electronics

Article

Feasibility Analysis and Implementation of Adaptive Dynamic Reconfiguration of CNN Accelerators

Ke Han * and Yingqi Luo

School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
* Correspondence: hanke@bupt.edu.cn

Abstract: In multi-tasking scenarios with dynamically changing loads, the parallel computing of convolutional neural networks (CNNs) causes high energy and resource consumption in the system. Another critical problem is that previous neural network hardware accelerators are often limited to fixed scenarios and lack the function of adaptive adjustment. To solve these problems, a reconfiguration adaptive system based on the prediction of algorithm workload is proposed in this paper. Deep Learning Processor Unit (DPU) from Xilinx has excellent performance in accelerating network computing. After summarizing the characteristics of hardware accelerators and gaining an in-depth understanding of the DPU structure, we propose a regression model for CNNs runtime prediction and a guidance scheme for adaptive reconfiguration combined with the characteristics of Deep Learning Processor Unit. For different DPU sizes, the accuracy of the proposed prediction model achieves 90.7%. With the dynamic reconfiguration technology, the proposed strategy can enable accurate and fast reconfiguration. In the load change scenario, the proposed system can significantly reduce power consumption.

Keywords: accelerators; convolutional neural networks; adaptive adjustment; prediction model; FPGA

Citation: Han, K.; Luo, Y. Feasibility Analysis and Implementation of Adaptive Dynamic Reconfiguration of CNN Accelerators. *Electronics* **2022**, *11*, 3805. https://doi.org/10.3390/electronics11223805

Academic Editor: Sunggu Lee

Received: 30 October 2022
Accepted: 17 November 2022
Published: 18 November 2022

1. Introduction

Convolutional neural networks (CNNs) have shown significant improvement and high maturity in many computer vision applications. For example, Advanced Driving Assistance System (ADAS) involves image classification, object detection, image segmentation, scene reasoning, and other tasks [1]. Demands for complexity and accuracy have been prioritized regarding using CNNs in the past few years. In addition to deep neural networks, many lightweight networks for edge terminal equipment and embedded devices have also been proposed. The option of using a lightweight network with depthwise separable convolution can greatly reduce the system burden in case accuracy is not required as much as aforementioned [2]. The deployment of CNNs in various scenarios has become a major trend of future development, which greatly increases the need for a single system to deal with multiple scenarios. In the face of today's complex application scenarios, the switching of the two types of CNNs and the difference in the running rate of the algorithms will lead to changes in the system workload.

Another reality that has to be faced is that a high degree of automation means a significant increase in computing power consumption, which cannot be ignored. On the premise of ensuring computability, the key to reducing power consumption is to increase the adaptability of CNN accelerators for dynamically changing scenes. Fixed hardware architectures are prone to redundant or insufficient performance when the load changes. The key to high flexibility lies in the cooperation between general acceleration units and dynamic adjustment algorithms. Hence, the main work of adaptive reconfiguration is to adjust circuit characteristics in the FPGA according to changes in algorithm requirements. The Xilinx Deep Learning Processing Unit (DPU) [3] is a configurable computation engine optimized for convolutional neural networks. The degree of parallelism utilized in the engine

is a design parameter and can be selected according to the target device and application. It includes a set of highly optimized instructions and supports most convolutional neural networks, such as VGG [4], ResNet [5], GoogLeNet [6], YOLO [7], SSD [8], MobileNet [2], FPN [9], and others. At the same time, it also supports dynamic reconfiguration [10] of the circuit when the resources on the board are sufficient. Therefore, the DPU is very suitable as a design and verification platform of the model, which is conducive to collecting as much data as possible and guarantees strong expansibility. The adaptive platform based on DPUs can be used as a reference for other FPGA accelerators in similar work.

This study explores the characteristics of DPUs under standard convolution and depthwise separable convolution. Our work integrates the standard convolution network and lightweight network, and proposes a prediction model for adaptive reconfiguration based on the collected data sets. Moreover, an adaptive reconfiguration scheme is designed based on the model, which is optimized for the power consumption and resource utilization of CNNs on the DPU platform. The accuracy of the prediction model has reached more than 90%, and system flexibility is greatly improved with the aid of the adaptive reconfiguration scheme. The proposed method can be used as an idea to construct an adaptive reconfiguration system. The main contributions of this paper are as follows:

1. Analysis of DPUs under dynamic load in terms of power consumption and resources that provides basic guidance for adaptive reconfiguration;
2. A runtime prediction model that considers a combination of the standard convolution and depthwise separable convolution for reconfiguration. It ensures the prediction accuracy of depthwise separable convolution;
3. A model-based adaptive reconfiguration scheme, successfully deployed and validated in DPUs. The adaptive reconfiguration function significantly improves the system resource utilization and reasonably reduces power consumption.

The rest of the paper is organized as follows: Section 2 provides a research background of CNN accelerators for two different convolutions, as well as related research on DPUs. Section 3 presents the basic research work of DPUs in terms of reconfiguration and power consumption. Section 4 introduces the thinking and implementation of DPUs runtime modeling and prediction. Section 5 describes the dynamic reconfiguration scheme based on Section 4. Section 6 shows the details and results of the experiment Section 7 concludes the paper and presents our vision for future work.

2. Background

Due to the high parallel computing characteristics, CNNs quickly attracted the interest of researchers in hardware implementation after it had already been applied on a large scale. Yu-Hsin Chen et al. proposed Eyeriss to minimize the energy cost of data movement [11]. Aiming at the pruned model, Sangkug Lym et al. proposed FlexSA, a flexible systolic array architecture, in order to make the systolic array more efficient for pruning and training [12]. To accelerate various CNN models, researchers tend to design a general accelerator architecture. FPGA has been used as the acceleration platform of CNNs by many researchers because of its flexibility and reconfigurability, which can maximize the benefit of hardware [13].

Standard convolution networks have high computational complexity and a large amount of data, which are the main reasons for the high cost and difficulty in deploying them at terminals with limited resources [14]. The lightweight network has the advantage of fast inference speed and low power consumption when implemented in hardware. It can be seen in Figure 1 that for the standard convolution, an image with size $M \times N$ and a number of input channels as C_{in} needs to be convolved with $C_{out} \times K \times K \times C_{in}$ pixels to get the output image of $R \times C \times C_{out}$. In contrast, input activations of each channel are convolved with weights in the same channel to produce output activations in the same channel in depthwise convolution. Ideally, deep convolutional layers account for a small fraction of total MAC operations [15,16]. Although depthwise separable convolution still takes advantage of the convolutional reuse in the spatial domain, most multipliers remain

unused when calculating the depthwise convolution layer. So, only a few multipliers process the entire depthwise convolution layer in a large number of clock cycles, which degrades the overall system performance.

