

Article

POSS-CNN: An Automatically Generated Convolutional Neural Network with Precision and Operation Separable Structure Aiming at Target Recognition and Detection

Jia Hou¹, Jingyu Zhang¹, Qi Chen¹, Siwei Xiang¹, Yishuo Meng¹, Jianfei Wang¹, Cimang Lu² and Chen Yang^{1,*} 

¹ School of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, China; hjwst0314@stu.xjtu.edu.cn (J.H.); zjingyu@stu.xjtu.edu.cn (J.Z.); qchen20134728@stu.xjtu.edu.cn (Q.C.); X1004312107@stu.xjtu.edu.cn (S.X.); 3121156027@stu.xjtu.edu.cn (Y.M.); jfwang@stu.xjtu.edu.cn (J.W.)

² Shenzhen Xinrai Sinovoice Technology Co., Ltd., Shenzhen 518000, China; cimang.lu@xinrai.com

* Correspondence: chyang00@xjtu.edu.cn; Tel.: +86-13810103039

Abstract: Artificial intelligence is changing and influencing our world. As one of the main algorithms in the field of artificial intelligence, convolutional neural networks (CNNs) have developed rapidly in recent years. Especially after the emergence of NASNet, CNNs have gradually pushed the idea of AutoML to the public's attention, and large numbers of new structures designed by automatic searches are appearing. These networks are usually based on reinforcement learning and evolutionary learning algorithms. However, sometimes, the blocks of these networks are complex, and there is no small model for simpler tasks. Therefore, this paper proposes POSS-CNN aiming at target recognition and detection, which employs a multi-branch CNN structure with PSNC and a method of automatic parallel selection for super parameters based on a multi-branch CNN structure. Moreover, POSS-CNN can be broken up. By choosing a single branch or the combination of two branches as the "benchmark", as well as the overall POSS-CNN, we can achieve seven models with different precision and operations. The test accuracy of POSS-CNN for a recognition task tested on a CIFAR10 dataset can reach 86.4%, which is equivalent to AlexNet and VggNet, but the operation and parameters of the whole model in this paper are 45.9% and 45.8% of AlexNet, and 29.5% and 29.4% of VggNet. The mAP of POSS-CNN for a detection task tested on the LSVH dataset is 45.8, inferior to the 62.3 of YOLOv3. However, compared with YOLOv3, the operation and parameters of the model in this paper are reduced by 57.4% and 15.6%, respectively. After being accelerated by WRA, POSS-CNN for a detection task tested on an LSVH dataset can achieve 27 fps, and the energy efficiency is 0.42 J/f, which is 5 times and 96.6 times better than GPU 2080Ti in performance and energy efficiency, respectively.

Keywords: target recognition; target detection; precision and operation separable structure; multi-branch; automatic search for super parameters



Citation: Hou, J.; Zhang, J.; Chen, Q.; Xiang, S.; Meng, Y.; Wang, J.; Lu, C.; Yang, C. POSS-CNN: An Automatically Generated Convolutional Neural Network with Precision and Operation Separable Structure Aiming at Target Recognition and Detection. *Information* **2023**, *14*, 604. <https://doi.org/10.3390/info14110604>

Academic Editors: Nikolaos Mitianoudis and Ilias Theodorakopoulos

Received: 22 September 2023

Revised: 17 October 2023

Accepted: 2 November 2023

Published: 7 November 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As one of the significant algorithms in the field of artificial intelligence, convolutional neural networks (CNNs) combined with the application of target recognition and detection are in the ascendant, such as NASNet [1], FractalNet [2], CFP [3], FEDNet [4], MCRF-ResNets [5], EfficientNet [6], ViT [7], NFNet [8], AmoebaNet [9], ENASNet [10], etc. [11–13].

EfficientNet [6] is a novel network architecture proposed by Google in 2019, which scales the depth, width, and resolution using a simple yet highly effective compound coefficient, thereby balancing these three dimensions to lead to better performance. Experimental results indicate that EfficientNet achieves better accuracy on various datasets, including ImageNet, CIFAR-100, Flowers, and three other transfer learning datasets. Vision Transformer (ViT) [7] is proposed for computer vision tasks by applying the principles of the Transformer model. The design method of ViT is by splitting images into patches of fixed size, linearly embedding each of them, and then feeding the resulting sequence of

vectors into a transformer structure. Experiments show that ViT attains excellent performance when pre-trained on ImageNet, CIFAR-100, and VTAB, while requiring substantially fewer computational resources. Normalized Free Network (NFNet) [8] adapts an adaptive gradient clipping technique instead of batch normalization, which is designed to address the issue regarding the training of large neural networks. The evaluation results show that the proposed model is up to $8.7\times$ faster to train along with matching the test accuracy of EfficientNet [6] on ImageNet. For target detection and recognition tasks, there are many different research methods, among which, branch method is used to further improve the accuracy of target recognition. In addition, after the appearance of NASNet [1], people began to focus on AutoML and there are also more and more CNNs generated by AutoML technology for target detection and recognition tasks. NASNet can automatically search for blocks and generate a high-performance CNN structure based on specific datasets. Like NASNet, most of the models generated from automatic searches are based on reinforcement learning and evolutionary learning algorithms. AmoebaNet [9] acquired the models by improving the evolutionary algorithm. ENASNet [10] and NAONet [14] obtained more efficient neural search architecture by optimizing the neural network structure. For the branch technology, FractalNet [2] adopted a branch structure and “nontrivial routing” technique, which is a classical model based on an improved widening width method. Moreover, in the field of target detection, Mask R-CNN [15] added a branch based on Faster R-CNN [16] to achieve a multi-task mode for target detection and instance segmentation.

However, the blocks generated by AutoML technology are sometimes relatively complex and there are no suitable small models for simple tasks, except for automatic searching. Moreover, most of the models generated by a branch structure are aimed at pursuing high precision, which may lead to an increase in parameters and operation. When implemented on hardware architecture, the transmission delay of the data process and the power consumption of memory access will rise. Meanwhile, an enormous operation will significantly increase the inference time and energy consumption when the computing power of hardware architecture is fixed. As shown in Figure 1, with the increase in energy efficiency, the size of memory access increases gradually, which is almost 1000 times the Multiple-Add operation [17], and, unfortunately, the proportion of memory access and ALU-operation has increased in the recent years [18].

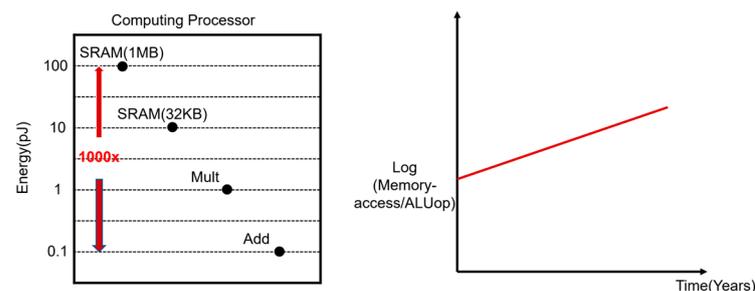


Figure 1. Memory access and memory operation.

To solve the aforementioned issues, this paper tries to find a balance between precision and operation. Combined with “branch” and “automatic searching” technologies, a CNN architecture of “one network with multi-model” is designed, so that the multi-model can be obtained by one training. Based on different recognition and detection tasks, firstly, initialize a multi-branch CNN model with no layer. Secondly, train and search for building blocks automatically. Whenever the training iteration ends, select the building blocks with excellent performance, and add to each branch in parallel and iteratively until POSS-CNN is generated. POSS-CNN can be broken up; if we take a single branch or the combination of two branches as the “benchmark”, as well as the complete POSS-CNN, we can generate a total of seven CNN models with different precision and operations. Moreover, POSS-CNN is convenient for hardware implementation and can be applied to different performance hardware platforms at the same time. The contributions of this paper are as follows:

- (1) A design method of a multi-branch CNN structure is proposed in this paper. As mentioned above, we can obtain seven CNN models by one-time training with different branching modes. Compared with other schemes that only generate models with a fixed precision, the method proposed in this paper can improve efficiency and save time.
- (2) A methodology of automatic parallel selection for super parameters based on a multi-branch CNN structure is proposed, which makes the generation process of a multi-branch CNN structure with separable precision and operation automatic and procedural.
- (3) POSS-CNN: An automatically generated convolutional neural network with a precision and operation separable structure aimed at recognition and detection. A parameter tuning method named Parameters Substitution with Nodes Compensation (PSNC) reduces the parameters of the network model.

The rest of this paper is organized as follows. After reviewing relevant literature and stating the motivation in Section 2, Section 3 mainly introduces a new POSS-CNN architecture. The design of a multi-branch CNN structure with separable precision and operation is introduced in Section 3.1, the methodology of automatic parallel selection for super parameters for a multi-branch CNN is described in Section 3.2, and an architecture overview is shown in Section 3.3. Experimental results and discussions are included in Section 4, followed by the Discussion and Conclusions in Sections 5 and 6, respectively.

