Analysis on Lightweight Network Methods and Technologies

Xirui Liu *

School of International Education, Wuhan University of Technology, Wuhan, Province 143000, China

* Corresponding Author Email: 324206@whut.edu.cn

Abstract. There are currently two main schools of deep learning. One is academic. They pursue stronger performance through powerful, complex models. The other is the engineering school. Their purpose is to efficiently deploy models to various hardware platforms. Complex models have better performance. However, it also brings unavoidable consumption. With the increasing depth of convolutional neural networks, lightweighting has become a key research direction. There are currently four main methods for designing lightweight networks. This article will first introduce CNN model compression and basic convolution operations.This paper also introduces the model compression based on AutoML and the automatic animation design based on NAS. Finally, according to the above three points, this paper introduces the application of the above methods in artificially designed neural networks.This paper mainly introduces the step-by-step evolution of the existing methods. This paper analyzes aspects of current neural network improvements and emerging problems. The significance of this paper is to summarize and deepen the solved problems and key problems in the lightweight process through past experience.

Keywords: Depthwise Separable Convolution, Grouped Convolution, Model Compression, NAS, AutoML

1. Introduction

Mobile target detection based on deep learning is an important research direction in computer vision. Object detection combines the segmentation and recognition of objects into one. Its accuracy and real-time performance is an important capability of the whole system. With the development of computer technology and the wide application of computer vision principles, the use of target detection technology to track targets in real time is becoming more and more popular. The dynamic real-time tracking, positioning and identification of targets by deep neural networks has more and more extensive application value in intelligent transportation systems, intelligent monitoring systems, military target detection, and surgical instrument positioning in medical navigation surgery. It has made life more convenient. At present, on the mobile terminal, deep learning is often unable to exert its powerful functions due to insufficient carrier computing power and insufficient resources. In response to the above problems, more and more attention is devoted to lightweight networks.

This paper summarizes four approaches to design lightweight networks. They in turn are manual design of lightweight models, NAS-based automated design, CNN model compression, and AutoML-based automatic compression. This paper introduces the method of realizing automatic design and automatic compression through basic convolution calculation. This paper also analyzes the mixed application of the three methods to solve different types of problems, and illustrates their role in lightweight development. The main contribution of this paper is to summarize the current lightweight refinement direction and the reasons for the direction selection. At the same time, this paper filters out most of the insignificant directions, and provides the ideas for the main research directions of subsequent articles.

2. Four ways to implement lightweight networks

There are 4 existing lightweight network implementation schemes:

The trained model can be compressed. Thus, a more accurate and fast lightweight network can be obtained. The specific implementation methods of this method are: knowledge distillation, weight quantization, pruning and attention transfer.

An efficient and fast lightweight network can be directly trained. For example, MobileNet and Xception[1] both use depthwise separable convolution to build. ShuffleNet uses the grouped convolutions for acceleration.

NAS-based automated design of neural networks.

Automatic model compression is based on AutoML.

3. Basic Convolution Operation Categories

3.1. Standard convolution

The standard convolution calculation is presented as Formula 1:

$$HWNK^2M \tag{1}$$

H represents the feature map width. W means height. N can be taken as the number of input feature channels. K*K represents the convolution kernel size. M represents the number of output convolution channels.

3.2. Grouped Convolution

The calculation amount of the grouped convolution can be presented as Formula 2:

$$\frac{HWNK^2M}{G} \tag{2}$$

Grouped convolution is a variant of standard convolution. It was first proposed in AlexNET [1]. Input feature channels are divided into G groups.It can be found that his calculation amount is $1/G$ of standard convolution.

3.3. Depthwise convolution

Depthwise convolution refers to dividing the input feature maps into C groups. Each group does k×k convolution. The final calculation amount isas Formula 3:

$$HWK^2M \tag{3}$$

It is equivalent to 1/N of ordinary convolution. The effect is same to collecting the spatial features of each channel individually.

3.4. Pointwise convolution

Pointwise refers to performing k ordinary 1×1 convolutions on the input feature map. It is mainly used to change the feature dimension of the input channel. Its computational cost is as Formula 4:

$$HWNM \tag{4}$$

Its role is equivalent to the exchange of information between channels.