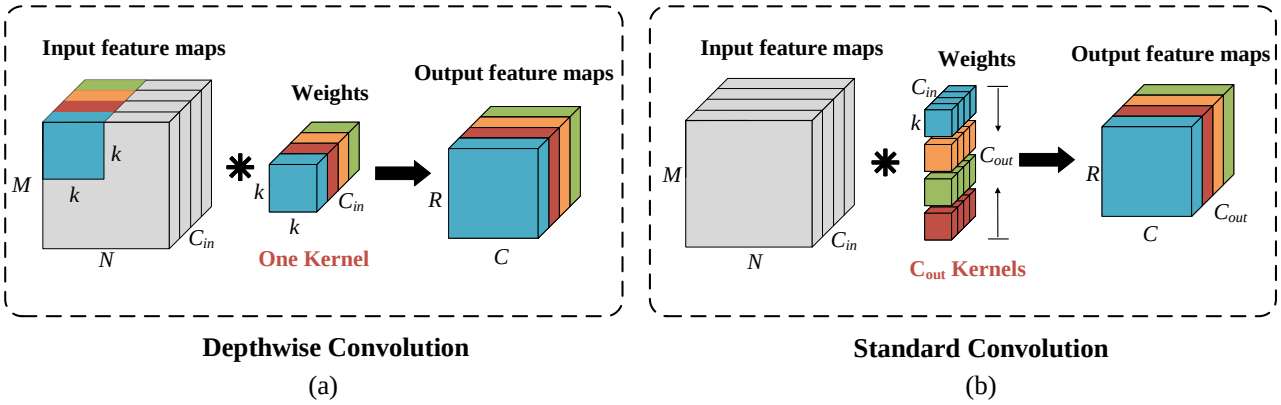

Figure 1. Comparison of computational modes between (**a**) depthwise convolution with (**b**) standard convolution. The same color corresponds to the same convolution channel.

Due to their low computational intensity, they cause great resource redundancy in high parallelism accelerators [17]. Most previous research has focused on the related work of deep standard convolution and rarely combines depthwise separable convolution with it or takes depthwise separable convolution as the object of acceleration alone. For some scenarios that do not require large computation, it is necessary to use lightweight networks and further optimize their hardware deployment.

In terms of flexible deployment of Xilinx DPUs, Rajesh Kediad et al. studied the design space exploration of DPU based on a FPGA platform [18], demonstrating the feasibility and flexibility of different DPU deployments. Furthermore, they estimated the energy consumption of different CNN models on the DPU platform [19]. Rajesh Kediad's research motivated us to use DPUs for adaptive reconfiguration. For the characteristics of DPUs when running on different networks, Yutian Lei et al. analyzed the resource utilization of DPUs running different CNN models [20]. At the same time, there is growing interest in predicting the performance of accelerators in FPGAs through algorithmic models [21].

Dynamic reconfigurability techniques have been applied to inference in convolutional neural networks. JinYong Yin's work leveraged coprocessor mechanism and partial reconfiguration to realize accelerator dynamic reconfiguration online [22]. Lei Gong et al. proposed a new accelerator architecture for implementing CNNs on FPGAs in which the static and dynamic reconfigurability of the hardware was cooperatively utilized to maximize the acceleration efficiency [23]. Hasan Irmak et al. presented a dynamically reconfigurable CNN accelerator architecture composed of reconfigurable macroblocks and utilized the device resources according to model parameters [24]. However, the previous work focused on the reconfiguration within the CNN algorithm during the inference process. Moreover, few studies have focused on circuit reconfiguration patterns following changes in task load. Internal reconfiguration is beneficial to the maximization of resource utilization. Nevertheless, for the case requiring low delay and multiple CNNs, internal reconfiguration will increase the inference time and is not compatible with multiple CNNs. Therefore, a single CNN algorithm is taken as the minimum unit of reconfiguration in our study. Standard convolution and depthwise separable convolution is combined to achieve multi-dimensional reconfiguration.

3. Design Exploration of DPU Architecture and Power

This section explores the architecture of DPUs and their operating characteristics and provides a basis for subsequent research. The detailed hardware architecture of a DPU is shown in Figure 2. The DPU fetches instructions from the off-chip memory to control

the calculation of the computing engine. The instructions are generated by the Vitis AI compiler [25]. The AI compiler maps the model to a highly-efficient instruction set and performs sophisticated optimizations such as layer fusion, instruction scheduling, and reuses on-chip memory as much as possible. On-chip memory is used to buffer input, intermediate, and output data to achieve high throughput and efficiency. Input data for maps and weights is reused as much as possible to reduce the memory bandwidth. A deep pipelined design is used for the computing engine. The processing elements (PE) take full advantage of the fine-grained building blocks such as multipliers, adders, and accumulators in Xilinx devices.

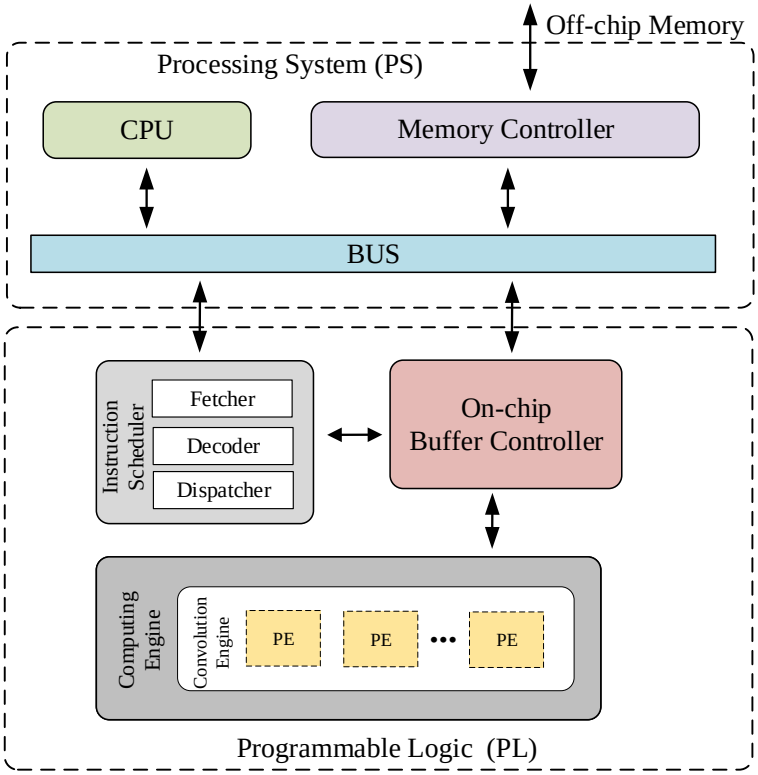

Figure 2. DPU hardware architecture.