2. Related Work and Motivation

2.1. Related Work

The development history of target recognition in recent years is mostly based on the development track of CNN, as listed in Table 1. LeNet [19] is the cornerstone of 2D CNN. AlexNet [20] improved the model structure of LeNet, which raised the recognition image resolution to $224 \times 224 \times 3$. AlexNet adopted five convolutional layers, while only adding a pooling layer after the first, second, and fifth convolutional layers. VggNet [21] adopted a 3×3 kernel size to form 16-layer and 19-layer CNN structures. The error rates of Top-1 and Top-5 of VggNet tested on ImageNet are 24.4% and 7.1%, respectively. ResNet [22] proposed residual blocks and skip connection methods to stack into a depth structure. DenseNet [23] employed a technique in which each input layer receives all previous output layers to obtain more characteristic information.

Table 1. Summary of CNN mainstream models.

CNN	Top-1	Top-5	Parameters (Million)	Operation (MFlops)
AlexNet	37.5%	17%	60	720
VggNet	24.4%	7.1%	138	15,300
ResNet-152	21.43%	5.71%	62	23,000
DenseNet	20.8%	5.29%	34	12,000

Target detection tasks not only identify where the kind of target belongs but also detect the relative position of the target in the image or video. In 2012, R-CNN [24] employed CNN and a region extraction algorithm to achieve the supervised target detection tasks, which is a milestone in the development of deep learning for target detection. In 2014, SPPNet [25] proposed a “pyramid structure” of pooling layers based on R-CNN, which improved the inference speed of R-CNN. In 2015, Fast R-CNN [26] used the minimum loss function to finetune the “confidence value” and “bounding box” of all layers to improve the inference speed of R-CNN. Faster R-CNN [16] directly proposed an RPN (Region Extraction Network) and “anchor” mechanism based on R-CNN, which is equivalent to achieving target detection by using two networks. Faster R-CNN obtained 73.2% mAP and 7 fps on a Pascal VOC dataset. YOLO [27] achieved the fastest speed with only one network, which obtained 63.4% mAP on an Ascal VOC dataset, but its speed was

only 45 fps. YOLO9000 [28] achieved 78.6% mAP and 40 fps on a Pascal VOC dataset. YOLOv3 [29] achieved excellent performance in both accuracy and speed. Meanwhile, models generated from automatic searching have also achieved good performance in the application of target detection.

In general, the automatic design of CNN applied to simple tasks often depends on the method of random search and grid search to select the super parameters. A Bayesian optimization search algorithm [30] is more intelligent than the grid search and random search methods. The Particle swarm optimization algorithm successfully applied a meta-heuristic search algorithm [31], which designed a new structure search design method. Y. Sun, et al. used an evolutionary learning algorithm to search and design a CNN [32]. Differential evolution [33] and harmonic search [34] are other meta-heuristic algorithms. Reinforcement learning [35,36] can automatically generate a high-performance CNN architecture for a given task. NASNet [1] is a classical algorithm for automatic searches based on reinforcement, which makes adjustments and optimization progress automatically. The principle of NASNet is to provide search space, and use strategies to iteratively search out the network structure with the best performance. NASNet finally used 500 Tesla P100 GPUs to train for more than 4 days, and obtained a series of excellent network structures with good performance, with only 2.4% test error rate on the CIFAR10 dataset, even surpassing a series of excellent networks such as GoLeNet [37], ResNet [22], and MobileNet [38]. The best structure of NASNet tested on the ImageNet dataset can obtain a respective 82.7% and 96.2% of Top-1 and Top-5 in accuracy, which reduces the operation amount by 28% compared with the most advanced prior model. Moreover, NASNet can reach 43.1% mAP on a COCO dataset. Moreover, ENASNet [10] tested on a CIFAR10 dataset can obtain an error rate of 2.89%. This performance is slightly higher than NASNet, but the search time is only 11.2% of NASNet. As shown in Table 2, NAONet [14] tested on a CIFAR10 dataset can obtain an error rate of 2.93%, which is slightly lower than NASNet, but it only needs 200 GPUs to train for 0.3 days. AmoebaNet [9] tested on a CIFAR10 dataset can obtain an error rate of 3.34%, as shown in Table 2, but the parameters are only 11.6% of NASNet. PNASNet [39] tested on a CIFAR10 dataset can obtain an error rate of 3.41%, as shown in Table 2, but the parameters are only 11.6% of NASNet, and the operation is 21 times less than NASNet. BlockQNN [40] tested on a CIFAR10 dataset can obtain an error rate of 3.54%, which is lower than NASNet in Table 2, but it only takes 32 GPUs to search for 3 days.

Table 2. Summary of Automatic CNN models.

Network	Test Error	Parameters	GPU	Time (Days)
NasNet	2.4%	27.6	500	4
ENASNet	2.89%	4.6	1	0.45
NAONet	2.93%	2.5	200	0.3
AmoebaNet	3.34%	3.2	450	7
PNASNet	3.41%	3.2	100	-
BlockQNN	3.54%	39.8	32	3

2.2. Motivation

The first issue of the existing CNN models based on automatic searching is that building blocks are always relatively complex. As shown in Figure 2, the network structures, NASNet, NAONet, ENASNet, BlockQNN, etc., all contain a certain number of hidden layers and other network layers, the structure of the network model is relatively complex, and there are no suitable small models for simple tasks. ShuffleNet [41] structure utilizes two new operations, pointwise group convolution and channel shuffle, to greatly reduce computation cost while maintaining accuracy, but only five layers of the structure were used before the activation function in this architecture, and the complexity of this structure is less than 150 MFlops. Moreover, although these models achieve excellent performance,

their overall structure is relatively simple, with only one excellent depth structure being obtained after training.

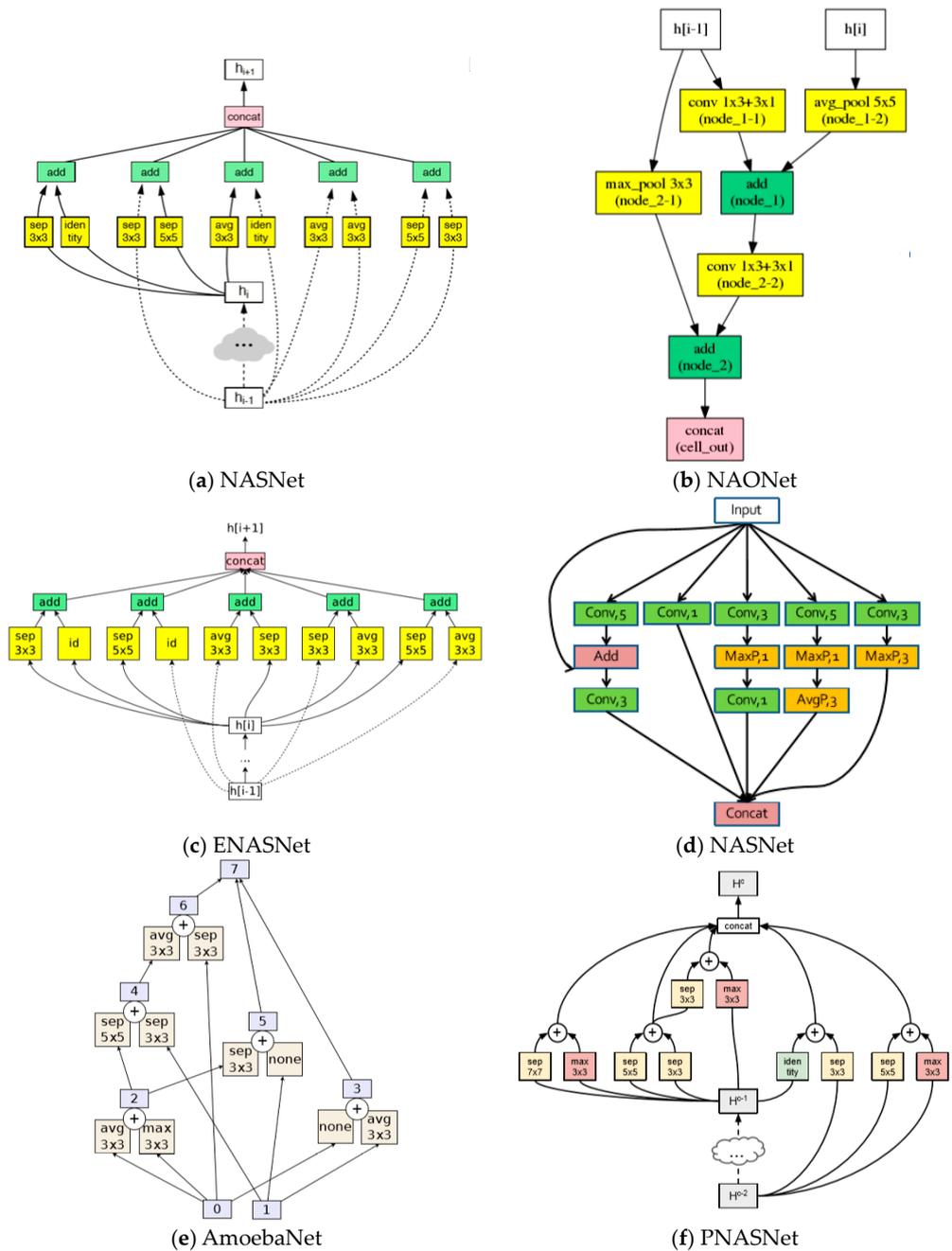


Figure 2. Blocks of automatic models.