3.5. Channel Shuffle

Because of the Groupedconvolution, the exchange of information is limited to each group. The lack of information exchange between groups leads to a decrease in accuracy. Therefore, an information exchange mechanism between groups, called Channel shuffle, is involved.

3.6. Bottleneck

Bottleneck [2] structure is to reduce the number of parameters. It mainly reduces the impact of the reduction of the parameter amount on the accuracy. Bottleneck has three steps. PW first reduces the

dimensionality of the data. After that, PW performs the convolution of the conventional convolution kernel. Finally, PW upscales the data, similar to the hourglass shape.

4. Automatic model compression

4.1. Introduction of CNN Model Compression

CNN model compression reduces the amount of model computation from the perspective of compressing model parameters. Among the methods proposed by Deep compression, there are three methods for CNN model compression. They are pruning, weight sharing and weight quantization, and Huffman coding. Pruning is to prune some unnecessary network weights and to retain important weights. Weight sharing is when multiple neurons connect the same weight. Weight quantization is to use fewer bits to represent a weight. Huffman coding can effectively reduce redundancy.

CNN model compression can continue the idea of Deep compression [3]. There are four categories of compression algorithms, includingparameter pruning and sharing, low-rank factorization, transferring or compressing convolutional filters, and knowledge distillation. Parameter pruning and sharing concerns and removes redundant parts of the model. Low-rank decomposition uses matrix, tensor decomposition to estimate the most informative parameters in the advancedCNNs. The method of migrating or compressing convolutional filters designs convolutional filters with special structures to reduce the consumption of storage and computation. Knowledge distillation trains a more compact neural network to reproduce the output of larger networks.

4.2. AMC

Traditional model compression techniques rely on manual design. This requires algorithm designers to explore a large design space and make trade-offs between model size, speed, and accuracy. In AMC [4], it proposes the strategies to provide model compression using reinforcement learning. This strategy saves labor costs while maintaining accuracy.

4.2.1. Problem Definition

Model compression can be divided into Fine-grained pruning and Coarse-grained pruning in dimension. Fine-grained pruning mainly implements pruning weights and non-important tensors. This enables high compression rates while maintaining accuracy. Coarse-grained pruning aims to prune the entire regular region of the weight tensor. The weight tensor is $n \times c \times k \times k$. c represents the number of input channels. n represents the number of output channels. k is the convolution kernel size. After fine-grained pruning, the weight tensor is $n \times c' \times k \times k$. The sparsity is c'/c.

The accuracy of model compression is sensitive to the sparsity of each layer, requiring fine-grained action spaces. Therefore, the paper chooses to propose a continuous compression ratio control strategy through DDPG agent. This strategy is encouraged when models shrink and speed upwhen accuracy is lost.

4.2.2. Search space

AMC defines 11 features for each layer t to represent the state of st:

$$(t, n, c, h, w, stride, k, FLOPs[t], reduced, rest, a_{t-1}) \tag{5}$$

In Formula5, t is the layer number. The convolution kernel size is $n \times c \times k \times k$. The input feature size is $c \times h \times w$. $FLOPs[t]$ is the $FLOPs$ of layer Lt. $Reduced$ is the reduced $FLOPs$ of the previous layer. $Rest$ represents the $FLOPs$ of the next layer. These features are all normalized to [0,1].

4.2.3. Search Evaluation Strategies

By limiting the sparsity rate of each convolutional layer, the paper proposes two loss functions for latency-critical and quality-critical applications. A way to reduce the size under the premise of ensuring accuracy. The other is to achieve optimal accuracy with sufficient hardware conditions.

5. AI automates the design of neural networks

5.1. NasNet

NasNet is based on the AutoML method, which first performs a neural network architecture search on a small dataset so that AutoML finds the best convolutional layers and flexibly stacks this to create the final network. At the same time, the best learned framework is transferred to ImageNet image classification and coco object detection.

Nas designs Normal cells and Reduction cells based on human experience. The former does not change the convolution of the size of the input feature map, while the reduction cell reduces the length and width of the input feature map to half of the original convolution, and reduces the size by changing the size of the stride.

