR and the accelerator. The Shape module contains a number of network operators such as MaxPool, Concat, MeanPool, etc. to support network diversity.

Based on the network analysis in Section III, it can be learnt that the convolutional computation is the most computationally intensive and time consuming part of the bit-posture estimation network. In the conventional computational flow, in the case that the input size and output size of the feature map are equal, the convolutional computation flow is shown in the computational flow 1, and the determinants of the time complexity consumed by the convolutional computation are the number of convolutional kernels, the size of the convolutional kernels, the image size, and the image channel, and the size of the convolutional kernels is generally 2 × 2 or 3 × 3 in the conventional convolutional network, and in this work, this part is first parallelised This part is first parallelised in this work by setting Kh×Kw multipliers to unroll the loop part so that the computation can be completed in one clock cycle in the accelerator's computation flow. In addition to parallelisation by unfolding the convolution kernel, parallelisation can also be achieved by unfolding the input channel loops. In addition, the present work can also set up more than one of the above operations at the same time to achieve multiple convolutional kernels at the same time to parallelise the computation of the output channel. The efficiency of parallel processing for input and output channels depends on the amount of computational resources, and this work parameterises the hardware code of this design so that the accelerator can set the degree of parallelism according to the different computational resources of the edge device in practical applications. The computational flow II shows the computational complexity after parallelisation.

calculation process 1 ： Conventional Convolution Process

```
for N to Number of Kernel
  for H to Height of Input, W to Width of Input
        for C to Channel of Input
          for Kh to Height of Kernel, Kw to Width of Kernel
           output[n][h][w]+=Input[c][h+kh][w+kw]*Kernel[n][c][kh][kw]
```

calculation process 2 ： Accelerating the convolution process

```
for N to Number of Kernel / Parallelism of Kernel
  for H to Height of Input, W to Width of Input
        for C to Channel of Input / Parallelism of Channel
        output[n][h][w]+=Input[c][h+kh][w+kw]*Kernel[n][c][kh][kw]
```

Fig. 4 shows the convolutional processing flow inside the re-accelerator, where the size of the convolutional kernel is assumed to be 3 × 3. In order to parallelise the convolutional kernel, we add two row buffers in the data collation stage, which are used to generate three rows of feature data for computation. After the data collation is completed, the feature points of each row start flowing into the computational units for multiplication computation.The number of computational units depends on the size of the convolutional kernel as well as the parallelism of the input and output channels. In the figure it is assumed that the input channel parallelism and output channel parallelism is 3. After the completion of the multiplication operation, the obtained 9 pixel points and the results of each channel are summed up through the addition tree to get the single channel result of a pixel in the output result.

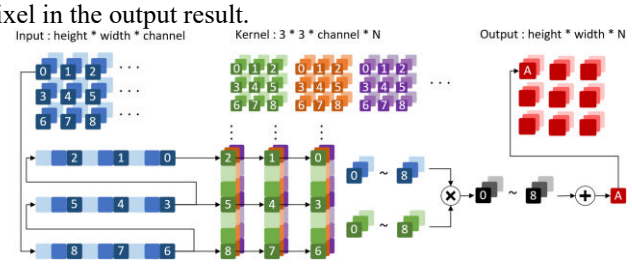

Fig.4 Convolutional Processing Flow

When the on-chip resources are sufficient, the output of the convolution can be derived by repeating the above operation. However, not all edge-side chips have sufficient resources, especially when the number of parameters of some convolution operations is large, the computational resources consumed to perform one convolution are too much, in order to cope with this situation, we can also lighten the convolution processing. The process of distributed convolution is shown in Fig. 5, which divides the image and convolution kernel into two pieces by channel, and performs two convolution and quantisation operations on the two parts to obtain the corresponding results. After that, the previously obtained results are summed and quantised by channel to obtain the full results of this convolution. This calculation method is to divide the original one convolution instruction into two convolution instructions and one summing instruction. Practical tests show that the distributed convolution process consumes about 1.05 times as much time as a normal convolution operation, while the input image and the number of parameters remain unchanged, and most of the extra time consumed is the time spent on moving data between the DDR and the FPGA. Distributed convolutional computation allows resource-scarce edge chips to perform convolutional computation with high number of parameters at a small cost of time while maintaining computational accuracy.

