remote sensing

MDPI

Review

Review of Image Classification Algorithms Based on Convolutional Neural Networks

Leiyu Chen [1], Shaobo Li [1,2,*], Qiang Bai [1], Jing Yang [1,2], Sanlong Jiang [1] and Yanming Miao [3]

[1] College of Mechanical Engineering, Guizhou University, Guiyang 550025, China; gs.lychen19@gzu.edu.cn (L.C.); cme.qbai18@gzu.edu.cn (Q.B.); jyang23@gzu.edu.cn (J.Y.); gs.sljiang19@gzu.edu.cn (S.J.)
[2] State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China
[3] Key Laboratory of Advanced Manufacturing Technology of the Ministry of Education, Guizhou University, Guiyang 550025, China; gs.ymmiao21@gzu.edu.cn
* Correspondence: lishaobo@gzu.edu.cn; Tel.: +86-851-83622068

Abstract: Image classification has always been a hot research direction in the world, and the emergence of deep learning has promoted the development of this field. Convolutional neural networks (CNNs) have gradually become the mainstream algorithm for image classification since 2012, and the CNN architecture applied to other visual recognition tasks (such as object detection, object localization, and semantic segmentation) is generally derived from the network architecture in image classification. In the wake of these successes, CNN-based methods have emerged in remote sensing image scene classification and achieved advanced classification accuracy. In this review, which focuses on the application of CNNs to image classification tasks, we cover their development, from their predecessors up to recent state-of-the-art (SOAT) network architectures. Along the way, we analyze (1) the basic structure of artificial neural networks (ANNs) and the basic network layers of CNNs, (2) the classic predecessor network models, (3) the recent SOAT network algorithms, (4) comprehensive comparison of various image classification methods mentioned in this article. Finally, we have also summarized the main analysis and discussion in this article, as well as introduce some of the current trends.

Keywords: image classification; convolutional neural networks; deep learning

Citation: Chen, L.; Li, S.; Bai, Q.; Yang, J.; Jiang, S.; Miao, Y. Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sens.* **2021**, *13*, 4712. https://doi.org/10.3390/rs13224712

Academic Editor: Claudio Persello

Received: 7 October 2021
Accepted: 10 November 2021
Published: 21 November 2021

1. Introduction

Image classification, as a classical research topic in recent years, is one of the core issues of computer vision and the basis of various fields of visual recognition. The improvement of classification network performance tends to significantly improve its application level [1], for example to object-detection [2], segmentation [3], human pose estimation [4], video classification [5], object tracking [6], and super-resolution technology [7]. Improving image classification technology is an important part of promoting the development of computer vision. Its main process includes image data preprocessing [8], feature extraction and representation [9], and classifier design [10].

The focus of image classification research has always been image feature extraction, which is the basis of image classification. Traditional image feature extraction algorithms focus more on manually setting specific image features. This method has poor generalization ability and portability. So, letting a computer have the ability to process images similar to biological vision is what researchers dream of. ANN is an abstract biological neural network, which is a mathematical operation model composed of a large number of interconnected neurons. It approximately simulates the neural network processing of neural signals. Initially, McCulloch and Pitts analyzed biological neural networks and proposed an internal logical operation mathematical model of neuron activity—MP neuron model [11]. Rosenblatt added learning functions to the MP model and proposed a single-

layer perceptron model, putting the research on neural networks into practice for the first time [12].

After that, Huber and Wiese et al. studied the visual cortex of the cat's brain and found that biological visual neurons perceive information based on local regional stimulation, they concluded that visual perception is stimulated layer by layer through multi-level receptive fields [13]. Later, researchers have tried to use the multilayer perceptron to learn features and trained the model with the backpropagation (BP) algorithm [14]. This discovery inspired researchers to construct a computer neural network similar to a biological vision system has become a reality, and CNN was born. Lecun et al. presented the first batch of the CNN model—LeNet-5 [15]. However, due to the lack of large-scale training data, it is also limited by the theoretical foundation and computer computing power, the recognition results of LeNet-5 on complex images were not ideal [16]. At that time, this model only had an excellent performance on handwriting recognition tasks.

Hinton et al. proposed an effective learning algorithm for learning difficulties in multi-hidden-layer neural networks [17], thus opening a new chapter in deep learning. Subsequently, researchers realized the convolution operation on the GPU, which greatly improved the computational efficiency of the network. Compared with the CPU operation speed, it increased by 2–24 times [18]. Since then, deep learning has attracted more and more attention. Krizhevsky et al. built the AlexNet model based on the LeNet-5 [19]. In the ILSVRC2012 ImageNet competition, it surpassed the second-best entry by a huge advantage. After AlexNet achieved excellent results in the ImageNet image classification competition, researchers began to study CNN more deeply, Zeiler and Fergus proposed a visualization technique to understand CNNs and proposed ZFNet [20]. Min Lin et al. proposed NIN network [21], which contributed to the control of the parameter amount and the number of channels. Next, refs. [22–27] yielded high performance during the ILSVRC2014-2017 classification challenge, they all made great innovations on the original basis. From 2017 to the present, more models with superior performance have appeared one after another. CNNs have increasingly demonstrated irreplaceable superiority in image classification.

With the successful application of CNN to large-scale visual classification tasks, around 2015, the application of CNNs has finally taken off in the remote sensing image analysis field [28,29]. A variety of CNN-based scene classification methods have emerged by using different strategies of exploiting CNNs [30–36]. Generally, CNN-based remote sensing image scene classification methods can be divided into three types: (i) The pre-trained CNNs is used as a feature extractor [37–44]. (ii) Fine-tune the pre-trained CNNs on the dataset [30,45–52]. (iii) Globally initialize the weights of CNNs for training [31,53–55]. As we all know, the CNN-based image classification method was originally designed for computer vision. However, many researchers have successfully applied them to the field of the remote sensing. It is necessary to systematically summarize the image classification methods based on CNN to make researchers get inspiration in their new work. Although there are some surveys on CNNs [56–58], they have not comprehensively introduced almost all classic CNN architectures. This review is dedicated to detailing the development of almost all typical CNNs in image classification tasks and hopes to provide more help for the inspiration of designing CNN models in remote sensing image scene classification field.