5.2. MnasNet

The Mnasnet [5] search space is similar to NasNet, but both accuracy and speed are considered when searching for the objective functionis considered as presented in Formula 6 as below：

$$maximize \quad ACC(m) \times [\frac{LAT(m)}{T}]^{\omega} \tag{6}$$

m represents the model, $ACC(m)$ represents the accuracy of the target model, $LAT(m)$ represents the time-consuming, and T represents the target time-consuming. The paper proposed gradient-based reinforcement learning method to find the Pareto optimality, while improving the $ACC(m)$ and $LAT(m)$Pareto.

Although each block of the MnasNet algorithm contains the different types of convolutional layers, each convolutional layer contains a Depthwise convolution operation, which is still somewhat different from algorithms such as Mobile:

The MnasNet model uses more 5×5 Depthwise convolutions.

The hierarchical search space proposed by MnasNet allows each block of the model to include the different convolutional layers.

MnasNet adopts the idea of reinforcement learning to establish the connection between blocks. A hierarchical search method is used between each block. In this way, it searches for the convolution model of each convolution layer, the convolution sum, the connection method of the convolution layer, and the global optimal solution of the filter size. However, the idea of artificial neural network design is still continued, which has some limitations.

5.3. Development direction of Nas

The first is the improvement of horizontal scalability. The scale-out capability allows one to add more units that build capacity and performance, while providing a more manageable cluster of file storage, rather than a series of siloed devices, each requiring separate management attention. Currently it's just a high-end extension. However, it's a feature built into the controller's operating system. It should be portable to devices in range.

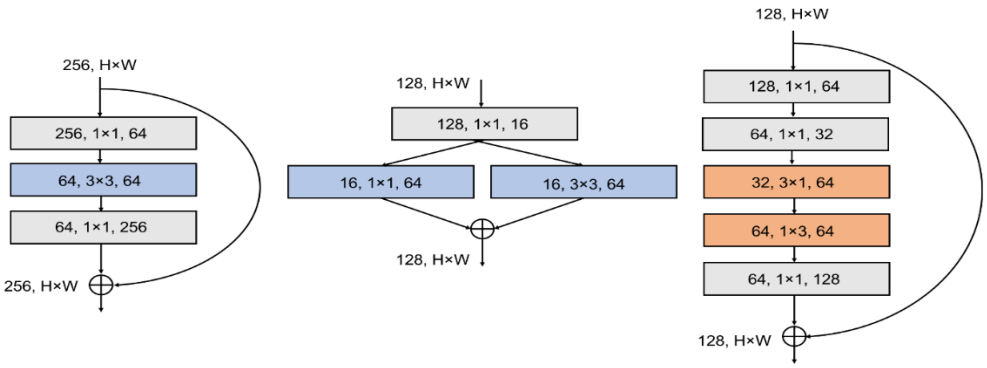

Figure 1 The structures of ResNet(left), SqueezeNet(middle), and SqueezeNext(right) [7]

The second aspect is to come up with more efficient methods to provide better performance. Currently, people increase processing power and throughput by using larger traditional NAS boxes or by adding nodes to a clustered NAS configuration. In this way, NAS performance is greatly improved. However, more efficient methods are still needed to speed up access times and throughput rates.

6. Manual design of lightweight networks

6.1. SqueezeNet

Although the SqueezeNet [6] series has less applications, its architectural ideas are still worth learning.

6.1.1. SqueezeNet

As one of the early studies focusing on lightweight networks, SqueezeNet mainly uses the fire module for parameter compression. Its core is the fire module. The fire module is compressed by the squeeze convolution layer in the input layer, and the parameter amount is reduced by 1×1 convolution. It then uses the expand convolution layer, which is sequentially synthesized by 1×1 and 3×3 convolutions to achieve dimension expansion. To sum up, the core of SqueezeNet, contains three parameters, which are the number of convolution kernels of the Squeeze layer and the number of two convolution kernels of the expansion. They are respectively expressed as $s_{1x1}e_{1x1}$ and e_{3x3}, these parameters usually satisfy$s_{1x1} < (e_{1x1} + e_{3x3})$.

6.1.2. SqueezeNext

As an upgraded version of SqueezeNet, SqueezeNext[7] mainly follows the residual structure in its design. Figure 1 shows the structure of ResNet, SqueezeNet, and SqueezeNext.