The DPU IP can be configured with various convolution architectures that are related to the parallelism of the convolution unit. The architecture for the DPU IP includes B512, B800, B1024, B1152, B1600, B2304, B3136, and B4096. *B* is the symbol of DPU size and the size of the following number represents the degree of computing parallelism, which equals the peak number of operations per cycle. Meanwhile, the peak number of operations per cycle is equal to the product of three dimensions of parallelism in the DPU convolution architecture: pixel parallelism, input channel parallelism, and output channel parallelism. The higher the degree of parallelism, the more FPGA resources are consumed. At the same time, if the resources are sufficient, FPGA can support the deployment of up to three core DPUs.

Due to the parameter characteristics and computation amount of different CNNs, their performance on the same DPU is different. The power consumption and performance of seven typical CNNs deployed on the dual-core B4096 DPU was measured. As seen from Figure 3, the operating power consumption of different networks varies greatly. Taking yolov3 and ssdmobilenetv2 for detection tasks as examples, the backbone network in yolov3 is standard convolution. In the forward calculation, the network fully utilizes the PE computing power inside the DPU. Most layers of ssdmobilenetv2 are depthwise separable convolution, which are less than one-third of the former in terms of computing power, but their power consumption is reduced from 13.45 W to 7.53 W. When running a lightweight

network in a specific task scenario, the traditional convolution deployment idea should be abandoned and the optimal solution to improve the system in terms of power consumption and resources must be proposed, which is the focus of our work.

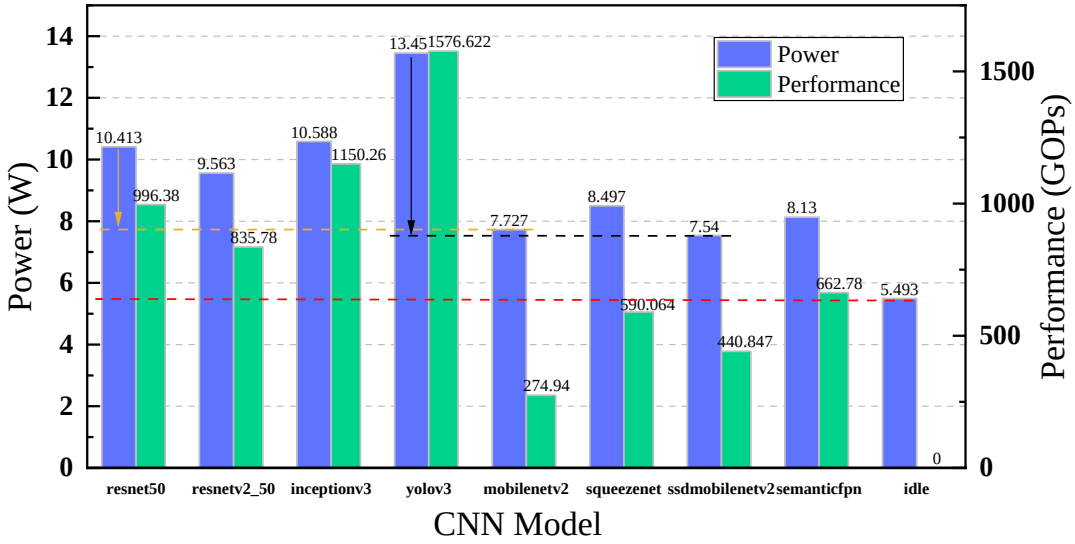

Figure 3. Power consumption and performance of different networks on dual-core B4096 DPU. The last column is the idle power consumption of the DPU when no network is running.

4. Proposed Approach for Runtime Estimation

The first principle to achieve reconfigurability is predicting the performance of different networks on different hardware configurations. Estimating the runtime of CNNs on different configurations of the DPU platform is a key issue to select the hardware configuration that satisfies the running conditions of the algorithm.

4.1. Roofline Model for DPU

The performance of DPUs is modeled to find the difference between standard convolution and depthwise separable convolution. The computation intensity of the algorithm determines its performance on the computing platform [26]. When computation intensity is less than a specific value, the algorithm will not be able to exert the best performance of the computing platform due to the limited bandwidth. When the intensity crosses this value, the actual performance is fixed below the theoretical maximum performance of the computing platform. The depthwise convolution layers of lightweight networks often suffer from memory bottlenecks.

The runtime of individual convolutional layers on the B4096 DPU is measured and collected. Then the computational load of each layer is added to get the relationship between computation intensity and performance. Figure 4 shows the relationship between computation intensity and actual performance of nearly 600 standard convolution and depthwise separable convolution layers. The blue line indicates the roofline performance of the model, which is the theoretical performance of a DPU with a frequency of 300 M and parallelism of 4096. It can be clearly seen that for B4096 DPU, depthwise separable convolution layers are all located in the memory bottleneck. In contrast, standard convolution layers are more computationally intensive and achieve better performance on the same circuit.

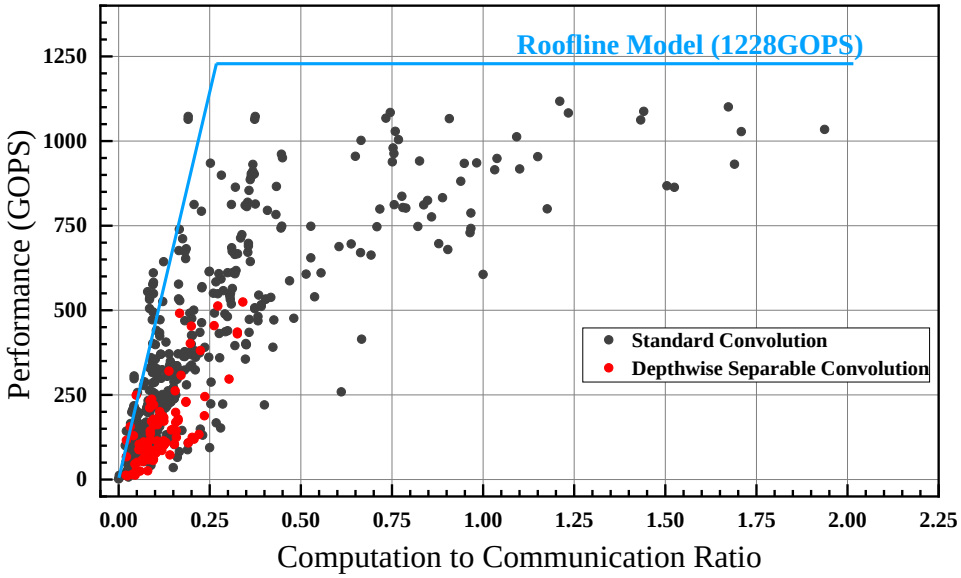

Figure 4. Relationship between computation intensity and the performance of standard convolution and depthwise separable convolution on B4096 DPU.

Due to the difference in the computing mode between standard convolution and depthwise separable convolution, low resource utilization will occur when computing them on the same platform. Moreover, as the degree of parallelism increases, this situation becomes more and more evident. When it comes to scenarios suitable for lightweight networks, an overly parallelized computing circuit has become the most significant drawback. The difference in computation intensity makes it impossible for a fixed hardware circuit to perform both algorithms efficiently. Therefore, our work will continue to explore how to solve the above problems by using a prediction model to guide reconfiguration when hardware circuits can be changed dynamically.