Section 2 of this article introduces the basic structure and principle of classic neural networks and the necessary network layers of convolutional neural networks; Section 3 explains the network structure, operating principles, and advantages of the classic network model in the image classification algorithm. It also summarizes the strong training strategy and representative pure/mixed/non-CNN models that has superior performance from 1998 to 2021; Section 4 introduces the commonly used datasets for image classification and compares the performance and characteristics of the network mentioned in the previous section; finally, our review and the work direction on image classification are itemized analysis.

2. Overview of CNNs

This section will introduce the basic concepts of CNNs, which will lead the reader to a preliminary underlying of its fundamental data transmission and its components so that the comprehension of the following sections will be easier.

2.1. Neural Network

2.1.1. Neuron

The biological nervous system is a network composed of many neurons. Similarly, neurons are also the basic processing unit of artificial neural networks. The principle of operation is that multiple input values undergo mathematical transformation to obtain an output value (Figure 1). The mathematical transformation relationship between the input signal and the output value is

$$f\left(b + \sum_{i=1}^{n}(x_i \times w_i)\right) \tag{1}$$

$f(\cdot)$ is the activation function, there are many activation functions, such as ReLU, Sigmoid, Tanh, etc.

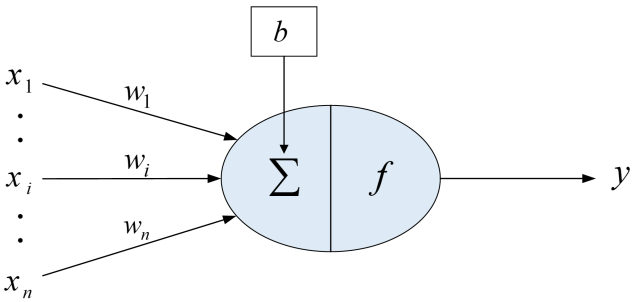


Figure 1. Neuron model, x_i is the input signal, n is the number of signals, the weight value of the input signal is w_i, bias is b and output of neurons is y.

2.1.2. Multilayer Perceptron (MLP)

MLP is composed of the Input layer, Hidden layer (one or more), and Output layer. It contains multiple basic unit neurons, which conduct signal transmission through layer-by-layer conduction between neurons. Figure 2 is an example of the structure of MLP. H is the vector of output value of the hidden unit $H = F(W_h X + B_h)$, Y is the vector of output value of the output unit $Y = F(W_y H + B_y)$. Here, X is the input value matrix, W_h and W_y are the weight matrix between layers, B_h and B_y are the bias matrices.

The BP algorithm is divided into two stages: forward propagation and backpropagation. After the calculation of the output layer is completed, the forward propagation ends. The backpropagation involves the update of various parameters, which is an important part of network learning. The first thing that backpropagation needs to determine is the loss function of the model. There are many loss functions, see Section 2.2.5. The loss function in the Figure 2 is L_2 Loss:

$$Loss(y, y^*) = \frac{1}{m} \times \sum_{i=1}^{m}(y_i^* - y_i)^2 \tag{2}$$

By calculating the mean square error between y^* and y, the network weights w and bias b can be updated by obtaining the partial derivative of the loss function: $w' = w - \eta \times (\partial Loss/\partial w)$ and $b' = b - \eta \times (\partial Loss/\partial b)$. With a suitable learning rate η, the loss of the y and y^* can be gradually minimized. That is to make the y closer to y^*, so as to achieve the effect of network training.

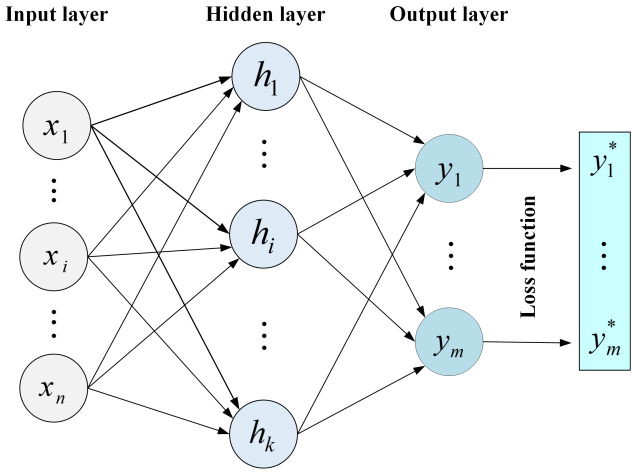


Figure 2. The structure of the MLP. It has n input values and m output values, including k hidden units. x_n is the input value. The direction of the arrow is the direction in which the input value is transmitted. The hidden unit is h_k, it receives the input value of the previous layer. y_m is the output unit, and the real value is y_m^*.

2.2. CNN Architecture

The main structure of CNN is the convolutional layer, pooling layer, nonlinear activation layer, and fully connected layer. Generally, the image is preprocessed [8] and then input into the network through the input layer, processed by several alternately arranged convolutional layers and pooling layers, and then classified by the fully connected layer.

Compared with MLP, CNN [59,60] adds a very characteristic convolutional layer and pooling layer. In the face of more pixels and larger data sets, CNN will have outstanding cost performance in terms of model size and the performance will be better. On the one hand, the convolutional layer has the characteristics of a local receptive field, which retains the input shape so that the correlation between the features of the image pixels in the length and width directions can be effectively identified. On the other hand, the convolutional layer repeatedly calculates the same convolution kernel and different positions of the input through a sliding window, that is, using parameter sharing and sparse connection to effectively avoid the training parameter size from being too large. The pooling layer reduces the computational burden by reducing the number of connections between the convolutional layers [61] and alleviates the excessive sensitivity of the convolutional layer to the position. CNN ensures the invariance of the input image pixels in displacement, scaling, and distortion to a certain degree [62].