Compared with the popular depth separation convolution at that time, SqueezeNext directly uses separation convolution. The design mainly follows the following strategies:

Under this process, the parameter amount can be changed from K squared to 2×K.

Bottleneck Module: The amount of parameters is related to the output dimension. Since the calculation of the depth separation convolution in the terminal system is not efficient, the depth separation convolution is not used to reduce the calculation amount.However, the Squeeze layer of SqueezeNet is used to compress the output dimension. Two Squeeze layers are used at the beginning of each block, in order to reduce 1/2 dimension in each layer.

Fully Connected Layers: In AlexNet, the parameters of the fully connected layer account for 96% of the model. It can be seen that it is necessary to reduce the fully connected layer. Therefore, in SqueezeNext, the bottleneck Module can reduce the input dimension of the fully connected layer, thereby reducing the network's parameters to improve operating efficiency.

6.2. ShuffleNet

As a very important series in the lightweight network, the ShuffleNet[8] series introduced the channel shuffle operation in v1, and proposed the channel split operation in v2, which accelerated the network while reusing features, and achieved excellent results.

6.2.1. ShuffleNetv1

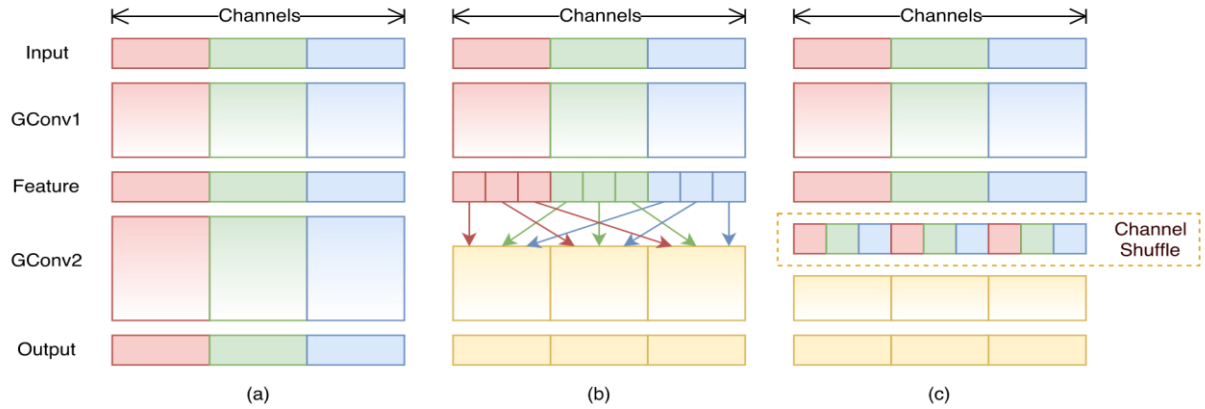

Figure 2. Channel shuffle with two stacked group convolutions [8]

As mentioned earlier, the core of ShuffleNetv1 was to use channel shuffle to make up for the lack of information exchange caused by the previous grouping operation, so that the shortcomings of pointwise can be greatly reduced, and the network can use pointwise grouping convolution as much as possible. Moreover, the dimension of convolution can also be increased, and the accuracy can be better maintained while improving the speed. Figure 2 shows the channel shuffle.

In the current mainstream lightweight networks, pointwise convolution is usually used to reduce the dimensionality in order to decrease the complexity of the network and to realize the lightweight of the network. However, due to the high dimension of the output, this will lead to a significant increase in the pointwise convolution overhead. For small networks, expensive pointwise convolutions may bring the significant performance degradation. For example, in the ResNext [9] unit, pointwise accounts for 93.4% of the computation. To solve this problem, this ShuffleNetv1 introduced the grouped convolution and proposed the different implementation methods:

Dimensional isolation of all groups can greatly reduce the amount of computation. However, this will result in a specific output being only associated with a small part of the input. It will block the exchange of information between groups, resulting in a large loss of accuracy and reducing the expressiveness of the network.

It redistributes the dimensions of the output, first dividing each group into multiple subgroups. It then merges the subgroups of each group with the subgroups of other groups, outputting to different groups. Finally it enables information exchange between groups to better preserve the flow of information between groups to improve the accuracy.