4.2. Regression Prediction Model for Runtime

Our study models the two types of convolution separately to resolve the difference between them and obtain the runtime of a complete network by predicting the runtime of each convolution layer and summing them together. The error between the predicted value and the actual value of the network runtime is used as the index to evaluate the prediction model. Computation intensity is included in the model as a characteristic parameter because it is closely related to the performance and runtime of CNNs, as can be seen from Table 1 and the previous analysis.

In the process of convolution operation, the factors that determine the calculation time are as follows:

1. DPU size ($Size_{DPU}$);
2. MAC operand (OPS);
3. Memory (Mem);
4. Intensity (Int);
5. Kernel size (k);
6. Input map size (M);
7. Output map size (R);
8. Number of input channels (C_{in});
9. Number of output channels (C_{out}).

The purpose of establishing the prediction model is to find the relationship between the runtime of a single-layer network on a specific DPU and the above independent variables. The runtime of a layer of CNN can be written as:

$$Runtime = f_n(Size_{DPU}, OPS, Mem, Int, k, M, R, C_{in}, C_{out}) \quad (1)$$

Since the characteristics of each hardware circuit are different, it is difficult to solve this type of CNN accelerator performance evaluation problem by theoretical derivation. Hence, linear regression, polynomial regression, decision tree regression, and random forest are selected as candidate models instead of complex machine learning methods. This kind of model also ensures low computational cost and avoids excessive complexity. A certain amount of measured data on B4096 DPU for model training was collected and part of the measured data for testing was reserved. Based on the data set, the prediction accuracy of the model verify whether it meets the requirements. Because the number of samples of depthwise separable convolution is less than that of standard convolution, the prediction performance will be slightly worse than that of standard convolution.

The r2_score metric and the mean of the prediction error are used to evaluate the model, which measures how well the prediction model fits the actual data. The closer the r2_score is to one, the better the performance of the model is. The calculation formula of the r2_score is as follows:

$$r2\_score = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2}$$

where i is the number of samples, y_i is the observation value, $\hat{y}_i$ is the prediction value, $\bar{y}$ is the average of observation values.

Through its estimated performance on B4096 DPU, Table 1 shows that the average error of the random forest model is 8.53% for standard convolution and 10.48% for depthwise separable convolution. The random forest regression model builds multiple unrelated decision trees by randomly selecting samples and features. Each decision tree can get a prediction result based on the samples and features extracted, and the regression prediction result of the whole forest can be obtained by taking the average of the results of all trees. Therefore, the random forest model is selected as the prediction model in our research, which is easy to implement and has little computational overhead.

Table 1. Prediction performance of regression models on B4096 DPU.

Prediction Model	Layer	Score	Mean Error
Linear regression	StandConv	0.972	12.23%
	DepthwiseConv	0.915	15.21%
Polynomial regression	StandConv	0.974	11.77%
	DepthwiseConv	0.922	13.67%
Decision tree	StandConv	0.959	10.19%
	DepthwiseConv	0.901	11.95%
Random forest	StandConv	0.968	8.53%
	DepthwiseConv	0.926	10.48%

In addition, we also calculated the correlation factors between the parameters and the runtime according to the model. According to the correlation factors in Table 2, the runtime of a single layer is mainly related to the amount of computation, computation intensity, the size of the kernel, and the size of the memory occupied. Nevertheless, the remaining features are still incorporated into the prediction model to get better prediction results.

Table 2. Correlation between the characteristics considered and the runtime.

Features Considered	Correlation Factor with Runtime
MAC operand	0.96
Memory	0.78
Intensity	0.58
Kernel size	0.39
Input map size	0.25

Table 2. *Cont.*

Features Considered	Correlation Factor with Runtime
Output map size	0.24
Input channel	−0.07
Output channel	0.34

5. Proposed Reconfiguration Scheme

5.1. Optimal Energy Consumption Scheme

The power consumption of a CNN running on FPGA depends on the running frame rate and the power consumption at runtime. DPUs with higher parallelism will use more resources and have higher power consumption.

In a real-time processing scenario, a system's total energy consumption to process each frame mainly comes from the sum of the running power consumption during real-time inference and the idle power consumption of the system. In Figure 5, if the running frame rate is F_s, the allowable processing time of each frame is $1/Fs = T_s$ and below. The actual runtime for DPU configurations meeting the above conditions is T_r. The power consumption during operation is P_r. The idle time is T_i, $T_i + T_r = T_s$. Idle power consumption is P_i. Then the energy consumption of actually running an image is:

$$E = P_r \cdot T_r + P_i \cdot T_i = (P_r - P_i) \cdot T_r + P_i \cdot T_s \tag{3}$$

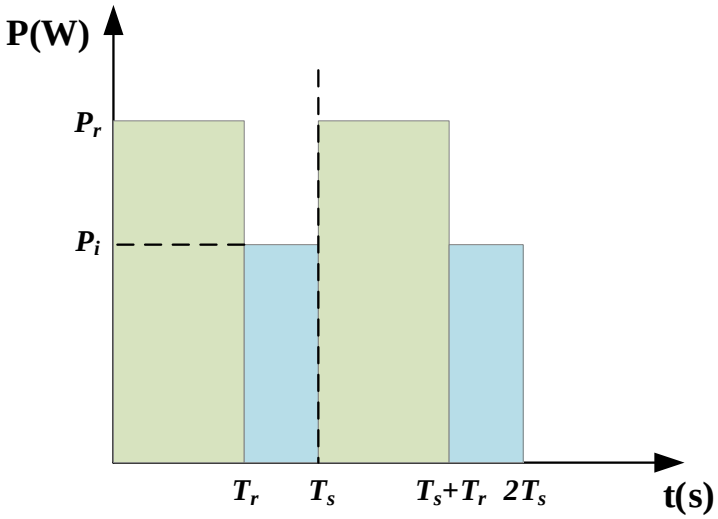

Figure 5. Power consumption in a real-time processing scenario. The green rectangle represents the energy consumption during operation and the blue rectangle represents the idle energy consumption of the system

For the measurement of system power, the voltage and current values are obtained by detecting the programmable power rails of the board and the energy consumption of each DPU for real-time fixed frame inference is calculated by Equation (3).

Because ResNet and MobileNet have become the basic components of most networks, seven representative networks are selected as research objects. Figure 6 shows that with the increase of DPU parallelism, the energy consumption of a specific network at a fixed frame rate tends to increase. Therefore, a DPU with a smaller size should be selected during reconfiguration to reduce the power consumption.