The other issue is that the networks mentioned in related works are all selected according to the best precision performance except PNASNet. However, if only pursuing high-precision performance, it may lead to enormous parameters and operation. Large numbers of parameters may lead to an increase in the transmission delay of data processing in hardware implementation, as well as power consumption. Moreover, a large number of operations will increase inference time and energy consumption under the same computation of the hardware architecture. As shown in Figure 3, complex networks with excellent precision performance may frequently lead to huge numbers of parameters and operations, such as ResNet and ResNeXt. Even for simple models, ResNet-50 and ResNeXt-50, their operation can reach billions of times, and their parameters can reach tens of millions. Thus,

it is very difficult to achieve model migration directly on hardware processing architecture or embedded devices.

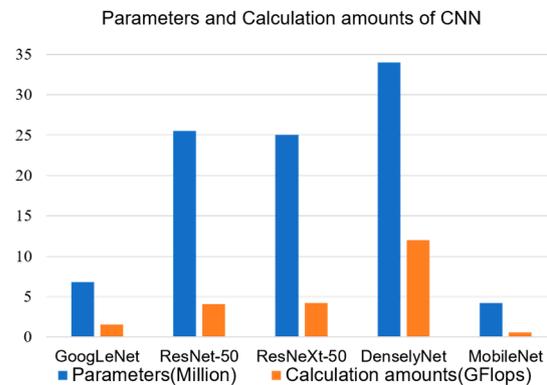


Figure 3. Parameters and Calculation amounts of CNN.

To solve the aforementioned issues, this paper tries to design a novel CNN model that can find a balance between precision and operation. Combined with the technology of “branch” and “automatic search”, we design a multi-branch CNN in order to obtain a variety of CNN models. For different tasks, initialize the multi-branch CNN, train and search for high-performance building blocks in an adaptive and automatic manner at the same time, and then add them to each parallel branch, until a high-performance multi-branch CNN is generated. This multi-branch CNN can be broken up; if we take a single branch or the combination of two branches as the “benchmark”, as well as the entire CNN, we can generate seven models with different precision and operations.

3. Materials and Methods

3.1. Multi-Branch Structure with Separable Precision and Operation

In the field of deep learning, researchers have made many attempts to explore the design of network structures for excellent-performance CNN models, including the methodology of deepening depth, widening width, and changing connection mode. In this subsection, a method using branch technology is adopted to widen the width to design a multi-branch CNN with separable precision and operation. This multi-branch CNN can be broken up.

3.1.1. Basis of Branch Structure

In 2018, FractalNet [2] proposed a new strategy based on the technology of “branch” and “nontrivial routing” algorithm. As shown in Figure 4, where z is the input vector, $F_4(z)$ is the output vector, no signal in the whole network is the privileged signal, and each input of the join connection layer is the output of the previous convolutional layer. Drop-path makes each input of the connection layer reliable and independent, so that we can choose a single column or several columns to compose high-performance sub-networks. In these sub-networks, only one path is always in a dynamic state. FractalNet can achieve 4.60% error rate on CIFAR10 dataset, which is almost equivalent to ResNet (4.62%).

Referring to FractalNet, we proposed a design method of multi-branch CNN with separable precision and operation. This multi-branch CNN can be divided into single model or combined model according to different branch patterns, as well as the complete multi-branch CNN, and we can generate a total of seven CNN models with different precision and operations. When implemented on hardware architecture, it is convenient to choose an appropriate model according to the performance of the hardware architecture by itself.

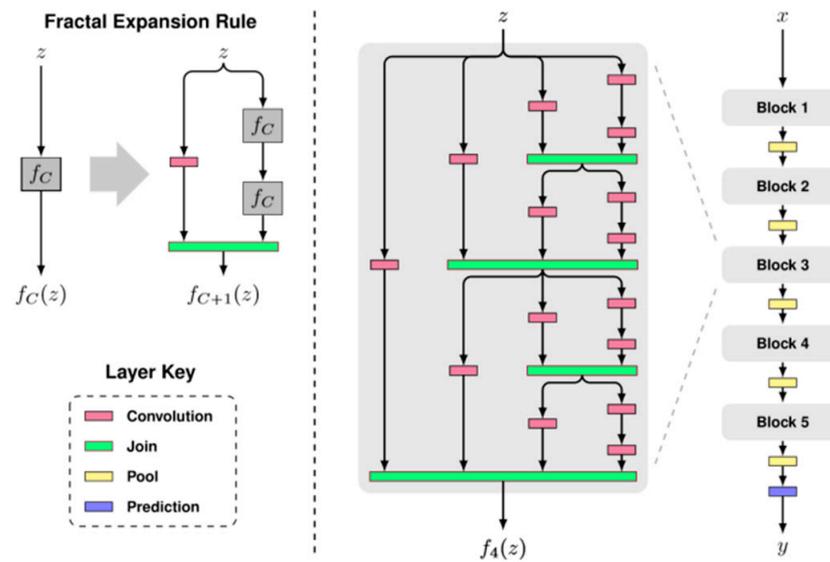


Figure 4. FractalNet.

3.1.2. Multi-Branch CNN Structure

Generally, for specific and different target recognition and detection tasks, the size of datasets with less classification and low resolution is about tens to hundreds of MB, while that of those with more classification and high resolution is about tens to hundreds of GB. Therefore, this paper lists two CNN architectures, each with three branches (branch technology can be easily extended to multi-branch or double branches). If the task dataset is less than 100 MB, select “Branch architecture one”, as shown in Figure 5a, in which the first branch is 1 convolutional layer, the second branch is 2 convolutional layers, and the third branch is 3 convolutional layers. If the dataset is larger than 100 MB, select “Branch architecture two”, as shown in Figure 5b, in which the first branch has 2 convolutional layers, the second branch has 4 convolutional layers, and the third branch has 8 convolutional layers. Both architectures have the convolutional kernel size of 3×3 with stride size of 1, as well as the channel size of the feature maps. The first branch is 16, the second branch is 32, and the third branch is 64. After each single convolutional layer, a 2×2 Max-Pooling layer is added to reduce the dimension. The number of neural nodes in the fully connected layer of the two architectures is also equal. The first branch is 256, 64, and the output categories; the second branch is 512, 128, and the output categories; and the third branch is 1024, 256, and the output categories.

Therefore, multi-branch CNN structure with separable precision and operation mainly focuses on widening the width to improve its performance. Firstly, the whole accuracy performance is improved, and the convergence speed of network training is raised. Secondly, break up the multi-branch CNN. By choosing a single branch, or the combination of two branches as the “benchmark”, as well as the complete multi-branch CNN, a total of seven models with different precision and operations can be generated. Moreover, some mobile and embedded systems from the field of “Internet of Things” or “Edge computing” need to adopt deep learning algorithms to solve complex problems, which may lead to some problems, such as energy consumption, on-chip cache, real-time operation, and short delay response. In addition, the data processing and memory operation of CNN itself account for more than 90% of the total operation [42–45]. Therefore, precision, parameters, and operation should be considered when designing CNN architecture at the level of the software algorithm. Our multi-branch CNN structure with separable precision and operation is generated by one-time training, with a relatively simple block, and the overall structure combines the evaluation of precision and operation.

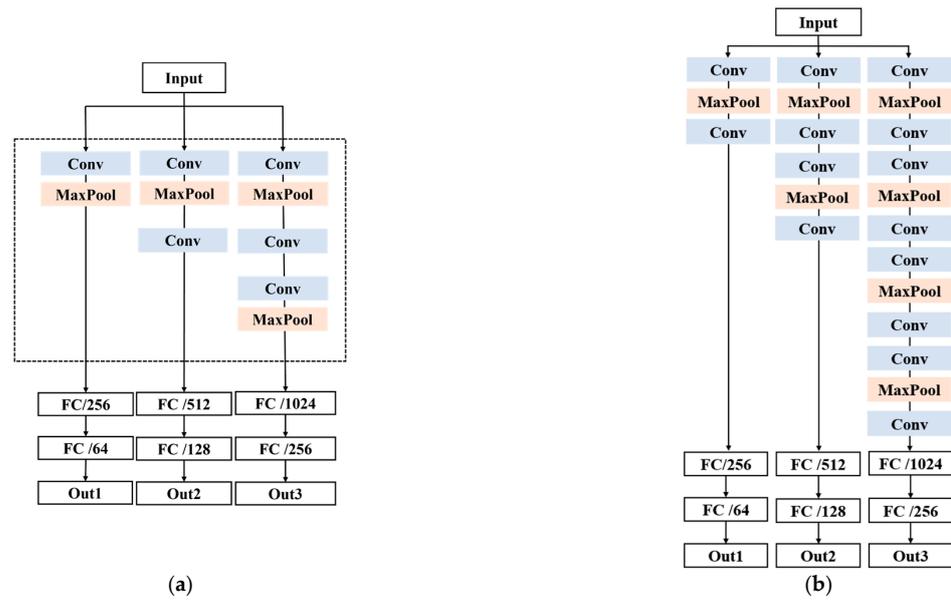


Figure 5. Multi-branch CNN structure: (a) Branch architecture one; (b) Branch architecture one.

3.1.3. Evaluation of Separable Precision and Operation

Multi-branch CNN with separable precision and operation is trained and tested on MNIST dataset. According to Section 3.1.2, “Branch architecture one” is enough.