6.2.2. ShuffleNetv2

Since the pointwise grouped convolution operation of ShuffleNetv1 and bottleneck will improve the MAC in the result, it leads to a large amount of non-negligible computational resource consumption. In order to achieve high performance and high accuracy, it is necessary to obtain large-dimensional convolutions with the same input and output without dense convolution and excessive grouping. In order to achieve this purpose, ShuffleNetv2 [10] is based on practice and guided by the actual inference speed. The design essentials of 4 lightweight networks can be described as below:

When the number of input and output channels is the same, the amount of memory access MAC is the smallest.

If the number of groups is too large, the grouped convolution will increase the MAC.

Fragmentation operations are not friendly to parallel acceleration.

The memory and time consumption brought by element-by-element operations cannot be ignored.

Based on the above 4 points, ShuffleNetv2, which took into account both speed and accuracy, is proposed. Its channel divided the input features into two parts, which achieves a feature reuse effect similar to DenseNet.

As shown in Figure 3, the channel spit operation is added to a and b on the basis of v1. At the beginning of each unit, the feature map is divided into two parts, namely c-c' and c', one of which is directly backward. The other branch contains 3 convolutions with the same input and output dimensions. Since the beginning of each unit is equivalent to a grouped convolution operation, v2 does not use the grouped convolution to avoid invalid operations. After the convolution operation is completed, the feature map is restored to the input size of the unit, and then channel shuffle is performed. The element-wise addition operation is removed to reduce a part of the

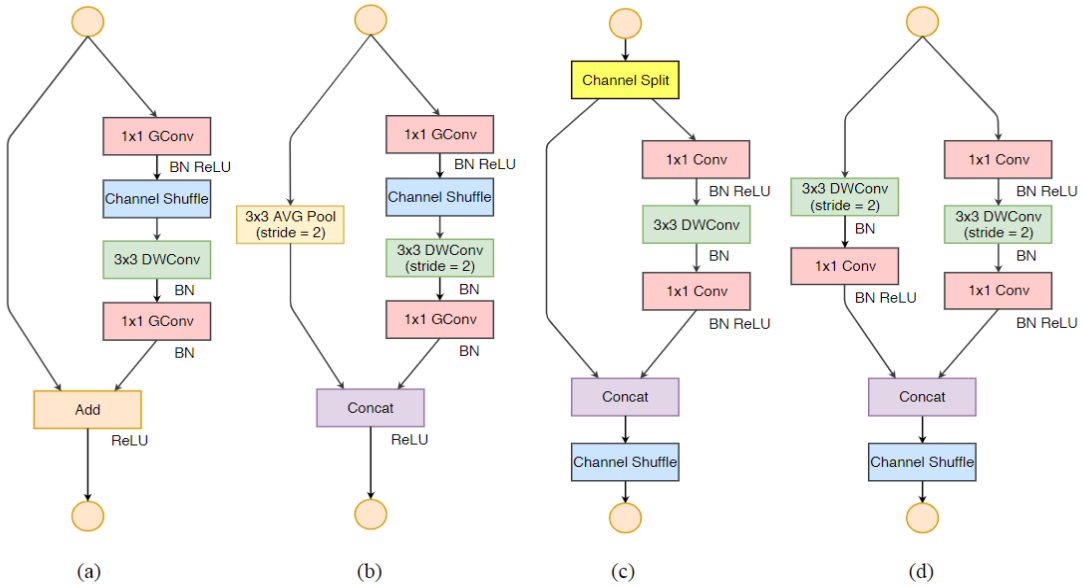

Figure 3. (a): Basic ShuffleNet unit; (b) ShuffleNet unit for spatial downsampling; (c): Basic Shufflenetv2 unit; (d) Shufflenetv2 unit for spatial downsampling [10]

computational load. By combining concat, channel shuffle, and channel split into one element-wise operation during implementation, the computational complexity is significantly reduced and the performance is further improved. However, the basic convolution unit of ShuffleNet still uses Depthwise and Pointwise to reduce the amount of calculation.

Although ShuffleNet reduces the amount of computation, it led new problems, as presented as below:

The channel shuffle operation consumes a lot of memory and pointer jumps, which leads to the problem of the excessive time-consuming.