2.2.1. Convolutional Layer

For CNNs with a certain depth, the convolution operation of multiple convolution layers can extract different features of the input. The bottom layer convolution generally extracts common features such as texture, lines, and edges, while the higher layer extracts more abstract features. The convolutional layer has several convolution kernels with learnable parameters. It is a matrix composed of learnable weights, which are generally 3×3, 5×5, and 7×7 weight matrices with equal length and width and an odd number. Usually, the convolutional layer will input the feature maps. The weight matrix of the convolution kernel corresponds to the local area of the connection feature map, and the convolution kernel sequentially performs convolution operations on the area on the feature map by sliding [63].

Generally, the size of the input feature maps is $H \times W \times C$ (height H, width W and channels C), each convolution kernel is $K \times K \times C$, this is, the number of convolution kernel should be the same as the number of input channels. Figure 3 is a schematic diagram of

the convolution process of the input feature maps ($5 \times 5 \times 3$) and a convolution kernel ($3 \times 3 \times 3$). The flow of data in the convolutional layer can be roughly expressed as:

$$feature\_surface_{out} = f\left(\sum_{i=3}^{3} M_i * W_i + B\right) \tag{3}$$

There, M_i represents a feature surface of the input feature maps, W_i is the weight matrix of the convolution kernel, the bias matrix is M, $f(\cdot)$ is the nonlinear activation function and $feature\_surface_{out}$ is an output feature surface.

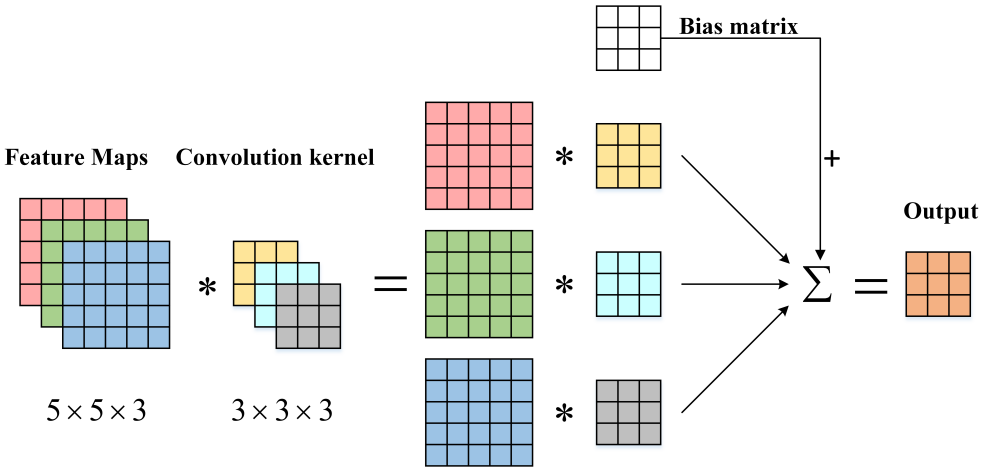
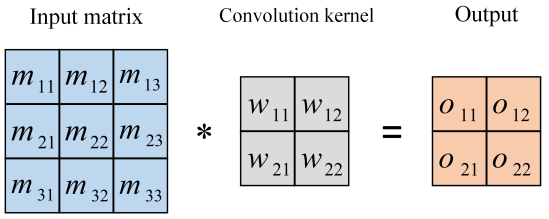


Figure 3. Schematic diagram of the convolution process.

The specific calculation in the convolution layer is the cross-correlation operation between the convolution kernel and the feature surfaces. For any input two-dimensional (2-D) matrix size i, convolution size k, strides s and padding p, the output feature surface size [64]:

$$o = \left[\frac{i + 2p - k}{s}\right] + 1 \tag{4}$$

As shown in Figure 4, simply assume that one of the above-mentioned feature surfaces and the cross-correlation operation of the convolution kernel, the input is feature surface matrix both height and width of 3, the convolution kernel starts from the top left of the input matrix and slides on the input array in order from left to right and top to bottom.



$$o_{11} = w_{11}m_{11} + w_{12}m_{12} + w_{21}m_{21} + w_{22}m_{22}$$
$$o_{12} = w_{11}m_{12} + w_{12}m_{13} + w_{21}m_{22} + w_{22}m_{23}$$
$$o_{21} = w_{11}m_{21} + w_{12}m_{22} + w_{21}m_{31} + w_{22}m_{32}$$
$$o_{22} = w_{11}m_{22} + w_{12}m_{23} + w_{21}m_{32} + w_{22}m_{33}$$

Figure 4. Convolution operation (2-D), kernel size = 2, strides = 1, padding = 0.

It is worth mentioning that the weight parameters of the convolution kernel are also updated through gradient backpropagation. When the convolution kernel processes the same batch of input feature maps, the parameters are fixed. Each pixel area is operated by the same convolution kernel sliding operation, which is the parameter sharing of the convolution kernel. This mechanism makes the operation simple and efficient, and can operate on a very large-scale dataset, which greatly reduces the amount of training parameters and reduces the risk of overfitting [65].

2.2.2. Pooling Layer

The pooling layer is generally after the convolutional layer. The main reasons for using the pooling layer are: Perform down-sampling and dimensionality reduction processing on the input image to reduce the number of convolutional layer connections, thereby reducing the burden of network computing [61]; Realize the scale invariance, translation invariance and rotation invariance of the input image [62]; Make the output feature map more robust to the distortion and error of a single neuron [66].