5.2. Optimal Hardware Resource Scheme

In the process of adaptive reconfiguration, the following question should be considered: should the strategy of using multiple small cores instead of one large core be chosen to reduce resource usage? Addressing this question requires calculating whether

the resources consumed by multiple small cores are less than that of large cores under the condition of providing the same computing power. Figure 7 shows that with the increase of hardware parallelism, the consumption of DSP as the primary computing unit increases significantly compared with other resources. So the DSP utilization determines whether to use a multi-core state or a single-core state. For example, when mobilenetv2 is running, due to the low resource utilization of depthwise separable convolution, one B3136 core is replaced by two B512 cores to save resources. Meanwhile, using two B2304 cores consumes more resources than one B4096 core when the processed CNN is yolov3, although they all meet the reconfiguration requirement.

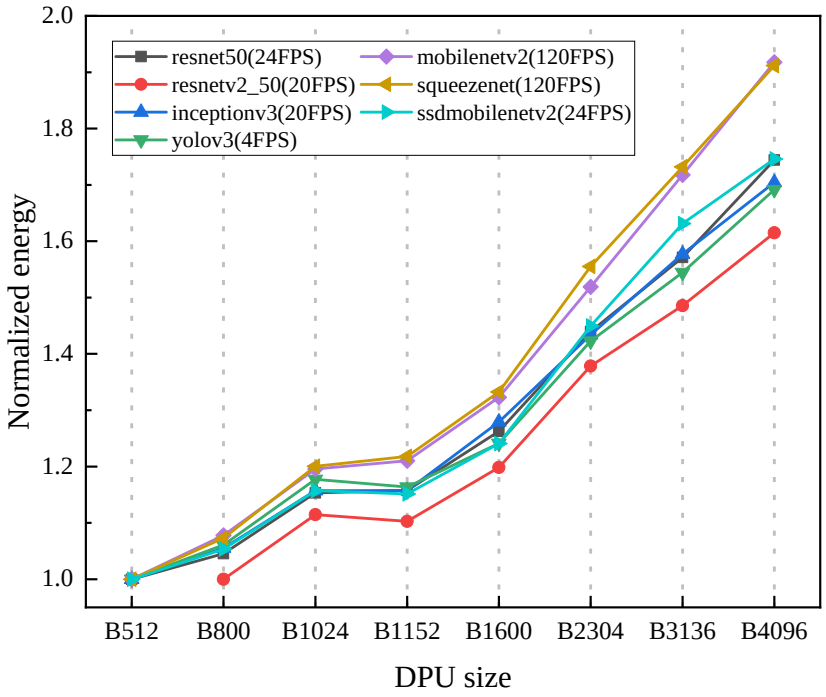

Figure 6. The energy consumption trend of some CNN for different DPU sizes.

5.3. Reconfigurable Scheme

According to the analysis of power and resources in the previous two sections, the network processing power increases with the increase of the DPU size for the real-time processing task scenario. Furthermore, the DSP minimum principle is used to reduce resource consumption. Therefore, a reconfiguration scheme is proposed based on the previous runtime prediction model in Figure 8. The reconfiguration scheme follows the design principle of greedy algorithms, and the steps are as follows:

1. For any CNN to be deployed, the prediction model is used to obtain its runtime under various configurations;
2. $RT(i,j)(0 \leq i \leq 7, j = 1,2,3)$ denotes the predicted runtime of the CNN to be run on a specific DPU configuration. Circuits that satisfy the reconfiguration requirements are laid out in a staircase pattern in Figure 7;
3. According to the previous optimal energy consumption reconfiguration scheme, the scheme first select the DPU configuration at the outermost edge of the ladder as a candidate (marked in red and blue in the figure);
4. The resources used by candidate circuits are then compared. The strategy can select the final reconfiguration circuit based on the principle of least DSP.

Through the above four steps, the circuit to be deployed can be quickly found by combining the low-complexity prediction model and the optimization strategy. Firstly, the prediction model can avoid the repeated trial and error process when deploying new algorithm models, which significantly improves the efficiency of reconfiguration. Moreover,

according to the proven characteristics of DPU, the greedy algorithm is chosen instead of a global comparison algorithm, which further reduces the time to compare data during reconfiguration and ensures a higher hit probability of the optimal solution. In 100 times verification, the optimal solution rate of the reconfiguration scheme reaches 97%.

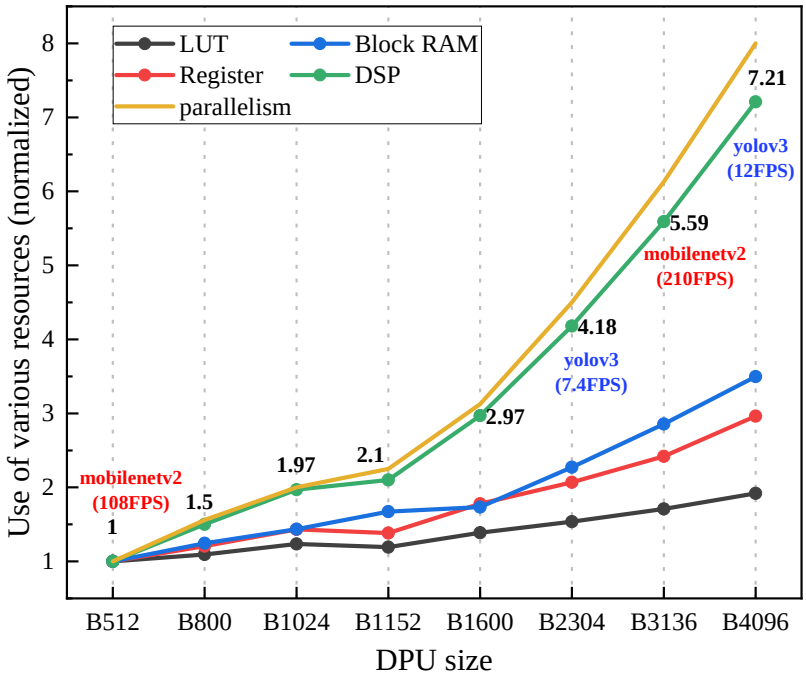

Figure 7. Normalized resource consumption of eight DPUs.

	···	$Size_{i-1}$	$Size_i$	$Size_{i+1}$	···
Single	···	*RT*(i-1,1)	*RT*(i,1)	***RT*(i+1,1)**	···
Dual	···	*RT*(i-1,2)	***RT*(i,2)**	***RT*(i+1,2)**	···
Triple	···	***RT*(i-1,3)**	***RT*(i,3)**	***RT*(i+1,3)**	···

Figure 8. Schematic diagram of the reconfiguration scheme.

In addition, using dynamic reconfiguration technology is beneficial to minimize the waste of hardware resources. There are many operators with low usage in CNNs, such as element-wise dot product, average pooling, Leaky ReLU, and so on. These modules remain idle in the circuit most of the runtime and increase the energy and resource consumption of the system. For the same size of DPU, the max and compact modes should be set. Max mode contains all operators, while compact mode contains only the common operators. For example, ResNet50 does not contain computation operations such as depthwise separable convolution, Leaky ReLU, and element-wise dot product. These operators can be removed at runtime. As can be seen from Table 3, the performance of the two modes on a B2304 DPU is almost the same. Nevertheless, the compact mode greatly reduces the hardware resource consumption.