The rules of channel shuffle are artificially designed, and there is artificial randomness in the information exchange between groups, which is difficult to predict.

6.3. MobileNet

Similar to ShuffleNet, MobileNet[11] also used basic block unit folding to form its neural network architecture. After many developments, it had successively proposed the use of depthwise separable convolution to build a lightweight network, and an innovative inverted residual with linear bottleneck unit. AutoML technology is combined with the manual fine-tuning for a more lightweight network architecture.

6.3.1. MobileNetv1

By using depthwise separable convolutions instead of standard convolutions, and using the width multiplication to reduce the amount of parameters, MobileNetv1 has successfully become a small, computationally-intensive convolutional neural network suitable for mobile devices.

The depthwise separable convolution factorizes the standard convolution into a depthwise convolution and a pointwise convolution. The computational cost of standard convolution is

$HWNK^2M$, and the total computational cost of depthwise separable convolution is presented in Formula 7:

$$\frac{depthwise+pointwise}{conv} = \frac{(K^2+M)HWN}{K^2MHWN} = \frac{1}{M} + \frac{1}{K^2} \tag{7}$$

In general, the number of output feature channels in the network architecture is much larger than the size of the convolution kernel, that is, the computational complexity of the depthwise separable convolution can be significantly reduced by 1/8 to 1/9 of the standard convolution computation.

In order to further reduce the amount of computation, the number of input and output feature channels M and N can be multiplied by the width factor $\alpha(\alpha \in (0,1)$,typical values of d are 0.25, 0.5 and 0.75), so that the total computation amount of the depthwise separable convolution can be further reduced, as presented as Formula 8 :

$$HWK^2 * \alpha M + HW * \alpha N * \alpha M \tag{8}$$

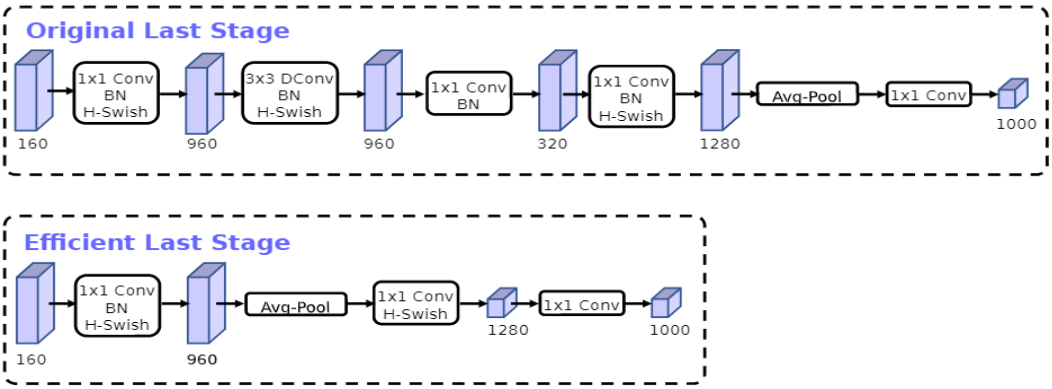

Figure 4. Comparison of original last stage and efficient [14]

6.3.2. MobileNetv2

Due to the reference to the traditional chain architecture such as VGGNet[12], the designs of MobileNetv1 mostly increase the network depth by means of layered convolution, thereby improving the recognition accuracy. Stacking will cause the problem of gradient disappearance in the convolutional layer. Since residual networks make it easier for the information to flow between layers, it can include feature reuse during forward propagation and mitigating gradient signal disappearance for back propagation. The improved version of MobileNetv2[13] adds skip Connection, and makes the following improvements to the basic blocks of Resnet and Mobilenetv1:

It kept the depthwise separable convolution of Mobilenetv1 to reduce the amount of convolution computation.

By adding the skip connection, feature reuse is provided during forward propagation

It adopted the inverted residual block structure. The structure used Pointwise convolution to first upgrade the feature map, and then to connect the ReLU to the features after the upgrade.