Average pooling and maximum pooling are the two most widely used pooling methods. Although there are max pooling and average pooling, some methods that can more effectively alleviate the over-fitting of convolutional neural networks are proposed, such as L_p Pooling [67], Mixed Pooling [68], Stochastic Pooling [69], Spatial Pyramid Pooling (SPP) [70] and Multi-scale Orderless Pooling [71], etc. For the classic convolutional neural network model, although the best pooling operation is not average pooling or max pooling, they are the two most classic methods [72]. In [72] Bourbeau mainly conducted a theoretical analysis on the performance of average pooling and maximum pooling. Figure 5 shows the down-sampling process of maximum pooling and average pooling, the relationship between the input and output matrix sizes in the pooling operation satisfies the following general relationship [64]:

$$o = \left[\frac{(i-k)}{s} + 1 \right] \tag{5}$$

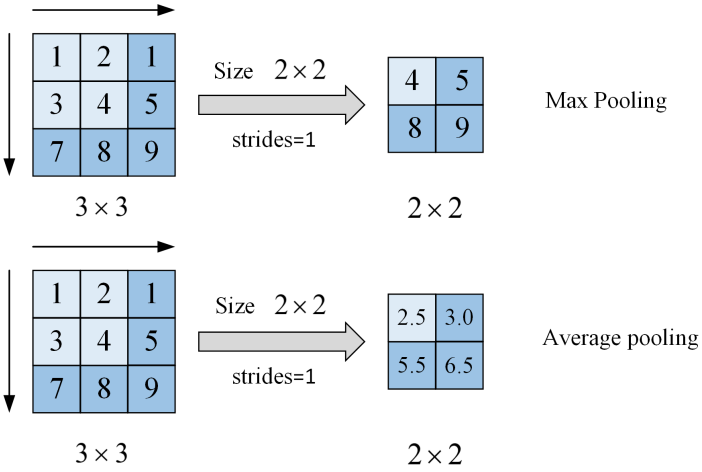


Figure 5. Max Pooling and Average pooling, it does not involve zero padding.

2.2.3. Nonlinear Activation Function

The activation function is to make the input and output have a functional relationship, which introduces nonlinear system into the neural network, and having a suitable nonlinear activation function can significantly improve the performance of the network [61]. Figure 6 shows several common activation functions. Among them, sigmoid and Tanh are called saturating nonlinearities. It can be seen from the Figure 6 and the formula definition that when the input is very large or very small, the Sigmoid function saturates at the output 0 or 1, and the Tanh function saturates at the output -1 or 1. To solve the problems caused by saturating nonlinearities, non-saturating nonlinearities such as ReLU [73], Leaky ReLU [74], PReLU [75], RReLU [76] and ELU [77] have been proposed. In terms of the time required for gradient descent training, the former functions are much slower than the latter, Neurons with non-saturating nonlinearities are called Rectified Linear Units (ReLUs). This type of deep convolutional neural network with ReLUs is several times faster than similar networks with tanh as the activation functions [73].

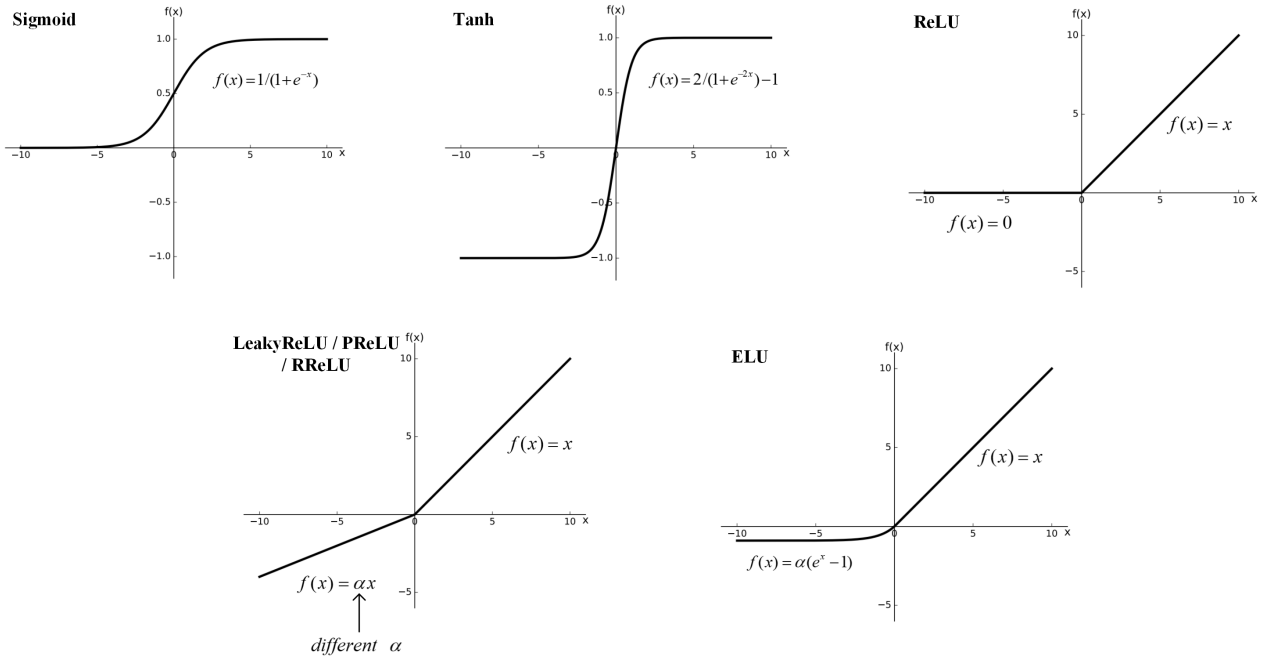


Figure 6. Pooling operation, it does not involve zero padding.

2.2.4. Fully Connected (FC) Layer

FC Layer is generally behind the continuous convolutional layer and pooling layer, and the neurons between its different layers are fully connected form. It integrates and classifies the local information with category discrimination extracted after convolution and pooling [78], and finally outputs the category information of the image. It contains several hidden layers, which extract high-level features from the previous network in a more complex form [77,79]. The number of neurons at the output end is the number of categories, and then the output vector is used to determine which category the image belongs to. In layman's terms, FC Layer acts as a classifier in CNNs.