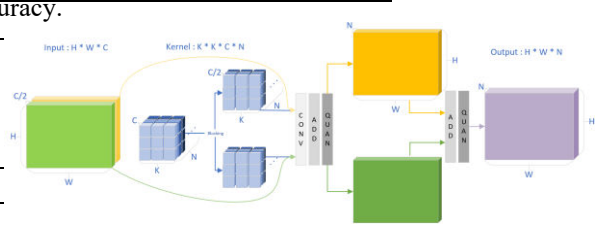

Fig.5 Flow of distributed convolution

The Shape module is responsible for providing support for operators other than convolutional computation in the network. In this work, the size of the operators in the Shape module is not fixed, and their number can be increased or decreased depending on the network to be accelerated. This approach allows the accelerator to support a wider variety of networks, and also reduces unnecessary operators according to the network to be accelerated, thus reducing unnecessary resource usage and power consumption. In this work, the operator library contains commonly used network operators such as MaxPool, Concat, Add, Upsampling, Mul and MeanPool. The hardware implementations of these classical operators have been described in many previous works and will not be repeated in this paper. In addition to these classical operators, in order to support newer network structures, we have also developed the HWD operator, which achieves better results in image processing algorithms

compared to other traditional downsampling algorithms[26].The hardware implementation of the HWD operator is shown in Fig. 6.

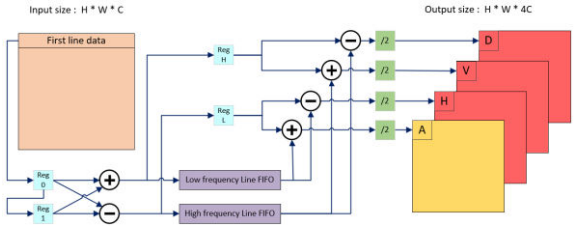

Fig.6 HWD operator architecture

V. RESULTS

A. Data sets and evaluation indicators

We chose the LineMod dataset as our test dataset during the functional validation of the pose estimation accelerator. This is a standard 6D object pose estimation dataset containing multiple objects, viewpoints, and occlusion situations, and this dataset is more conducive to verifying the ability of our accelerator to work in complex situations. For accuracy evaluation, we choose ADD (Average Distance of Model Points) as the judging metric. In the usual judging criteria, the ADD value of the network estimation result is less than or equal to 10% is regarded as the correct estimation of the object's pose, and this criterion is also adopted for our accuracy calculation.

In this work, the quantised PVNet and MobileNetV2 network accuracies are shown in Table 3.PVNet, due to its excellent performance in complex situations such as object occlusion, achieved 84.53% accuracy in all categories of the accuracy test, and 95.20% accuracy in the category of Eggbox, and MobileNetV2, due to its MobileNetV2 is slightly less accurate than PVNet due to its lightweight nature, achieving 78.42% accuracy. In addition, we also list the unquantised PVNet and MobileNetV2 network accuracies as a comparison, and it can be seen that the 8bit quantisation strategy adopted in this work has little effect on the accuracy.

Table.3 PVNet and MobileNetV2 network accuracy

Module	Data Type	GFLOPS (G)	AP
PVNet	Float-32	1.602	0.862
MobileNetV2	Float-32	1.297	0.809
PVNet	Int-8	--	0.845
MobileNetV2	Int-8	--	0.784

The results of the PVNet as well as MobileNetV2 networks are shown in Fig. 7 and Fig. 8. Since the present accelerator does not include the PnP object pose frame generation aspect of object pose estimation, the graphs are shown as a visualisation of the keypoints of the network output.