6.3.3. MobileNetv3

MobileNetv3[14] first uses AutoML to build the network, and then performs manual fine-tuning optimization. The search method uses platform-aware NAS for global search and NetAdapt for local search. The subsequent manual fine-tuning adjusts the structure of the front and rear layers of the network, and the bottleneck adds SE module. A computationally efficient h-swish nonlinear activation is proposed. Figure 4 shows the comparison of original last stage and efficient last stage.

6.4. ESPNet

ESPNet is mainly used for semantic segmentation, and its core lies in the ESO module. This module contains pointwise convolution and astrous convolution pyramids, which are used to reduce the computer complexity and resample the different features in the effective receptive field. The ESP

module is more efficient than other convolutional decompositions, such as MobileNet and ShuffleNet. At the same time, the ESPNet[15] paper also found that although the hole convolution pyramid brings a larger receptive field, the direct connected output will bring the strange grid lines. In order to solve this problem, the paper proposed the HFF operation, and the output is added hierarchically and then concatenated. Compared with adding extra convolution for post-processing, HFF operation can solve the grid texture more effectively without bringing too much computation. Adding a shortcut connection from input to output can ensure the gradient transfer of the network.

In ESPv2, it replaced pointwise convolution with the grouped pointwise convolution, and then replaced the computationally expensive spatial convolution with depthwise separable spatial convolution. Finally, HFF was used to eliminate the grid texture, and the output features are added for a feature extraction. This architecture operates more efficiently, however, its accuracy cannot be improved significantly.

7. Conclusion

This paper presents four approaches to design lightweight networks. The four approaches have their own advantages and are mutually progressive. This paper introduces the currently important and common structures and analyzes their principles by introducing the basic convolution operations. This paper analyzes the randomness of artificial design, which leads to its inefficiency in design, thus extending to the use of AI for design. It analyzes the working principles of current network design and compression based on NAS and AutoML respectively, and analyzes some methods to improve efficiency. This paper mentions the development direction of NAS with current deficiencies. It then introduces the different convolution structures based on convolution operations and their advantages and principles. This paper also analyzes the effect of the combination of NAS and AutoML on neural network design based on the analysis of different neural network structures. It analyzes the existing network and obtains the general law in the lightweight development of neural network.

References

[1] F, Chollet. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

[2] A. M., Saxe, Y, Bansal., j, Dapello, M, Advani., A., Kolchinsky, B. D., Tracey, & D. D., Cox (2019). On the information bottleneck theory of deep learning. Journal of Statistical Mechanics: Theory and Experiment, 2019(12), 124020.

[3] S., Han, H., Mao, & W. J., Dally (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

[4] Y., He, J., Lin, Z, Liu, H., Wang, L. J., Li, & S., Han (2018). Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

[5] M., Tan, B., Chen, R., Pang, V., Vasudevan, M., Sandler, A., Howard, & Q. V., Le (2019). Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2820-2828).

[6] F. N., Iandola, S. Han, M. W., Moskewicz, K., Ashraf, W. J., Dally, & K., Keutzer(2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.

[7] A., Gholami, K., Kwon, B., Wu, Z., Tai, X., Yue, P., Jin, ... & K., Keutzer (2018). Squeezenext: Hardware-aware neural network design. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 1638-1647).

[8] X., Zhang, X., Zhou, M., Lin, & J., Sun (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6848-6856).

[9] C., Szegedy, S., Ioffe, V., Vanhoucke,, & A. A., Alemi (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In Thirty-first AAAI conference on artificial intelligence.

[10] N., Ma, X., Zhang, H. T., Zheng, & J., Sun (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European conference on computer vision (ECCV) (pp. 116-131).

[11] A. G., Howard, M., Zhu, B., Chen, D., Kalenichenko, W., Wang, T., Weyand, ... & H., Adam (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[12] K., Simonyan, & A., Zisserman (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[13] M., Sandler, A., Howard, M., Zhu, A.,Zhmoginov, & L. C., Chen (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

[14] A., Howard, M., Sandler, G., Chu, L. C.,Chen, B., Chen, M., Tan, ... & H., Adam (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1314-1324).

[15] S., Watanabe, T. Hori, S., Karita, T., Hayashi, J., Nishitoba, Y., Unno, ... & T., Ochiai (2018). Espnet: End-to-end speech processing toolkit. arXiv preprint arXiv:1804.00015.