Under network training, the network output is generally subjected to softmax regression [61] for probability normalization before the loss function of the FC layer. Of course, Cu et al. [61] also analyzed the impact of multiple loss functions on network performance. The parameters of the FC layer are updated using gradient backpropagation. When a large model with more parameters is trained on a smaller dataset, the FC layer generally uses L_2 regularization and dropout [79]. The fundamental purpose of using them is to avoid overfitting the model. The classic CNN model basically uses the ReLU plus dropout method and has achieved good classification performance [80,81].

2.2.5. Loss Function

In addition to the various layer-types of CNN architecture introduced in the previous section, the final classification is achieved from the output layer that usually the last layer of the FC layer, as shown in Figure 2. Different loss functions also affect the performance of the CNN architecture and are applied to different visual tasks (e.g., image classification, face recognition, and object recognition). Here are some commonly used loss functions in CNN-based image classification methods (inherit the content of Section 2.1.2), as shown in Table 1.

All in all, Softmax+Cross-Entropy has become the usual loss function of the CNN model. There are also many improved versions based on it, such as center-loss [82], L-Softmax [83], A-Softmax [84], AM-Softmax [85], PEDCC-loss [86], etc., which play an important role in different visual tasks.

Table 1. Common loss functions for CNN models.

Loss Function	Equation	Characteristic
L1 (MAE)	$Loss(y, y^*) = \frac{1}{m} \times \sum_{i=1}^{m} \left\| y_i^* - y_i \right\|$	This function is widely used in regression problems. L1 Loss is called mean absolute error (MAE)
L2 (MSE)	$Loss(y, y^*) = \frac{1}{m} \times \sum_{i=1}^{m} \left(y_i^* - y_i \right)^2$	This function is widely used in regression problems. L2 Loss is called mean square error (MSE)
Softmax + Cross-Entropy	$Loss(y, y^*) = -\sum_i \frac{y_i}{\sum_{i=1}^{m} y_i} \log\left(y_i^*\right), i \in [1, m]$	This function usually employed as a substitution of the MSE in multi-class classification problems. It is also commonly used in CNN models

2.2.6. Optimizer

The flow of data in the CNN architecture is basically introduced in the above section. We clearly understand that the training of the network relies on the core step of gradient update, that is, it needs to compute the objective function (loss function) gradient by applying a first-order derivative with respect to the network parameters, and then the gradient information is transferred to the previous network layer in the form of partial differential calculation to achieve the update of the learning parameters of each network layer. The function of the optimizer is to provide a way to make the gradient update more reasonable, namely the macroscopic performance is that the entire network may converge faster, smaller local optimal value (smaller loss), cheaper calculation, etc. Table 2 ummarizes several commonly used optimizers including methods and characteristics.

Table 2. Different optimizers for CNN model.

Name	Method	Characteristics
Batch Gradient Descent (BGD)	It calculates the gradient of the whole training set and subsequently uses this gradient to update the parameters.	1. For a small-sized dataset, the CNN model converges faster and creates an extra-stable gradient using BGD. 2. Generally not suitable fora large training dataset 3. It requires a substantial amount of resources.
Stochastic Gradient Descent (SGD)	It samples by arbitrarily selecting part of the training sample.	1. For a large-sized training dataset, this technique is both more memory-effective and much faster than BGD. 2. Randomness and noise are introduced due to its frequent updates. Its convergence is not stable, but the expectation is still equal to the correct gradient descent.
Mini-batch Gradient Descent	It partitionss the training samples into several mini-batches, and then parameter updating is performed following gradient computation on every mini-batch.	This method combines the technical advantages of SGD and SGD, which has a steady convergence, more computational efficiency and extra memory effectiveness.
Momentum	It introduces a momentum parameter λ into SGD that accumulates historical gradient information.	When training falls into a local minimum, the gradient information with momentum can help the network escape and find the global minimum
Adaptive Moment Estimation (Adam)	It calculates an adaptive learning rate for each parameter in the model	The advantages of momentum and RMSprop are combined. It is widely used in deep learning and represents the latest trend of optimization.

3. Image Classification Based on CNN

In general, image classification describes the whole image by manually extracting features or feature learning methods, and then uses the classifier to identify the object category. Therefore, how to extract the features of the image is especially important. Object classification based on Bag of Words model [87] is widely used before deep learning. The simplest Bag of Words model framework can be designed as three processes of low-level

feature extraction, feature coding, and classifier design. The traditional image classification method before 2012 can be completed in these three steps, but the complete establishment of image classification model generally includes several processes such as low-level feature learning [88–90], feature coding [91–94], spatial constraint, classifier design [95], and model fusion. This type of traditional image classification method was widely used in the image classification algorithm in the early PASCAL VOC [96]. NEC Lab won the championship with SIFT and LBP, two nonlinear encoders and SVM classifiers in ILSVRC 2010 [97].

However, the emergence of CNNs has made a series of breakthroughs in the field of image classification and has achieved excellent performance on large-scale visual tasks [19,24,77,81]. The great success of deep CNNs (DCNNs) is attributed to its strong feature learning ability [77]. Different from the traditional image classification method, the classification method based on CNN is an end-to-end learning process, only the original image is input, the training and prediction process are carried out in the network, and the result is finally output. This method abandons the method of manually extracting specific image features and breaks the bottleneck of traditional classification methods. This is also the biggest advantage of CNNs for image classification. This section mainly introduces the image classification model based on CNN and introduces the representative classic models one by one in the order of the timeline.

3.1. Classic CNN Models

3.1.1. LeNet Network

In 1998, Lecun et al. built the LeNet-5 model used to digitally classify different people and was superior to all other methods at the time [15]. It was also the first time that the backpropagation was used in the training of CNNs. The LeNet-5 model is the cornerstone of the development of deep learning and the source of inspiration for various models in the future.