As shown in Table 3, the accuracy of LeNet trained and tested on MNIST dataset is about 99.35% after 35,000 iterations. After employing multi-branch CNN structure with separable precision and operation, test accuracy can reach 99.5% after only 25,000 iterations. As shown in Figure 6, where Y1, Y2, and Y3 represent different branches and three branches make up the seven branch models, the precision, parameters, and operation of the models generated by different branch modes are different.

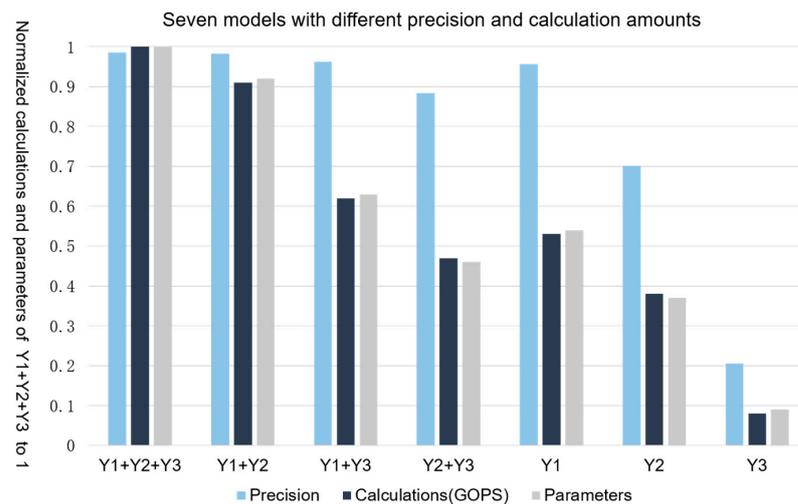


Figure 6. Seven models with different precision and operations.

Table 3. Multi-branch CNN tested on MNIST dataset.

Network	Accuracy	Iteration
LeNet	99.35%	35,000
Multi-branch CNN	99.5%	25,000

3.2. Automatic Parallel Selection for Super Parameters on Multi-Branch CNN Structure

This section puts forward a methodology of automatic parallel selection for super parameters based on multi-branch CNN, which aims at making the generation process of multi-branch CNN with separable precision and operation automatic and procedural, with the ability to generalize and suit the remedy to the case.

3.2.1. Design Flow

For specific and different target recognition and detection tasks, first, initialize a multi-branch CNN. This paper chooses a 3-branch based on experience, as shown in Figure 6. In addition, branch technology can be easily extended to a decision network with two or more branches.

Second, utilize the parallel characteristics of the branch, search blocks automatically as “Block 1”, which is usually composed of 1 convolutional layer and 1 Max-Pooling (2×2) layer, and then add them to each branch on the 3-branch CNN architecture iteratively during the training process until the 3-branch CNN architecture with the best performance is found, as shown in Figure 6. The channel size of the first branch is 16, the second branch is 32, and the third branch is 64. The super parameters of convolutional kernel size are shown in Table 4. Table 4 also makes statistics of parameters and operation, where C_i is the input channel size of the convolutional layer, C_o is the output channel size of the convolutional layer, and B is batch size.

Table 4. Super parameters of convolutional kernel size.

Kernel	Parameters	Operation
3×3	$(C_i \times 3 \times 3 + 1) \times C_o = 9C_i \times C_o + C_o$	$18 \times C_i \times C_o \times F_{ho} \times F_{wo} \times B$
5×5	$(C_i \times 5 \times 5 + 1) \times C_o = 25C_i \times C_o + C_o$	$50 \times C_i \times C_o \times F_{ho} \times F_{wo} \times B$
7×7	$(C_i \times 7 \times 7 + 1) \times C_o = 49C_i \times C_o + C_o$	$98 \times C_i \times C_o \times F_{ho} \times F_{wo} \times B$
9×9	$(C_i \times 9 \times 9 + 1) \times C_o = 81C_i \times C_o + C_o$	$168 \times C_i \times C_o \times F_{ho} \times F_{wo} \times B$
1×1	$(C_i \times 1 \times 1 + 1) \times C_o = C_i \times C_o + C_o$	$2 \times C_i \times C_o \times F_{ho} \times F_{wo} \times B$

Third, searching blocks “Blocks 2” and “Block 3” can be generated by using a similar approach to “Block 1”, and the difference lies in the parameters and operation of the network model. Obviously, 3-branch CNN architecture can also automatically adopt the greedy strategy to continue adding convolutional layers according to the complexity of the task, such as adding “Block 4” and “Block 5”. It should be noted that most of the blocks in multi-branch CNN are composed of 1 convolutional layer and 1 Max-Pooling layer, and the overall architecture is relatively neat without complex blocks, so that the first branch has 1 convolutional layer and 1 Max-Pooling layer, the second branch has 3 convolutional layers and 3 Max-Pooling layers, and the third branch has 5 convolutional layers and 5 Max-Pooling layers. It is obvious that the operation in each branch is different.

Finally, the number of neural nodes in the full-connection hidden layers is also automatically searched during the training process. The candidate set of neural nodes in the fully connected hidden layers is as follows: the first branch is {8, 16, 32, 64}, the second branch is {64, 128, 256}, and the third branch is {256, 512, 1024}.

The methodology of automatic parallel selection for super parameters for branch structures enables the design process of multi-branch CNN with separable precision and operation automatic and procedural. There are four advantages of this methodology: first, the architecture of CNN is not established at the beginning, but initializes the multi-branch architecture according to the specific and different application scenarios, and automatically and iteratively searches for the sub-network of improving performance in the super parameters’ search space to generate a complete multi-branch CNN with separable precision and operation. Second, the search space of super parameters can also be automatic according to specific and different tasks. Third, the whole search process adopts the greedy strategy, and utilizes the characteristics of a parallel multi-branch structure to obtain a complete

multi-branch CNN architecture. Finally, we can break up the multi-branch CNN and generate a total of seven models with different precision and operations by one-time training. Moreover, seven models can be applied to different performance hardware platforms at the same time.

3.2.2. Parameter Tuning of Model Optimization

A parameter tuning method named Parameters Substitution with Nodes Compensation (PSNC) [46] is proposed to reduce the parameters of the network model. The key idea of PSNC is to substitute most parameters of a fully connected layer with few parameters of a convolutional layer, and appropriately increase neuron nodes of the fully connected layer to compensate for loss of accuracy. By increasing the size of the convolution kernel, the resolution of the new output feature map will be significantly reduced compared to the resolution of the original output feature map. In this way, the resolution of the output feature map of each convolutional layer gradually decreases layer by layer when data flow through these convolutional layers, which means that few input parameters are used for the fully connected layer.

Taking LeNet tested on MINST as an example shown in Figure 7, the convolution kernels of the original LeNet-5 are $5 \times 5 \times 6$ and $5 \times 5 \times 16$, respectively. According to the PSNC method, the convolutional kernel sizes are increased to $7 \times 7 \times 6$ and $7 \times 7 \times 16$, respectively. Meanwhile, the hidden layer nodes of the fully connection layer are slightly up to 100 instead of 84. In this way, the PSNC method decreased the total parameters of LeNet-5 by 42%, from 60,850 to 35,303. Moreover, there was almost no loss in the training accuracy, from 97.6% to 97.5%.

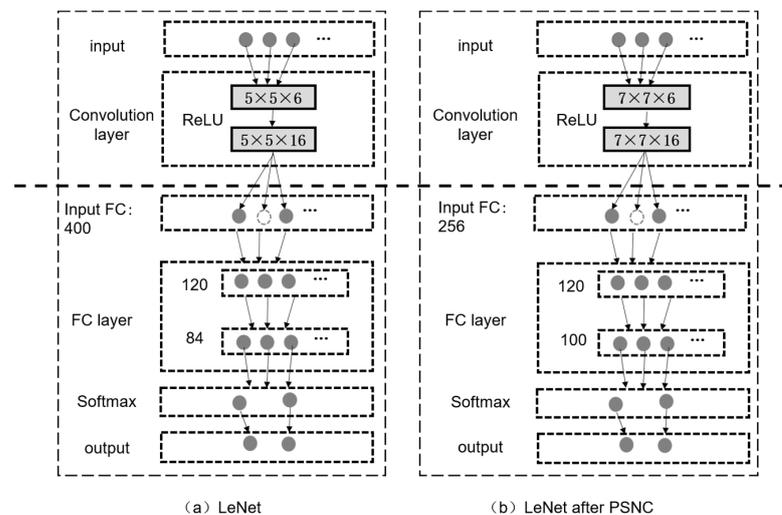


Figure 7. The comparison for LeNet before and after PSNC.

Based on the designed multi-branch CNN structure for target recognition and target detection, PSNC method is used to optimize the parameter of the network model. Table 5 lists the parameters for seven models of POSS-CNN on CIFAR10 before and after PSNC. It can be seen that the parameters of almost all models using PSNC decrease significantly. To intuitively show the advantage of PSNC, Figure 8 illustrates the percentage of parameter reduction. From this figure, compared with CNN structure before PSNC, the parameters reduce by 52.6% for model $Y1 + Y2 + Y3$, by 54.7 for model $Y3 + Y2$, by 47.6 for model $Y1 + Y3$, by 38.7% for model $Y1 + Y2$, by 55.0% for model $Y3$, and by 52.3% for model $Y2$. It is clear that it is beneficial for parameter optimization of PSNC POSS-CNN, thereby speeding the training process on CIFAR10 and improving the performance for application of target recognition.