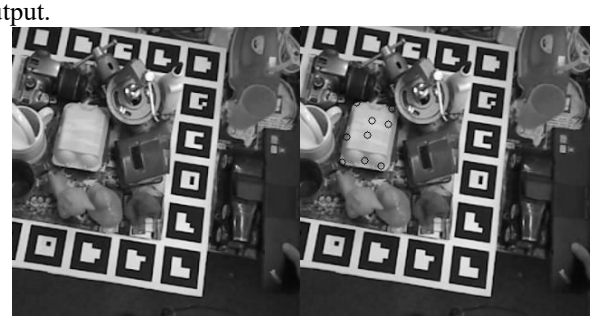

Fig.7 PVNet Network Attitude Estimation Effect

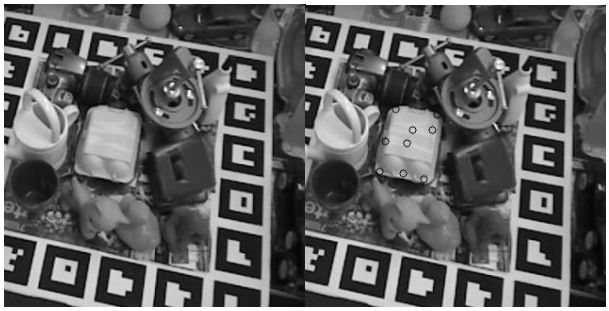

Fig.8 MobileNetV2 Network Attitude Estimation Effect

B. Resource usage and power consumption

Table 4 shows the resource consumption of our accelerator when deployed on the Virtex UltraScale+ VU9P platform. Compared to the work of [20], which is a dedicated accelerator, our LUTs and FFs consume less resources and achieve more flexible acceleration. Moreover, this data is the performance of the Shape module with all supported operators fully loaded, and in the actual deployment, the number of operators in the module can be deleted or reduced according to the different accelerated networks to reduce the resource consumption. The accelerator also supports real-time dynamic adjustment of the operating frequency to control the power consumption of the accelerator, the power consumption data under different operating frequencies are shown in Table 5.

Table.4 Accelerator resource consumption

Resource	LUTs	FFs	BRAMs	DSPs
Our Work	107954	157763	563.5	1033
[20]	131187	209200	348.5	642

Table.5 Power consumption of accelerators at different operating frequencies

Frequency (hz)	200M	125M	50M
Total Power(W)	11.585	9.706	7.997

C. Comparison of work

In this work, we calculate the FPS by measuring the difference between when the host computer sends the start signal to the accelerator and when the host computer receives the end signal from the accelerator.Most of the work calculates the FPS by measuring the difference in time between when the accelerator reads the initial data from the DDR and when it writes the result back to the DDR.For the same work, our way is lower in terms of the data compared to this way. , but our measurement is more in line with the performance effect when actually deployed. Table 6 shows the comparison of our work with other work. Compared to other existing state-of-the-art work, our flexible architecture outperforms [27] in terms of DSP utilisation, and our work meets the real-time requirements and outperforms [28] in terms of FPS. Compared to [20], a lightweight dedicated bit-pose estimation accelerator, our accelerator has some disadvantages in terms of frame rate and power consumption, but our architecture is more suitable for real-time scenarios with complex environments and higher accuracy.

Table.6 job comparison

	[28]	[29]	[20]	Our work	Our work
Platform	Arria10 SoC	ZC706	XCK325T	VU9P	VU9P

Network	MobileNet V2	MobileNet V2+SSDLite	MobileNetV2+ LightPose	PVNET	MobileNetV2 + Pixelwise Voting
Freq (MHz)	150	100	188	200	200
DSP Utilization	1278	728	642	1033	1033
FPS	226.2	64.8	411.6	98.36	130.24
Power(W)	--	9.9	5.03	11.585	11.585

VI. SUMMARY

We propose a low-power bit-posture estimation accelerator for multi-scenario applications and deploy it on a Virtex UltraScale+ VU9P FPGA. The accelerator achieves excellent power performance with guaranteed accuracy and real-time performance, allowing it to be used for real-time efficient pose estimation in complex environments.