The LeNet-5 network has 7 layers and contains approximately 60k parameters. As shown in Figure 7, the network is divided into two parts: convolution area and FC area. The basic unit of the convolution area is the convolutional layer (Conv) followed by the max-pooling layer (Pool), which is constituted by repeated stacking of the basic units of the convolution layer and the max-pooling layer. FC area contains three FC layers that each with fixed neurons, which are 120, 84, and 10 in order. This model uses the sigmoid [98] activation function and uses the softmax classifier in the output layer. When the output of the convolution area is passed into the FC layer area, the input layer of the FC area will flatten each feature map in the mini-batch. The vector length in each mini-batch is $channel \times height \times width$.

Although LeNet-5 can achieve good results in early MNIST, its performance on larger data sets is not satisfactory. First, neural network calculations are complex, and the calculation efficiency is low under the current hardware level. Secondly, the researchers did not have a lot of in-depth research in many fields such as parameter initialization and optimization algorithms, which caused the training of complex neural networks to be usually difficult. More than ten years after LeNet-5 was proposed, neural networks were once surpassed by other machine learning methods [99], such as support vector machines (SVM) [100].

3.1.2. AlexNet Network

In 2012, AlexNet constructed by Krizhevsky et al. turned out [19]. This network won the ILSVR 2012 with a huge advantage. It proved for the first time that the learned features can surpass the manually designed features, thus breaking the previous state of computer vision research in one fell swoop. Because the ability of a single GTX 580 GPU was limited at that time, it adopted cross-GPU parallel computing processing, which made the original AlexNet model architecture like the "columnar" CNN of Cireşan et al. [18].

The AlexNet network has 8 layers and contains about 60M parameters. It is very similar to the LeNet design concept, but there are significant differences. It can be seen in

Figure 8 that AlexNet contains 8 layers of transformations, including 5 layers of convolution and 2 layers of FC hidden layers, and 1 FC output layer. The final FC layer brings a huge amount of parameters to the model. The height and width of most images in ImageNet [101] are more than 10 times larger than those of MNIST images and occupy more pixels, so a larger convolution size in first layer is needed to extract object features. And the improvements of AlexNet are as follows:

- ReLU [73]. The activation function is changed from sigmoid to ReLU, it accelerates the model convergence and reduces the gradient disappearance.
- Dropout [79]. the model uses dropout to control the model complexity of the fully connected layer with $p = 0.5$ to alleviate the overfitting problem.
- Data augmentation. Introduced a large number of Data augmentation, such as flipping, cropping, and color changes, to further enlarge the datasets to alleviate the overfitting problem. Dropout and Data augmentation methods are widely used in subsequent convolutional neural networks.
- Overlapping pooling. There will be overlapping areas between adjacent pooling windows, which can improve model accuracy and alleviate overfitting.

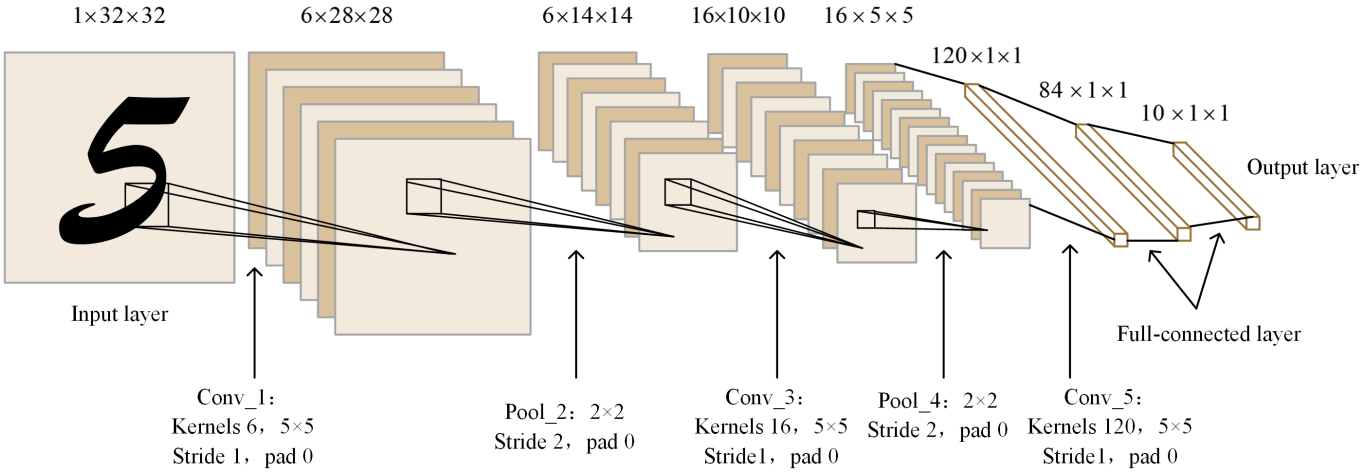


Figure 7. The architecture of the LeNet-5 network. The output shape is channel × height × width. Each convolutional layer uses size 5 × 5, padding 0, strides 1. Each pooling layer size 2 × 2 and strides 2.

3.1.3. VGGNet

In 2014, Simonyan et al. proposed the VGG model [24] and won the runner-up of ILSVR 2014. This model is similar to the AlexNet model, and also uses the structure of the convolution area followed by the FC area. The composition rule of the VGG module is to use several identical convolutional layers in succession followed by a maximum pooling layer, the convolutional layer keeps the input height and width unchanged, while the pooling layer halves it. The VGG network has a variety of different layer structure models, (Figure 9) is the VGG-16. It contains 16 weight levels, the network connects five blocks in series, and finally, two fully connected layers with 4096 and an output layer with 1000 classifications are connected.

Although the author of AlexNet made a lot of adjustments in the convolution size, the number of output channels, and the construction order, they did not provide regular ideas for the construction of the network. VGGNet gives the design to follow the idea, the improvements of AlexNet are as follows:

- Modular network. VGGNet uses a lot of basic modules to construct the model, this idea has become the construction method of DCNNs.
- Smaller convolution. A lot of 3 × 3 convolution filters are used on VGGNet, which can ensure that the depth of the network is increased, and the model parameters are reduced under the same receptive field compared with a larger convolution filter [102].