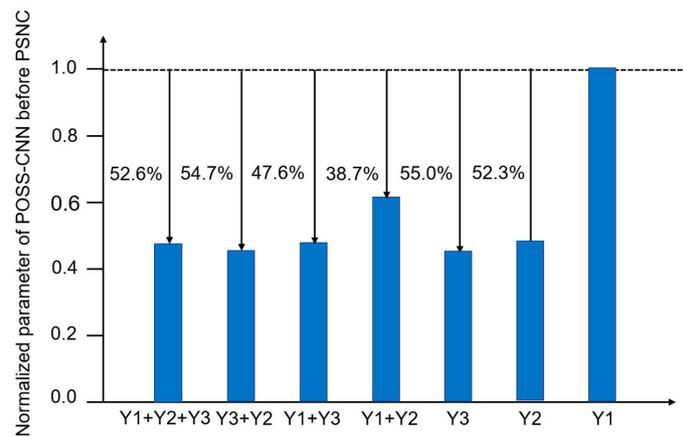


Figure 8. The percentage of parameter reduction using PSNC method.

Table 5. The parameter for seven models of POSS-CNN on CIFAR10 before and after PSNC.

CNN	Dataset	Models	Parameters before PSNC (Million)	Parameters after PSNC (Million)
POSS-CNN	CIFAR10	Y1 + Y2 + Y3	20.9	9.9
		Y3 + Y2	20.1	9.1
		Y1 + Y3	18.7	8.9
		Y1 + Y2	3.03	1.86
		Y3	17.9	8.07
		Y2	2.2	1.05
		Y1	0.81	0.81

3.2.3. Evaluation of Training Time and Accuracy

The methodology of automatic parallel selection for super parameters for the branch structure proposed in this paper is experimented on MNIST dataset. The experimental results are shown in Table 6. The accuracy of the whole architecture is 98.7%, the parameters are 231,190, and the operations are 461,334, but it only takes one hour. Compared with other search methods in Table 6, the time is just 50%, 33%, and 83% for the grid search method, random search method, and Bayesian optimization search method, respectively. The accuracy is higher than the grid search method and random search method, but slightly lower than 99% of the Bayesian optimization search method because the Bayesian optimization search method can search the model regardless of the parameters and operation, including 21 Conv-layer, with 8 layers having double branches. Therefore, the total parameters and operation of 3-branch CNN architecture are only 6.6% and 4.3% of the Bayesian optimization method, respectively. Moreover, the structure of grid search and random search methods should be determined in advance. Therefore, both search methods configured in this paper include 3 Conv-layer and 1 full-connection hidden layer. Three-branch CNN architecture’s super parameter candidate set of convolutional kernel size is still selected from Table 4, and the neural node candidate set of full-connection hidden layer is {8, 16, 32, 64, 128}; for grid search, see Tables 3 and 4 for the structure generated by the random search method.

Table 6. Experiments for different searching methods on MNIST dataset.

MNIST	Time (Hours)	Precision
Ours	1	98.7%
Grid Search	2	98.67%
Random Search	1.3	98.49%
Bayesian Search	6	99%

It should be noted that the methodology of automatic parallel selection for super parameters for the branch structure proposed in this paper is a dynamic and automatic search process. It is equivalent to grid search and random search methods trained for searching seven networks, so it saves a lot of time and cost. Moreover, it breaks up the 3-branch architecture by choosing a single branch, or the combination of two branches as the “benchmark”, as well as the complete multi-branch CNN, and a total of seven models with different precision and operations can be generated, as we can see in Table 7.

Table 7. Seven models with different precision and operations.

CNN Models	Precision	Operation	Parameters
Y1 + Y2 + Y3	98.5%	799,010	400,958
Y2 + Y3	98.2%	728,156	365,140
Y1 + Y3	96.3%	499,278	250,932
Y1 + Y2	88.3%	370,586	185,844
Y3	95.9%	428,244	215,114
Y2	70.1%	299,732	150,026
Y1	20.5%	70,854	35,818

3.3. POSS-CNN for Target Recognition and Detection

3.3.1. Multi-Branch Architecture for Target Recognition

In this subsection, the overall architecture of POSS-CNN is obtained, which employs a design method of multi-branch CNN structure and a method of automatic parallel selection for super parameters based on multi-branch CNN structure. The generation process of POSS-CNN applied for target recognition is as follows.

First, set up the precision subsection and initialize multi-branch CNN, the purpose is to stop training if the verification accuracy falls into the precision subsection after some iterations of training. The precision subsection in this paper includes four intervals with the total range of 0.6 to 1, which is (0.6, 0.7], (0.7, 0.8], (0.8, 0.9], and (0.9, 1]. In addition, the specific number of branches is 3 and the number of branches can be set to 2 or 4, 5, etc. Second, set up the super parameters’ search space. The setting methodology can be decided by the developer according to experience (or refer to Table 4). Each search result adds a 2×2 Max-Pooling layer as a block. The block is gradually added to the overall 3-branch CNN architecture. Third, start to search and train the model automatically. Specifically, exploit the parallel characteristics of 3-branch CNN architecture and the greedy strategy. Fourth, as shown in Figure 9, search for the convolutional kernel size automatically from Table 4 as “Block 1”. After adding a 2×2 Max-Pooling layer, put “Block 1” to each branch of the 3-branch CNN architecture for iterative training. Record the selected super parameters results, precision, and operation of each mode after each iterative training. Retain the models until the precision reaches 5% of the maximum precision. Select the model with the lowest operation value as the optimal result of Block 1. “Blocks 2” and “Block 3” can be generated by using a similar approach to “Block 1”, and the difference lies in the parameters of the network model.

In this paper, Block 3 is temporarily stopped from searching, so that after automatic search, the first branch of the 3-branch CNN architecture has 1 convolutional layer and 1 Max-Pooling layer, the second branch has 3 convolutional layers and 3 Max-Pooling layers, and the third branch has 5 convolutional layers and 5 Max-Pooling layers. For different branches, the difference lies in operation and parameters. In addition, the depth of the block is variable, which depends on specific and different requirements, such as setting it to 3, 4, 5, etc.

In addition, as shown in Figure 9, the number of neural nodes in fully connected hidden layers and the channel size of feature maps of each branch are also set in this paper. The first branch is {8, 16, 32, 64}, the second branch is {64, 128, 256}, and the third branch is {256, 512, 1024}. The fully connected layer of each branch in the 3-branch CNN architecture is composed of two hidden layers and one output category. Meanwhile, only

through one-time training, the 3-branch CNN architecture can be broken up into different branch modes to generate a total of seven models with different precision and operations. Moreover, seven models can be applied to different performance hardware platforms at the same time. The complete multi-branch CNN structure (also called the POSS-CNN) is shown in the lower right corner in Figure 9.

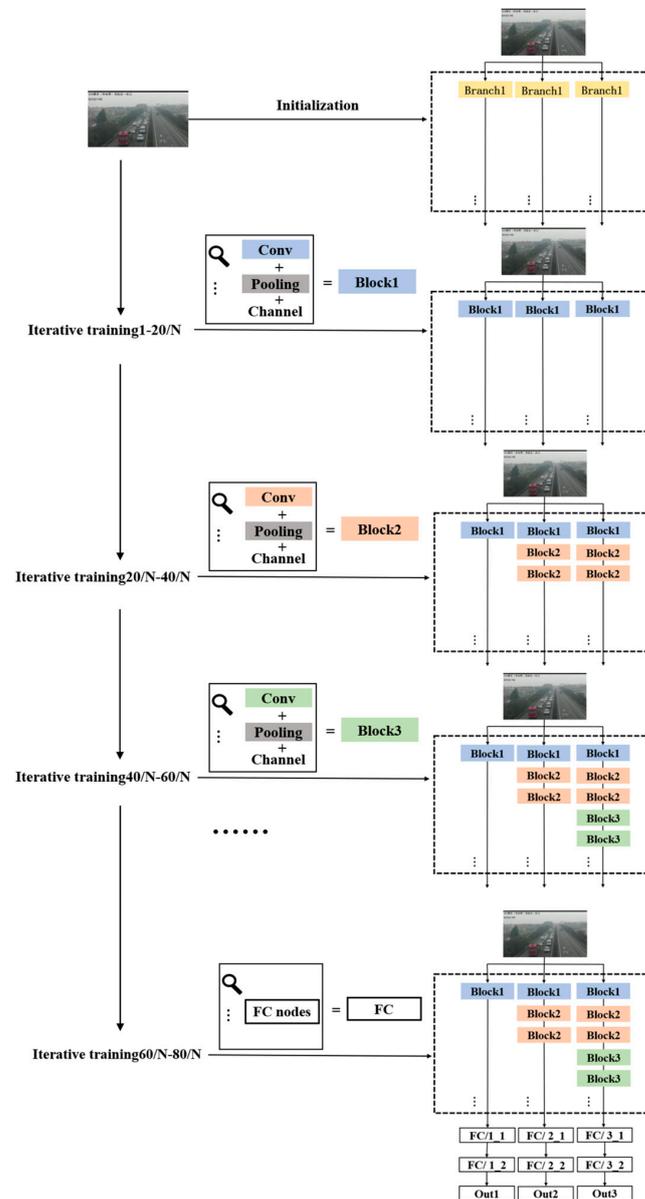


Figure 9. Generation process of POSS-CNN overall architecture.