REFERENCES

[1] Hui, W., Aiyuan, L.: A systematic approach for english education model based on the neural network algorithm. Journal of Intelligent & Fuzzy Systems 40(2), 3455–3466 (2021)
[2] Wang, J., Chai, W., Venkatachalapathy, A., Tan, K.L., Haghighat, A., Veli pasalar, S., Adu-Gyamfi, Y., Sharma, A.: A survey on driver behavior analysis from in-vehicle cameras. IEEE Transactions on Intelligent Transportation Sys tems 23(8), 10186–10209 (2021)
[3] Sun, Z., Ke, Q., Rahmani, H., Bennamoun, M., Wang, G., Liu, J.: Human action recognition from various data modalities: A review. IEEE transactions on pattern analysis and machine intelligence 45(3), 3200–3225 (2022)
[4] Sereno, M., Wang, X., Besan¸con, L., Mcguffin, M.J., Isenberg, T.: Collaborative work in augmented reality: A survey. IEEE Transactions on Visualization and Computer Graphics 28(6), 2530–2549 (2020)
[5] Kosch, T., Karolus, J., Zagermann, J., Reiterer, H., Schmidt, A., Wo´zniak, P.W.: Asurvey on measuring cognitive workload in human-computer interaction. ACM Computing Surveys 55(13s), 1–39 (2023)
[6] Guan, J., Hao, Y., Wu, Q., Li, S., Fang, Y.: A survey of 6dof object pose estimation methods for different application scenarios. Sensors 24(4), 1076 (2024)
[7] Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: Pixel-wise voting net work for 6dof pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4561–4570 (2019)
[8] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
[9] Yuan, R., Jun, P., Jingjing, L., et al.: Overview of artificial intelligence chip development. Micro/Nano Electronics and Intelligent Manufacturing 1(2), 20–34 (2019) 15
[10] Li, Z., Zhang, Y., Wang, J., Lai, J.: A survey of fpga design for ai era. Journal of Semiconductors 41(2), 021402 (2020)
[11] Kasiselvanathan, M., Sangeetha, V., Kalaiselvi, A.: Palm pattern recognition using scale invariant feature transform. International Journal of Intelligence and Sustainable Computing 1(1), 44–52 (2020)
[12] Agrawal, P., Sharma, T., Verma, N.K.: Supervised approach for object iden tification using speeded up robust features. International Journal of Advanced Intelligence Paradigms 15(2), 165–182 (2020)
[13] Ghaffari, S., Soleimani, P., Li, K.F., Capson, D.W.: Analysis and comparison of fpga-based histogram of oriented gradients implementations. IEEE Access 8, 79920–79934 (2020)
[14] Sikka, P., Asati, A.R., Shekhar, C.: Real time fpga implementation of a high speed and area optimized harris corner detection algorithm. Microprocessors and Microsystems 80, 103514 (2021)
[15] Ahmed, M.S., Aurpa, T.T., Azad, M.A.K.: Fish disease detection using image based machine learning technique in aquaculture. Journal of King Saud University-Computer and Information Sciences 34(8), 5170–5182 (2022)
[16] Silpa-Anan, C., Hartley, R.: Optimised kd-trees for fast image descriptor match ing. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008). IEEE
[17] Pan, S., Wang, X.: A survey on perspective-n-point problem. In: 2021 40th Chinese Control Conference (CCC), pp. 2396–2401 (2021). IEEE
[18] Zhang, J., Yao, Y., Deng, B.: Fast and robust iterative closest point. IEEE Transactions on Pattern Analysis and Machine Intelligence 44(7), 3450–3466 (2021)
[19] Sohrabizadeh, A., Wang, J., Cong, J.: End-to-end optimization of deep learning applications. In: Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 133–139 (2020)
[20] Wang, X., Zhang, Z., Wang, Y., Cai, C., Chen, G.: A fast and efficient fpga-based pose estimation solution for iot applications. In: 2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 1–4 (2022). IEEE
[21] Fan, H., Liu, S., Ferianc, M., Ng, H.-C., Que, Z., Liu, S., Niu, X., Luk, W.: A real-time object detection accelerator with compressed ssdlite on fpga. In: 2018 International Conference on Field-programmable Technology (FPT), pp. 14–21 (2018). IEEE
[22] Kim, V.H., Choi, K.K.: A reconfigurable cnn-based accelerator design for fast and energy-efficient object detection system on mobile fpga. IEEE Access 11, 59438–59445 (2023) https://doi.org/10.1109/ACCESS.2023.3285279
[23] Miyama, M.: Fpga implementation of 3-bit quantized cnn for semantic segmenta tion. In: Journal of Physics: Conference Series, vol. 1729, p. 012004 (2021). IOP Publishing
[24] Vanhoucke, V., Senior, A., Mao, M.Z., et al.: Improving the speed of neural net works on cpus. In: Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop, vol. 1, p. 4 (2011)
[25] Gysel, P., Motamedi, M., Ghiasi, S.: Hardware-oriented approximation of convo lutional neural networks. arXiv preprint arXiv:1604.03168 (2016)
[26] Xu, G., Liao, W., Zhang, X., Li, C., He, X., Wu, X.: Haar wavelet downsampling: A simple but effective downsampling module for semantic segmentation. Pattern Recognition 143, 109819 (2023)
[27] Zhao, R., Niu, X., Luk, W.: Automatic optimising cnn with depthwise separable convolution on fpga: (abstact only). In: Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 285–285 (2018)
[28] Su, J., Faraone, J., Liu, J., Zhao, Y., Thomas, D.B., Leong, P.H., Cheung, P.Y.: Redundancy-reduced mobilenet acceleration on reconfigurable logic for imagenet classification. In: Applied Reconfigurable Computing. Architectures, Tools, and Applications: 14th International Symposium, ARC 2018, Santorini, Greece, May 2-4, 2018, Proceedings 14, pp. 16–28 (2018). Springer