Table 3. Comparison of resource utilization between max and compact modes on B2304 DPU.

Arch	B2304 (Max)	B2304 (Compact)
LUT (Utilization)	84,041 (31.28%)	69,856 (26.01%)
REG (Utilization)	139,222 (25.79%)	126,754 (23.48%)
BRAM (Utilization)	422 (47.15%)	322 (35.98%)
DSP (Utilization)	844 (33.49%)	580 (23.02%)
Performance (FPS)	45.34	44.98

6. Evaluation and Results

The first part of the section describes the data processing and prediction results of the random forest model. The second part shows the establishment of the adaptive system. The third part describes the performance of the system in an ADAS scenario.

6.1. Model Training and Prediction

At present, the DPU platform supports a total of eight architectures. CNNs running under configurations of B800, B1152, B2304, and B4096 are selected as the training set. A variety of CNNs is divided into standard convolution layers and depthwise separable convolution layers with different parameters. Based on Xilinx ZCU102, we measured the runtime of all layers with the Vitis profiler and took the average of 50 runs as the final data to form a dataset with 2000 samples. A test set composed of 12 CNNs is used to evaluate the prediction performance after model training.

Figure 9 shows the prediction error for all CNN layers among different DPU sizes. Because the maximum error may contain some anomalies, its reference significance is low and the maximum error is not used as a metric. We count the mean error and median error. It can be seen that the maximum value of the mean error is 14.9% on the B512 DPU. The runtime of a CNN is accumulated by the prediction results of multiple layers. Therefore, when considering the prediction results of a single CNN, the prediction error is significantly reduced, which is what we hope to see. The mean prediction error of CNN is between 2% and 5%.

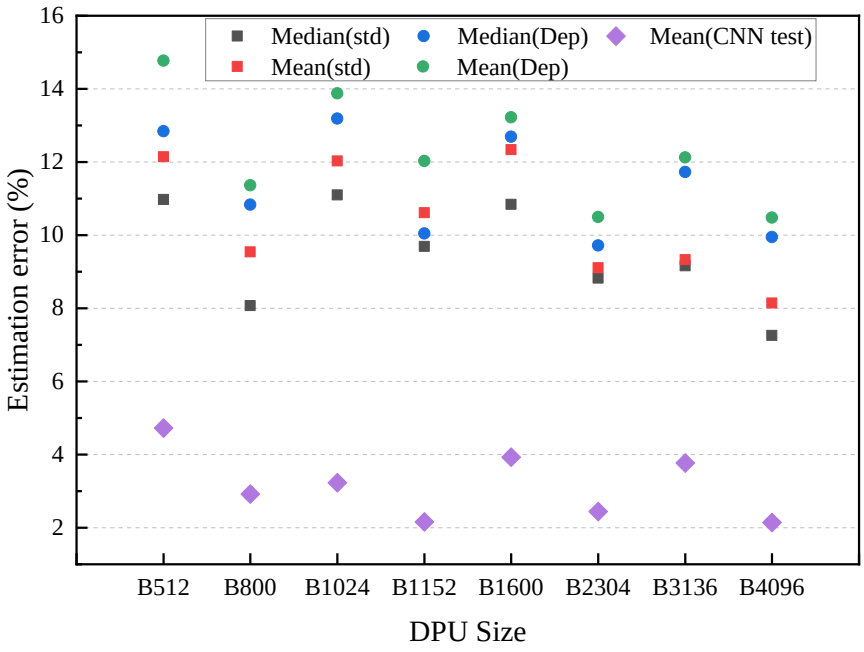

Figure 9. Single-layer prediction error for different DPU and the mean error of various CNNs.

Twelve CNNs for image classification and object detection are used as a test set to comprehensively evaluate our proposed model. Table 4 shows the basic parameters of each

CNN, including the number of layers, the number of MAC operations, and the size of MEM. Figure 10 shows the results of the prediction model for 12 CNNs in 2 DPU operation modes, B1024 and B4096. The execution time for different CNNs varies significantly, ranging from 4.7 ms to 82.8 ms for B4096 DPU and from 7.2 ms to 279.5 ms for B1024 DPU.

Table 4 also lists the CNN prediction error under B1024 and B4096 DPU. In the case of B4096 DPU, the prediction accuracy is basically maintained at 3% or less. Although the convolution layer prediction error in B1024 DPU increases slightly, the overall prediction error is less than 5%, which fully realizes the acceptable prediction results. Using a random forest model to train and predict the required convolution network can get satisfactory results, and it can be used as the basis of the reconfiguration scheme. Because the two algorithms are modeled separately, the maximum error is effectively reduced. For the test results under eight DPUs, the maximum error is 9.27%. Because the DPU of B500, B1024, B1600, and B3136 is the test set, the performance of the prediction model in these four sizes is slightly worse than the others. All in all, the prediction performance of the regression model after training is overall satisfactory.

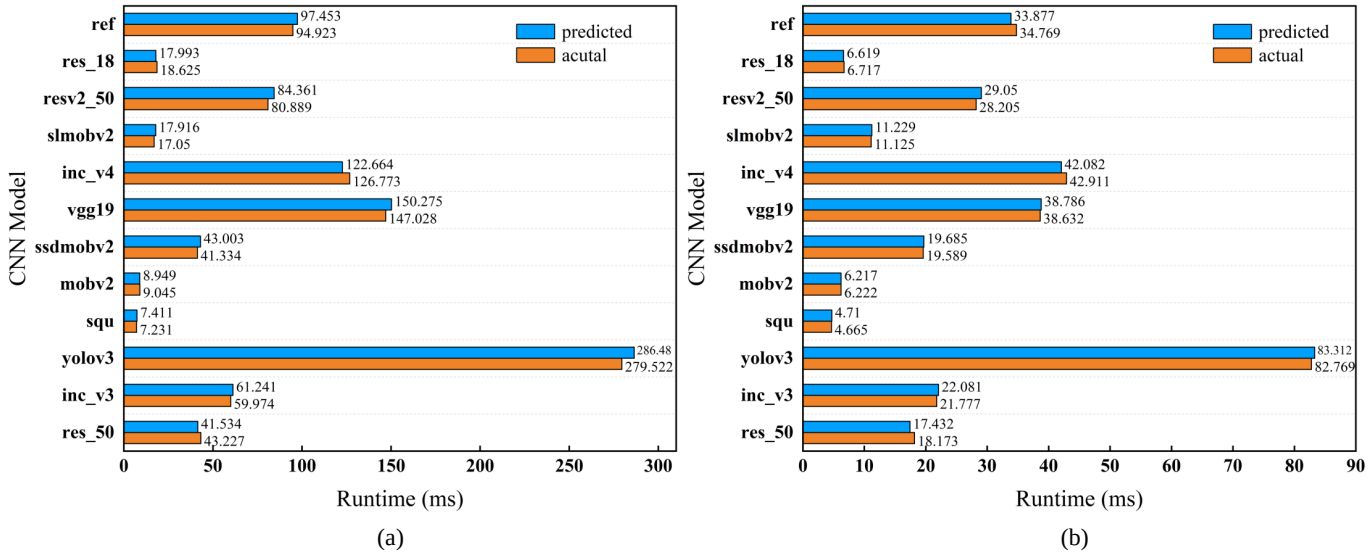

Figure 10. (**a**) Comparison of the predicted and actual runtime of 12 CNNs at B1024 DPU (**b**) Comparison of the predicted and actual runtime of 12 CNNs at B4096 DPU.