- Multi-Scale training. It first scales the input image to a different size $S \in (256, 512)$, and then randomly crops it to a fixed size of 224 × 224 and trains the obtained data of multiple windows together. This process is regarded as a kind of scale jitter processing, which can achieve the effect of data augmentation and prevent the model from overfitting [102].

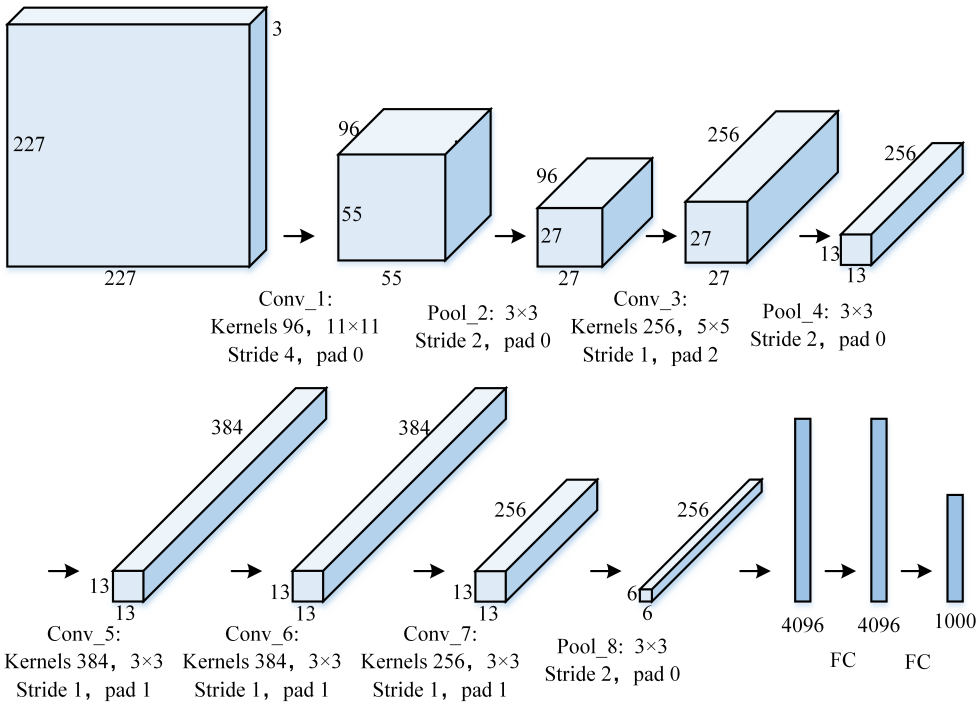


Figure 8. The architecture of the AlexNet network. The convolution size in the first layer is 11 × 11, the second layer is reduced to 5 × 5, and then all 3 × 3 is adopted. Conv_1, Conv_2, and Conv_5 layers are followed by a max-pooling layer with size 3 × 3 and strides 2. Finally, there are two fully connected layers of 4096 and an output layer of 1000 categories.

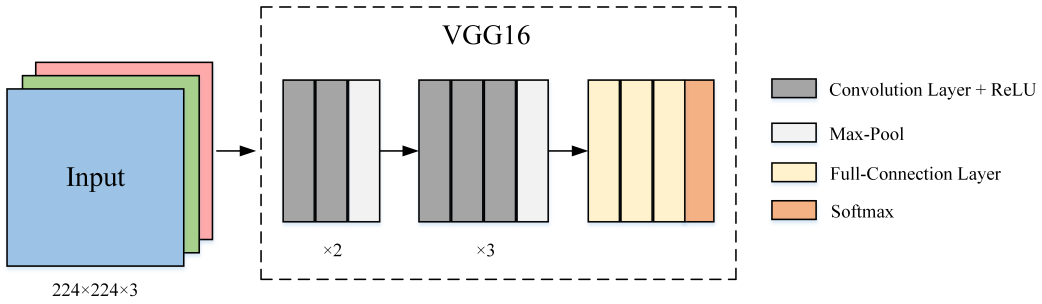
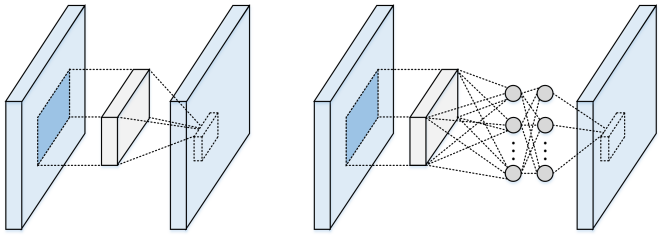


Figure 9. The architecture of the VGG-16 network. Conv: size = 3 × 3, stride = 1, padding = 1. Pool: size = 3 × 3, stride = 2.

3.1.4. Network in Network (NIN)

In 2014, Lin et al. proposed a network NIN model with a network-in-network structure [21]. Different from the linear filter used in the traditional convolutional layer plus the nonlinear activation function, the NIN model combines the MLP [103] with the convolution and uses a more complex structure of the micro neural network instead of the traditional convolutional layer. This new type of layer is called 'Mlpconv' (Figure 10).



(a) Linear convolution layer (b) Mlpconv layer

Figure 10. Comparison of linear convolution layer and mlpconv layer.

The difference between the NIN network and LeNet, AlexNet, and VGG is concatenating multiple Mlpconv composed of convolutional layers and MLP to build a deep network. For deep convolutional neural networks, the convolutional layer to achieve a good abstract representation usually requires the input data to be highly non-linear. The filter is a generalized linear model (GLM) for the low-level data, and the abstraction of GLM is low [104]. Higher-level filters combine lower-level concepts to generate higher-level abstract concepts. And MLP has a strong ability to express nonlinear functions, replacing GLM with a more effective function approximator—MLP can enhance the abstract expression ability of local models. The author believes that better abstract processing of each partial module before combining it into higher-level concepts is beneficial to the network, which constructs the mlpconv micro-network.