3.3.2. Multi-Branch Architecture for Target Detection

The generation process of POSS-CNN applied for target detection is as follows.

The “Convolutional structures” of POSS-CNN for target recognition through automatic searching can be regarded as a whole block to be embedded into the existing target detection network, such as YOLOv3. As shown in Figure 10, the embedded block is shown in the red frame, and the first half of the YOLOv3 structure is simplified in this paper. Thus, we can obtain POSS-CNN for target detection tasks. As shown in Table 8, the total parameters of YOLOv3 are 61.95 million, and the operation of YOLOv3 is 65.8 Gops. But the total parameters of POSS-CNN for target detection are 53.56 million, and the operation of POSS-CNN for target detection is 41.8 Gops, which is almost 86% and 63.5% of YOLOv3,

respectively. The embedded structure of the POSS-CNN for target detection is composed of three branches. The first branch has 2 convolutional layers with the kernel size of 3×3 , the second branch has 4 convolutional layers with the kernel size of 3×3 , and the third branch has 8 convolutional layers with the kernel size of 3×3 . Any Max-pooling layers should not be added.

Table 8. Comparison of YOLOv3 and POSS-CNN for target detection.

CNN	Parameters (Million)	Operation (Gops)
YOLOv3	61.95	65.8
POSS-CNN for target detection	53.56	41.8

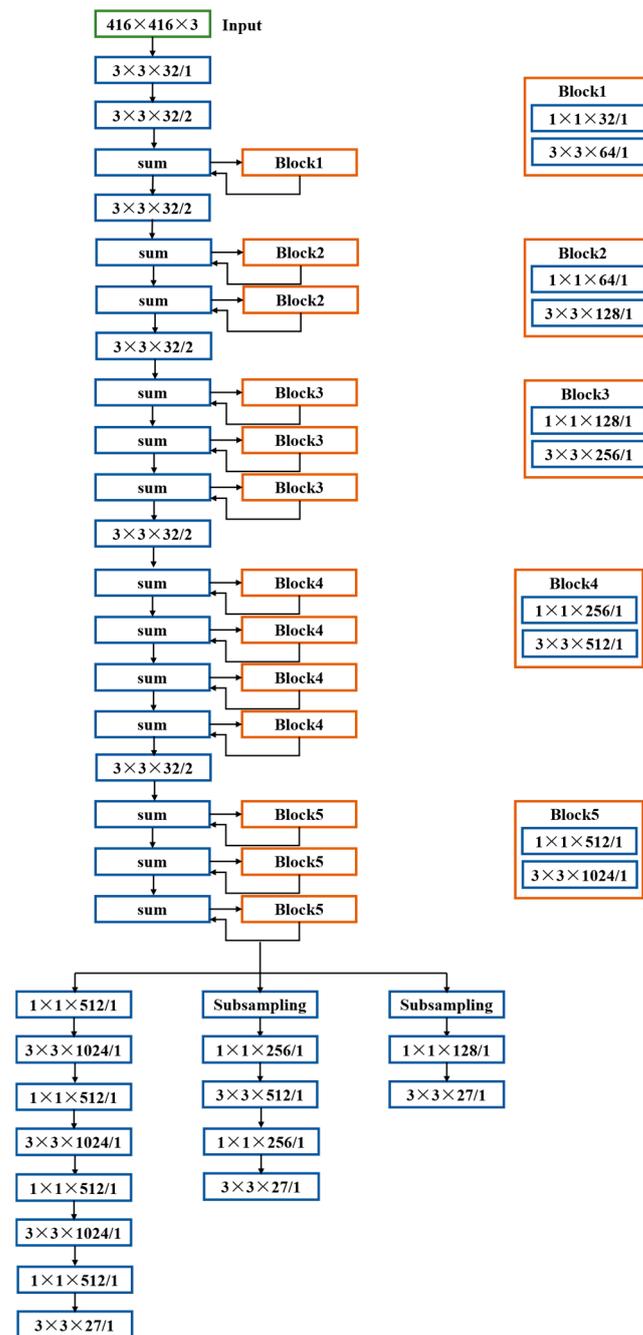


Figure 10. POSS-CNN embedded in target detection model.

4. Results

This section firstly introduces the experimental platform and dataset used in this paper. Secondly, we experiment and verify the overall architecture of target recognition and detection based on the CIFAR10 and LSVH datasets [47]. Finally, we adopt the WRA hardware accelerator to accelerate the core convolutional operation of POSS-CNN, and the relevant experimental results are analyzed.

4.1. Experimental Platform and Datasets

The experimental platform used in this paper is an Intel Xeon silver 4110 CPU (2.1 GHz) server. This server has an NVIDIA RTX 2080Ti v100 GPU. The memory of this GPU is about 10 G, and the power of this GPU is 250 W. The operating system of the server is Ubuntu 16.04. Moreover, this paper uses the Python 3.6 programming language and the open-source framework, Tensorflow, and Keras.

The dataset of this paper includes the MNIST (50 MB) dataset, which was mentioned above, as well as the CIFAR10 dataset with a resolution of $32 \times 32 \times 3$. The Python version of the CIFAR10 dataset is about 163 MB, including 50,000 training images and 10,000 test images. In addition, the LSVH dataset is used to test POSS-CNN for the target detection task. As shown in Figure 11, the LSVH dataset is about 3.53 GB, containing more than 14,000 pictures.

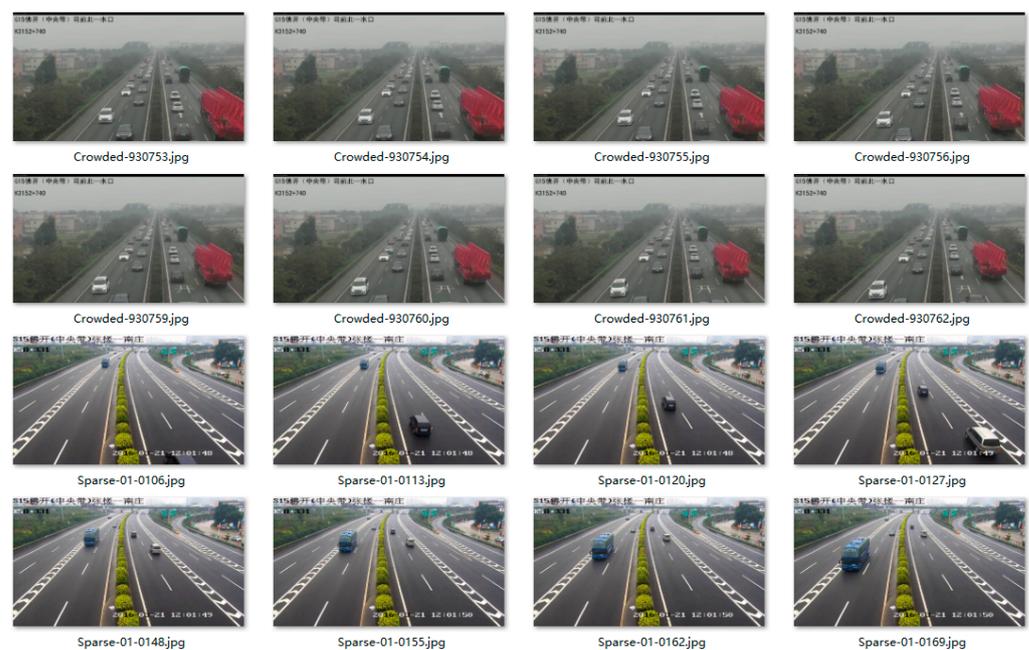


Figure 11. LSVH dataset.

4.2. Target Recognition

The POSS-CNN for target recognition proposed in this paper is trained and tested on the CIFAR10 dataset. The depth of all blocks is two, which means the block is composed of two Conv-layer without an additional Max-Pooling layer.

The experimental results are shown in Table 9. The precision of POSS-CNN for target recognition proposed in this paper is 86.4%, which is higher than VggNet, but lower than AlexNet. However, the operation and parameters of POSS-CNN for target recognition are reduced by 54.1% and 54.2% of AlexNet, and 71.5% and 71.6% of VggNet, respectively, as illustrated in Figure 12. At the same time, only through one-time training can POSS-CNN be broken up for target recognition. By choosing a single branch, or the combination of two branches as the “benchmark”, as well as the complete architecture, and a total of seven models with different precision and operations can be generated. Moreover, seven models

with different precision and operations can be applied to different performance hardware platforms at the same time. As shown in Table 9, seven models like Y1, Y2, Y3, Y1 + Y2, Y1 + Y3, Y2 + Y3, and Y1 + Y2 + Y3 have different parameters and operation.

Table 9. Overall architecture of target recognition tested on CIFAR10 dataset.