Table 4. Prediction results of the different CNN models.

CNN Name	MAC Ops ($\times 10^9$)	Data Required (MB)	# Layers	Prediction Error (B4096)	Prediction Error (B1024)
refinedet (ref)	10.12	26.44	48	2.57%	2.67%
resnet18 (res_18)	3.655	17.23	21	1.46%	3.39%
resnetv2_50 (resv2_50)	13.09	82.18	70	2.98%	4.29%
ssdlite_mobnetv2 (slmobv2)	1.50	18.85	61	0.93%	5.08%
inception_v4 (inc_v4)	24.54	101.25	166	1.93%	3.24%
vgg19 (vgg19)	39.27	172.98	19	0.40%	2.21%
ssd_mobnetv2 (ssdmobv2)	6.53	44.76	50	0.49%	4.04%
mobilenet_v2 (mobv2)	0.60	9.68	36	0.16%	1.06%
squeezenet (squ)	0.78	4.85	26	0.96%	2.49%
yolo_v3 (yolov3)	65.43	162.35	77	0.66%	2.49%
inception_v3 (inc_v3)	11.44	50.71	105	1.39%	2.11%
resnet50 (res_50)	7.72	51.25	59	4.08%	3.92%

6.2. Implementation of Adaptive Reconfiguration System

The schematic diagram of the adaptive system is shown in Figure 11. Our regression model is executed on an ARM CPU core available on Xilinx ZCU102 board, and the average

cost of our model is 1.8 ms for a CNN with about 30 layers. This time is negligible compared to the actual runtime and switching cycles of CNNs in the application. In addition, the prediction occurs on the CPU core and does not affect the execution of CNNs on DPUs. Therefore, our prediction model applies to both design time and runtime.

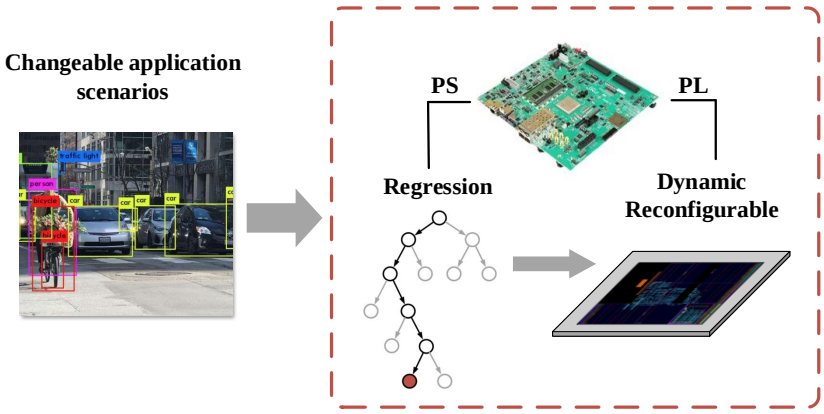

Figure 11. Schematic diagram of adaptive reconfiguration system implementation. The bold circle represents the process of a random forest regression. The brown circle indicates the final regression result.

The compiled xmodel file contains model parameters and model instructions. When the DPU hardware part is updated, the xmodel needs to be updated to match different DPU structures at the same time. Different xmodel files are pre-stored on the FPGA and the optimal DPU configuration is obtained through the reconfiguration scheme. Then the system loads the corresponding xmodel after configuring the DPU. In this way, Xilinx runtime (XRT) can transfer the instructions and data required at runtime to DDR according to xmodel. After that, the DPU is driven to start working by writing the instruction stream address in the control register of it. The DPU reads data and instructions from the memory through the INSTR bus to complete the inference of networks.

Through the three dimensions of DPU architecture (8), DPU number (3), and operator mode (2), 48 different configurations can be formed. In these configurations, the FPGA idle power consumption range is 1.45 W–5.49 W, the onboard DSP usage range is 110–1380, and the power consumption range of running ResNet50 is 2.25 W–12.41 W. The dynamic adjustment range is extensive.

6.3. Dynamic Reconfiguration in ADAS Scenario

A real-time inference scenario is assumed to test our system, including two CNNs often used in ADAS: Vpgnet [27] and Refinedet [28]. Vpgnet is a network for lane line and road sign detection and recognition, and the network used in this paper is its pruned version. Refinedet is an object detection network that can be used for pedestrian detection. There are two scenes: the vehicle running at low speed and high speed. In low-speed scenarios, Vpgnet and Refinedet run at 96 FPS and 32 FPS. Both are 180 FPS and 60 FPS in high-speed scenarios. The reconfiguration method is applied to the two algorithms, respectively.

The experimental results are shown in Table 5. Vpgenet at 96 FPS uses the B1024 dual-core configuration and chooses the compact mode. In this configuration, the measured power consumption is 4.95 W. Refindet requires a small amount of computation, and the single-core B1024 suffices. However, Refinedet contains element-wise dot product and transposed convolution operations, so max mode was selected. The working power consumption is 4.86 W. In the case of high load, Vpgnet uses the dual-core B3136 DPU, and configures the compact mode as in the case of low load. Refinedet is optimally configured by the reconfiguration scheme to include the max mode of the triple-core B512. The power consumption of the two is 7.17 W and 6.24 W, respectively. The advantage of the

reconfiguration scheme is that it can accurately and quickly find the appropriate circuit in only one run, avoiding the redundant, time-consuming compilation and selection processes.

Table 5 also presents the suboptimal solution for the adaptive reconfiguration scheme and the benchmark scheme for the fixed DPU configuration. The three-core B1152 DPU and three-core B4096 DPU are considered as the baseline in two scenarios for non-reconfiguration consideration. All three schemes meet the requirements of real-time processing at the current frame rate. By comparing the above three situations, it can be seen that the circuit obtained by the adaptive reconfiguration scheme has the lowest power consumption. In addition, the selected configuration is also the most resource-efficient because of the resource-minimum reconfiguration strategy.

Table 5. Results based on proposed reconfiguration scheme. The networks all run at the frame rate in parentheses, and the power consumption is the average of multiple measurements during real-time processing.

CNN Model (FPS)	Configuration	DPU Size	#Cores	Mode	Power (W)
Vpgnet (96)	most optimal	B1024	2	compact	4.95
	suboptimal	B1152	2	compact	5.07
	baseline	B1152	3	max	6.32
Vpgnet (180)	most optimal	B3136	2	compact	7.17
	suboptimal	B4096	2	compact	9.25
	baseline	B4096	3	max	12.63
Refinedet (32)	most optimal	B1024	1	max	4.86
	suboptimal	B1152	1	max	5.06
	baseline	B1152	3	max	6.93
Refinedet (60)	most optimal	B512	3	max	6.24
	suboptimal	B1024	2	max	6.44
	basline	B4096	3	max	14.54

7. Conclusions and Future Works

Through modeling and experiments in actual scenarios, the feasibility of adaptive dynamic reconfiguration based on Xilinx DPU has been proved to explore the reconfiguration of FPGAs under various CNN deployments in the future and alleviate the power consumption caused by high computing power questions. The proposed prediction model and reconfiguration scheme achieve an accuracy of 90.7%, which provides guidance for resource-saving and energy consumption reduction for specific CNN deployments. Through the experimental data in the simulation scenario, the reliability of our adaptive system is verified, and the resources and power consumption are significantly optimized. At the same time, our research can be effectively extended to other FPGA refactoring designs similar to CNN accelerators as a methodological basis for adaptive reconfiguration. In the future, we will continue to explore efficient reconfigurable systems and develop separate hardware architectures for standard convolution and depthwise separable convolution. We plan to further solve the compatibility of depthwise separable convolution through data stream reconfiguration so as to achieve more a efficient software/hardware co-design of dynamic reconstructed circuits.

Author Contributions: Conceptualization, K.H.; methodology, K.H. and Y.L.; software, Y.L.; validation, Y.L.; formal analysis, Y.L.; investigation, K.H. and Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, K.H. and Y.L.; visualization, Y.L.; project administration, K.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Beijing Natural Science Foundation grant number 4222072 and National Key Research & Development Program of China grant number 2020YFC1511702.

Data Availability Statement: Not applicable.

Acknowledgments: The research is supported by School of Electronic Engineering, Beijing University of Posts and Telecommunications. Xilinx Zynq UltraScale+ MPSoC ZCU102 used in this paper is sponsored by Beijing Natural Science Foundation.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
FPGA	Field Programmable Gate Array
ADAS	Advanced Driving Assistance System
FPS	Frame Per Second
DDR	Double Data Rate
XRT	Xilinx Runtime
DPU	Deep Learning Processing Unit

References

1. Shi, J.; Tian, X.; Zheng, Z.; Zhang, T. Application Research of CNN Accelerator Design Based on FPGA in ADAS. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *768*, 072014. [CrossRef]
2. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
3. Xilinx. DPUCZDX8G for Zynq UltraScale+MPSoCs Product Guide (PG338). 2021. Available online: https://docs.xilinx.com/r/en-US/pg338-dpu (accessed on 25 September 2022).
4. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
5. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
6. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
7. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
8. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
9. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
10. Xilinx. Vivado Design Suite User Guide: Dynamic Function eXchange (UG909). 2022. Available online: https://docs.xilinx.com/r/en-US/ug909-vivado-partial-reconfiguration/Introduction (accessed on 25 September 2022).
11. Chen, Y.H.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid-State Circuits* **2016**, *52*, 127–138. [CrossRef]
12. Lym, S.; Erez, M. Flexsa: Flexible systolic array architecture for efficient pruned dnn model training. *arXiv* **2020**, arXiv:2004.13027.
13. Zhang, S.; Cao, J.; Zhang, Q.; Zhang, Q.; Zhang, Y.; Wang, Y. An fpga-based reconfigurable cnn accelerator for yolo. In Proceedings of the 2020 IEEE 3rd International Conference on Electronics Technology (ICET), Chengdu, China, 8–12 May 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 74–78.
14. Liu, X.; Yang, J.; Zou, C.; Chen, Q.; Yan, X.; Chen, Y.; Cai, C. Collaborative Edge Computing With FPGA-Based CNN Accelerators for Energy-Efficient and Time-Aware Face Tracking System. *IEEE Trans. Comput. Soc. Syst.* **2022**, *9*, 252–266. [CrossRef]
15. 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, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
16. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1314–1324.
17. Ryu, S.; Oh, Y.; Kim, J.J. Mobileware: A High-Performance MobileNet Accelerator with Channel Stationary Dataflow. In Proceedings of the 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), Munich, Germany, 1–4 November 2021; pp. 1–9.
18. Kedia, R.; Goel, S.; Balakrishnan, M.; Paul, K.; Sen, R. Design Space Exploration of FPGA-Based System With Multiple DNN Accelerators. *IEEE Embed. Syst. Lett.* **2021**, *13*, 114–117. [CrossRef]

19. Goel, S.; Kedia, R.; Balakrishnan, M.; Sen, R. INFER: INterFerence-aware Estimation of Runtime for Concurrent CNN Execution on DPUs. In Proceedings of the 2020 International Conference on Field-Programmable Technology (ICFPT), Maui, HI, USA, 9–11 December 2020; pp. 66–71.
20. Lei, Y.; Deng, Q.; Long, S.; Liu, S.; Oh, S. An Effective Design to Improve the Efficiency of DPUs on FPGA. In Proceedings of the 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), Hong Kong, China, 2–4 December 2020; pp. 206–213.
21. Shahshahani, M.; Bhatia, D. Resource and Performance Estimation for CNN Models using Machine Learning. In Proceedings of the 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Tampa, FL, USA, 7–9 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 43–48.
22. Yin, J.; Yang, J.; Yang, H.; Du, Z. A CNN accelerator on embedded FPGA using dynamic reconfigurable coprocessor. In Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing, Sanya, China, 19–21 December 2019; pp. 1–5.
23. Gong, L.; Wang, C.; Li, X.; Zhou, X. Improving HW/SW Adaptability for Accelerating CNNs on FPGAs Through A Dynamic/Static Co-Reconfiguration Approach. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1854–1865. [CrossRef]
24. Irmak, H.; Ziener, D.; Alachiotis, N. Increasing Flexibility of FPGA-Based CNN Accelerators with Dynamic Partial Reconfiguration. In Proceedings of the 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), Dresden, Germany, 30 August–3 September 2021; pp. 306–311.
25. Xilinx. Vitis AI User Guide (UG1414). 2022. Available online: https://docs.xilinx.com/r/en-US/ug1414-vitis-ai (accessed on 25 September 2022).
26. Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; Cong, J. Optimizing fpga-based accelerator design for deep convolutional neural networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2015; pp. 161–170.
27. Lee, S.; Kim, J.; Shin Yoon, J.; Shin, S.; Bailo, O.; Kim, N.; Lee, T.H.; Seok Hong, H.; Han, S.H.; So Kweon, I. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 1947–1955.
28. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-shot refinement neural network for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4203–4212.