CNN	Models	Accuracy	Operation (Ops)	Parameters (Million)
AlexNet	Original	89% [20]	43,225,160	21.62
VggNet	VGG + BN	82.6% [48,49]	67,251,600	33.64
POSS-CNN for target recognition	Y1 + Y2 + Y3	86.4%	19,848,513	9.9
	Y3 + Y2	86.6%	18,232,666	9.1
	Y1 + Y3	78.9%	17,754,486	8.9
	Y1 + Y2	73.4%	3,709,874	1.86
	Y3	69%	16,138,639	8.07
	Y2	73.6%	2,094,027	1.05
	Y1	23%	1,615,847	0.81

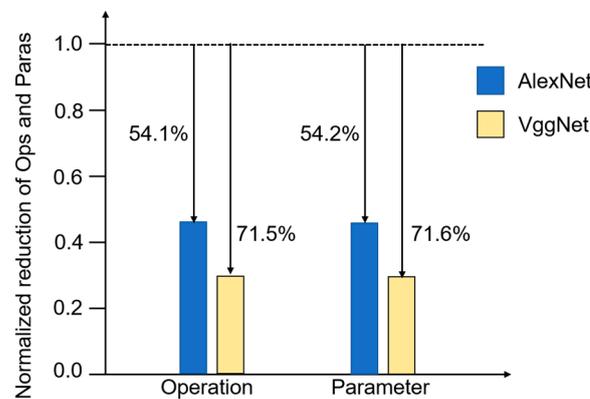


Figure 12. Reduction in operation and parameters normalized to AlexNet and VggNet.

4.3. Target Detection

POSS-CNN for target detection proposed in this paper is trained and tested on the LSVH dataset. As shown in Table 10, first, in terms of precision performance, POSS-CNN for target detection can achieve 45.8% mAP, in which the Sparse mAP is 56.8%, and the Crowded mAP is 51.3%. Compared with YOLOv3, Faster R-CNN, and YOLOv2, the overall precision performance of POSS-CNN for target detection is slightly lower than YOLOv3, because the Crowded mAP is lower than YOLOv3. The overall mAP of the POSS-CNN for target detection is slightly lower than Faster R-CNN. However, the overall mAP of POSS-CNN for target detection is higher than YOLOv2.

Table 10. Overall architecture of target recognition tested on LSVH dataset.

LSVH	MAP	Sparse mAP	Crowded mAP	Operation (Gops)	Parameters (Million)
Ours	45.8	56.8	51.3	41.8	53.56
YOLOv3	62.3	51.8	75.5	65.8	61.95
Faster R-CNN	59.5	58	61.1	46.7	58.19
YOLOv2	32.4	42.1	31.3	57.4	50.59

Secondly, in terms of operation and parameter performance, POSS-CNN for target detection has the lowest amount of operation, which is 41.8 Gops. Figure 13 shows the comparison of operations and parameters. Compared with YOLOv3, the operation and

parameters of the POSS-CNN for target detection are reduced by 57.4% and 15.6%, respectively. Compared with Fast R-CNN, the operation and parameters of the POSS-CNN for target detection are reduced by 11.7% and 8.6%, respectively. Compared with YOLOv2, the operation of POSS-CNN for target detection is reduced by 37.2%, while the parameter is almost equivalent to YOLOv2.

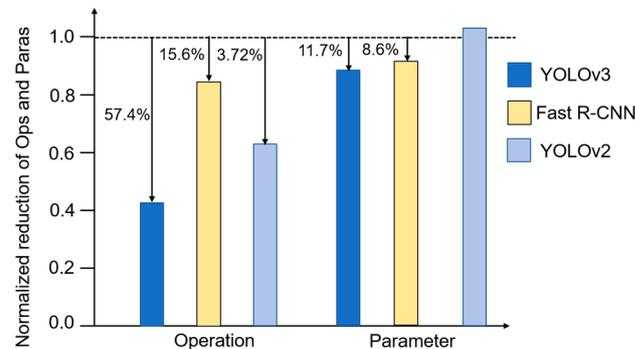


Figure 13. Reduction in operation and parameters normalized to YOLOv3, Fast R-CNN, and YOLOv2.

4.4. FPGA Acceleration and Comparison

The core convolutional operation of the POSS-CNN proposed in this paper is accelerated by a Winograd-based highly efficient and dynamically Reconfigurable Accelerator (WRA). WRA is designed for quickly evolving CNN models, and we proposed a cost-effective convolutional decomposition method (CDW) to extend the application of the fast Winograd algorithm. Based on CDW, a high-throughput and reconfigurable processing element (PE) array was designed to exploit the parallelism of Winograd. Moreover, a highly compact memory structure employing four levels of data reuse schemes was proposed, which can achieve maximum data reuse and minimize external bandwidth requirements. Provided with dynamically reconfigurable capability, WRA implements CDW and other convolutions (e.g., standard convolution, depthwise separable convolution, and group convolution) on a unified hardware architecture.

As shown in Table 11, POSS-CNN for target recognition tested on GPU RTX2080Ti can reach 1919 fps. After accelerating the core convolutional operation by the WRA accelerator, this performance can reach 2392 fps, which is 1.3 times higher than the platform of GPU RTX2080Ti.

Table 11. Experimental results of acceleration by WRA on target recognition.

CNN	Hardware Platform	Performance (fps)
POSS-CNN for recognition	WRA [48,49]	2392
	GPU	1919

The total operation of POSS-CNN for target detection based on the LSVH dataset is 41.8 Gops. As shown in Table 12, the inference performance of the GPU RTX2080Ti platform is about 6 fps, and the energy efficiency is 41 J/f. The inference performance of the CPU i7-8570H platform is about 1 fps, and the energy efficiency is 45 J/f. After accelerating the core convolutional operation by the WRA accelerator, the inference performance can reach 27 fps, and the energy efficiency is 0.42 J/f. Compared with the platforms of GPU RTX2080Ti and CPU i7-8570H, the performance of the WRA platform is increased by 5 times and 27 times, respectively. The energy efficiency is increased by 96.6 times and 106.1 times, respectively.

Table 12. Experimental results of acceleration by WRA on target detection.

CNN	Hardware Platform	Frequency (Hz)	Power (W)	Speed (fps)	Energy Efficiency (J/f)
POSS-CNN for detection	WRA [48,49]	330 M	35	27	0.42
	GPU	1.545 G	250	6	41
	CPU	2.20 G	45	1	45

5. Discussion

This paper discussed a new CNN architecture for the application of target recognition and target detection. On the one hand, a multi-branch CNN structure is conducted with separable precision and operation. For the networks with less classification and low resolution, the size of the datasets is about tens to hundreds of MB. Whereas those networks with more classification and high resolution need datasets of tens to hundreds of GB. Therefore, the branch architecture shown in Figure 5a is selected for target recognition and detection tasks whose datasets are less than 100 MB. On the other hand, a methodology of automatic parallel selection for super parameters is put forward based on a multi-branch CNN structure. Referring to FractalNet [2] and the combined experience, the 3-branch is chosen as the branch number. Actually, it is not necessarily the 3-branch; it can be easily changed to two or more branches. Generally, the search block automatically is used as “Block 1”, and added to each branch of the 3-branch CNN architecture when the best performance is found by training. Similarly, “Blocks 2” and “Block 3” can be generated, and the difference lies in the parameters and operation of the network model. Actually, it is scalable to continue adding convolutional layers, such as “Block 4” and “Block 5”. The measure results indicate that the accuracy is equivalent to AlexNet and VggNet tested on the CIFAR10 dataset, and YOLOv3 tested on the LSVH dataset, respectively. Moreover, the operation and parameters of the whole model are far less than those of AlexNet, VggNet, and YOLOv3.

The superiority of our work is balancing the accuracy of models and computational burden; that is, the proposed CNN structure has fewer operations and parameters along with equivalent performance. The reduction in operations and parameters is because the PSNC method is used for optimization of the network structure. The PSNC method is to substitute most parameters of the fully connected layer with a few parameters of the convolutional layer. Furthermore, this does not cause any degradation in performance. One reason for this is that neuron nodes of a fully connected layer are appropriately increased to compensate for the loss of accuracy. Another is that when targeted for target recognition or detection tasks, the multi-branch architecture is initialized, and further iterated by adaptively searching for building blocks with excellent performance, and then adding these blocks to the three different branches. Actually, the proposed method still has some limitations, which mainly lie in the following two aspects. The first aspect is that the constructed network model is evaluated only on a few certain datasets, and not experimentally trained and tested on more datasets, such as ImageNet. The other aspect is that for a single branch model, such as Y1, the accuracy is significantly inadequate with the compared network models.

Therefore, future research studies will focus on improving and optimizing the models by iteratively conducting the training process and testing process. Combining the advantages of state-of-the-art network models, future research will explore more advanced multi-branch CNN structures, thereby achieving higher accuracy and fewer operations and parameters on various datasets, such as ImageNet, CIFAR-100, and VTAB. The details lie in the following three aspects. The first aspect is to carry out batch sampling experiments and iteratively train these samples, thereby gaining a good usable model. The second aspect is to continue using branch architecture efficiently so that different branches can understand or detect different types of targets in complex scenarios. The last aspect is to further divide the depth of the building blocks in detail to explore more suitable adaptive and automated networks.

6. Conclusions

In this paper, based on the application of target recognition and target detection, we propose a new POSS-CNN architecture. First, we propose a multi-branch CNN structure with separable precision and operation. Only through one-time training can the CNN structure be broken up into different branch modes to generate a total of seven models with different precision and operations. Second, a methodology of automatic parallel selection for super parameters based on a multi-branch CNN structure is put forward, which can make the generation process of a multi-branch CNN structure automatic and procedural. Finally, we obtained POSS-CNN for target recognition and detection, and it was tested on the CIFAR10 dataset, where the test accuracy can reach 86.4%, which is equivalent to AlexNet and VggNet, but the operation and parameters of the whole model are 45.9% and 45.8% of AlexNet, and 29.5% and 29.4% of VggNet, respectively. The mAP of POSS-CNN for a detection task tested on the LSVH dataset is 45.8, which is inferior to the 62.3 of YOLOv3. Compared with YOLOv3, the operation and parameters of the model in this paper are reduced by 57.4% and 15.6%, respectively. After being accelerated by WRA, POSS-CNN for a detection task tested on the LSVH dataset can achieve 27 fps, and the energy efficiency is 0.42 J/f, which is a respective 5 times and 96.6 times better than GPU 2080Ti in performance and energy efficiency.

Author Contributions: Conceptualization, C.Y., J.Z. and J.H.; methodology, J.H. and J.Z.; software, J.Z. and J.H.; validation, J.H. and Q.C.; formal analysis, Q.C.; investigation, S.X. and J.W.; writing—original draft preparation, J.H. and J.Z.; writing—review and editing, J.H. and C.L.; data curation, C.L.; visualization, Y.M.; supervision, C.Y. and C.L.; project administration, C.Y.; funding acquisition, C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation of China under Grant 62176206, and in part by Shenzhen Park of Hetao Shenzhen–Hong Kong Science and Technology Innovation Cooperation Zone Program under Grant HTHZQSW-S-KCCYB-2023040.

Data Availability Statement: Not applicable.

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

References

1. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Separable Image Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8697–8710. [\[CrossRef\]](#)
2. Larsson, G.; Maire, M.; Shakhnarovich, G. FractalNet: Ultra-Deep Neural Networks without Residuals. International Conference on Neural Information. *arXiv* **2017**, arXiv:1605.07648. [\[CrossRef\]](#)
3. Quan, Y.; Zhang, D.; Zhang, L.; Tang, J. Centralized Feature Pyramid for Object Detection. *IEEE Trans. Image Process.* **2023**, *32*, 4341–4354. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Pan, J.; Li, Z.; Wei, Y.; Chen, Z.; Nong, Y.; Zhou, B.; Huan, W.; Zou, J.; Pan, Z.; Liu, W. FEDNet: A real-time deep-learning framework for object detection. In Proceedings of the 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI), Changchun, China, 26–28 May 2023; pp. 1020–1025. [\[CrossRef\]](#)
5. Zhang, X.; Liu, W.; Wu, G. Multi-Branch Cascade Receptive Field Residual Network. *IEEE Access* **2023**, *11*, 82613–82623. [\[CrossRef\]](#)
6. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.
7. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16×6 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929. [\[CrossRef\]](#)
8. Brock, A.; De, S.; Smith, S.L.; Simonyan, K. High-Performance Large-Scale Image Recognition Without Normalization. *arXiv* **2021**, arXiv:2102.06171. [\[CrossRef\]](#)
9. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized evolution for image classifier architecture search. *Conf. Artif. Intell. (AAAI)* **2019**, *33*, 4780–4789. [\[CrossRef\]](#)
10. Kang, M.-G.; Kim, H.-H.; Kang, D.-J. Finding a High Accuracy Neural Network for the Welding Defects Classification Using Efficient Neural Architecture Search via Parameter Sharing. In Proceedings of the International Conference on Control, Automation and Systems (ICCAS), PyeongChang, Republic of Korea, 17–20 October 2018; pp. 402–405.

11. Zhao, H.; Yan, K.; Cao, M. Multi-branch Attention Fusion Network for Aspect Sentiment Classification. In Proceedings of the 2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 21–23 April 2023; pp. 892–896. [[CrossRef](#)]
12. Talemi, N.A.; Kashiani, H.; Malakshan, S.R.; Saadabadi, M.S.E. AAFACE: Attribute-Aware Attentional Network for Face Recognition. In Proceedings of the 2023 IEEE International Conference on Image Processing (ICIP), Kuala Lumpur, Malaysia, 8–11 October 2023; pp. 1940–1944. [[CrossRef](#)]
13. Chi, W.; Liu, J.; Wang, X.; Feng, R.; Cui, J. DBGNet: Dual-Branch Gate-Aware Network for Infrared Small Target Detection. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–14. [[CrossRef](#)]
14. Luo, R.; Tian, F.; Qin, T.; Lin, T.-Y. Neural Architecture Optimization. *arXiv* **2018**, arXiv:1808.07233. [[CrossRef](#)]
15. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 386–397. [[CrossRef](#)]
16. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [[CrossRef](#)] [[PubMed](#)]
17. Canny, J.; Zhao, H.; Jaros, B.; Chen, Y.; Mao, J. Machine learning at the limit. In Proceedings of the IEEE International Conference on Big Data, Santa Clara, CA, USA, 29 October–1 November 2015; pp. 233–242. [[CrossRef](#)]
18. Kang, M.; Kim, Y.; Patil, A.D.; Shanbhag, N.R. Deep In-Memory Architectures for Machine Learning Accuracy Versus Efficiency Trade-Offs. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 1627–1639. [[CrossRef](#)]
19. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]
20. Krizhevsky, A. Imagenet classification with deep convolutional neural networks. *IEEE Adv. Neural Inf. Process. Syst.* **2012**, *60*, 1097–1105. [[CrossRef](#)]
21. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556. [[CrossRef](#)]
22. 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. [[CrossRef](#)]
23. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2261–2269. [[CrossRef](#)]
24. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587. [[CrossRef](#)]
25. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [[CrossRef](#)]
26. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448. [[CrossRef](#)]
27. 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. [[CrossRef](#)]
28. Redmon, J.; Farhadi, A. Yolo9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525. [[CrossRef](#)]
29. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767. [[CrossRef](#)]
30. Lévesque, J.C.; Gagné, C.; Sabourin, R. Bayesian Hyperparameter Optimization for Ensemble Learning. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, New York, NY, USA, 25–29 June 2016. [[CrossRef](#)]
31. Sun, Y.; Xue, B.; Zhang, M.; Yen, G.G. A particle swarm optimization based flexible convolutional autoencoder for image classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2295–2309. [[CrossRef](#)]
32. Fielding, B.; Lawrence, T.; Zhang, L. Evolving and Ensembling Deep CNN Architectures for Image Classification. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8. [[CrossRef](#)]
33. Wang, B.; Sun, Y.; Xue, B.; Zhang, M. A hybrid differential evolution approach to designing deep convolutional neural networks for image classification. In *Australasian Joint Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 237–250. [[CrossRef](#)]
34. Lee, W.Y.; Park, S.M.; Sim, K.B. Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm. *Optik* **2018**, *172*, 359–367. [[CrossRef](#)]
35. Baker, B.; Gupta, O.; Naik, N.; Raskar, R. Designing neural network architectures using reinforcement learning. *arXiv* **2016**, arXiv:1611.02167. [[CrossRef](#)]
36. Neary, P. Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning. In Proceedings of the 2018 IEEE International Conference on Cognitive Computing (ICCC), San Francisco, CA, USA, 2–7 July 2018; pp. 73–77. [[CrossRef](#)]
37. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. 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. [[CrossRef](#)]
38. 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. [[CrossRef](#)]

39. Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.J.; Li, F.F.; Yuille, A.; Huang, J.; Murphy, K. Progressive Neural Architecture Search. *arXiv* **2017**, arXiv:1712.00559. [[CrossRef](#)]
40. Zhong, Z.; Yan, J.; Wu, W.; Shao, J.; Liu, C.L. Practical Block-wise Neural Network Architecture Generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2423–2432. [[CrossRef](#)]
41. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856. [[CrossRef](#)]
42. 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* **2017**, *52*, 127–138. [[CrossRef](#)]
43. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826. [[CrossRef](#)]
44. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Int. Conf. Mach. Learn.* **2015**, *37*, 448–456. [[CrossRef](#)]
45. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the Conference on Artificial Intelligence (AAAI), San Francisco, CA, USA, 4–9 February 2017; pp. 1–12. [[CrossRef](#)]
46. Yang, C.; Zhang, J.; Chen, Q.; Xu, Y.; Lu, C. UL-CNN: An Ultra-Lightweight Convolutional Neural Network aiming at Flash-based Computing-In-Memory Architecture for Pedestrian Recognition. *J. Circuits Syst. Comput. (JCSC)* **2021**, *30*, 2150022. [[CrossRef](#)]
47. Hu, X.; Xu, X.; Xiao, Y.; Chen, H.; He, S.; Qin, J.; Heng, P. SINet: A Scale-Insensitive Convolutional Neural Network for Fast Vehicle Detection. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 1010–1019. [[CrossRef](#)]
48. Yang, C.; Wang, Y.; Wang, X.; Geng, L. WRA: A 2.2-to-6.3 TOPS Highly Unified Dynamically Reconfigurable Accelerator Using a Novel Winograd Decomposition Algorithm for Convolutional Neural Networks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2019**, *66*, 3480–3493. [[CrossRef](#)]
49. Yang, C.; Wang, Y.; Wang, X.; Geng, L. A Stride-based Convolution Decomposition Method to Stretch CNN Acceleration Algorithms for Efficient and Flexible Hardware Implementation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2020**, *67*, 3007–3020. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.