NIN was proposed shortly after the advent of AlexNet, and their convolutional layer settings are similar. But in NIN, those different designs and contributions are summarized as follows:

- Mlpconv. MLP layer is equivalent to a 1×1 convolutional layer. Now, it is usually used to adjust the channels and the parameters, and there are also explanations that cross-channel interaction and information integration are possible.
- Global average pooling (GAP). The FC layer is no longer used for output classification, but a micro-network block with the number of output channels equal to the number of label categories is used, and then all elements in each channel are averaged through a GAP layer to obtain the classification confidence.

Ref. [21] pointed out that the reason why the model uses GAP is that it is more interpretable and more meaningful than the fully connected layer. In addition, the fully connected layer is prone to overfitting due to too many parameters, and it relies too much on dropout regularization. The GAP can be regarded as a structural regularization method and replacing it with the FC layer can greatly reduce the amount of model parameters and effectively prevent the model from overfitting.

3.2. GoogLeNet/InceptionV1 to V4

In 2014, GoogLeNet [23] proposed by Christian Szegedy et al. won the ILSVR2014 championship. This model absorbs the idea of NIN and the theoretical work of Arora et al. [105], and introduces the concept of the Inception module. In the following years, researchers made several improvements to the Inception module, and the performance of the model also improved.

3.2.1. InceptionV1

GoogLeNet has 22 layers, including about 6M parameters. The basic module of the network is the Inception module (Figure 11a). This module contains 4 parallel branches. The first three branches use convolutional layers with different sizes to extract information under different spatial sizes. Among them, 1×1 convolution can reduce the number of channels and compress information to reduce model complexity. The max-pooling effect of the last branch is to reduce the resolution, followed by 1×1 convolution to adjust the depth after the pooling. All in all, this unique design increases the width of the network

model and the adaptability to different scales or even resolutions, achieving the effect of multi-scale fusion. The GoogLeNet model is similar to VGGNet, and its convolution part also uses modular splicing.

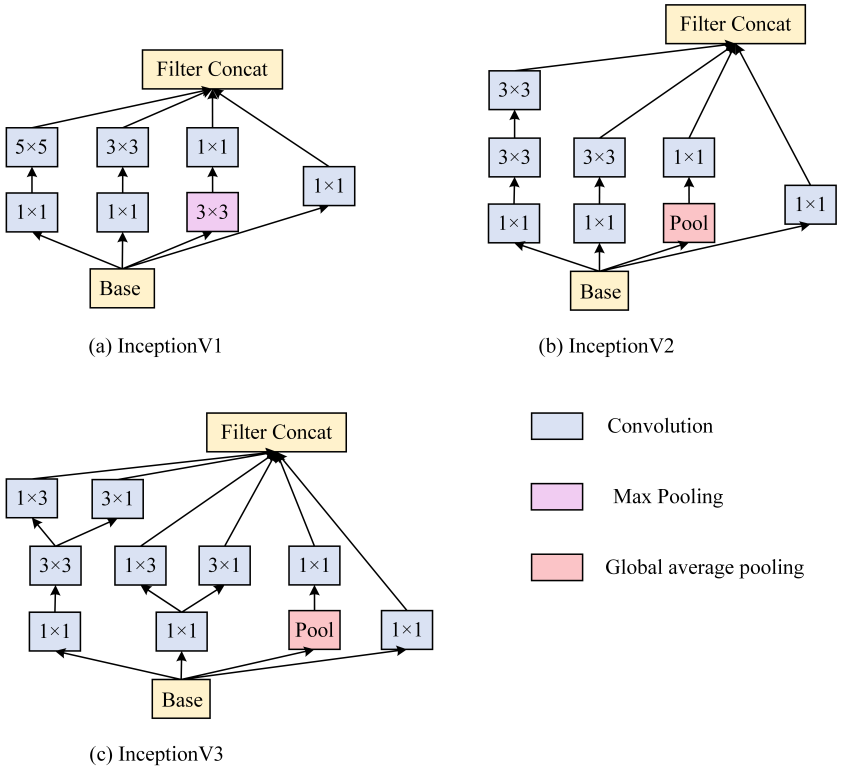


Figure 11. InceptionV1 to V3 module.

The most direct way to improve network performance is to increase the network depth and network width (the number of neurons in each layer), but the following disadvantage is that as the network size increases, the number of parameters increases, which makes the network more prone to overfitting, and the usage of computing resources will increase dramatically. Ref. [23] believes that the fundamental way to solve the above shortcomings is to transform the fully connected layer and even the convolutional layer into sparse connections. First, the connection of the biological nervous system is also sparse. Second, the main research results of Arora et al. [105] show that if the probability distribution of the dataset is representable by a large, very sparse deep neural network, then the optimal network topology can be constructed layer by layer by analyzing the correlation statistics of the activations of the last layer and clustering neurons with highly correlated outputs. Therefore, the progress made by inceptionV1 lies in the following:

- Inception module. Although the early traditional neural networks used random sparse connections, computer hardware was inefficient in computing non-uniform sparse connections. The proposed Inception module can not only maintain the sparsity of the network structure but also use the high computational performance of the dense matrix, thereby effectively improving the model's utilization of parameters.
- GAP. Replaced the fully connected layer to reduce the parameters.
- Auxiliary classifier. An auxiliary classifier used for a deeper network is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss.

As a result, the parameters of GoogLeNet are only 1/12 of AlexNet, but the performance is greatly improved.

3.2.2. InceptionV2

Compared with inceptionV1, the improvements of inceptionV2 [106] are as follows:

(1) Smaller convolution. The 5×5 convolution is replaced by the two 3×3 convolutions. This also decreases computational time and thus increases computational speed because a 5×5 convolution is 2.78 more expensive than a 3×3 convolution.

(2) Batch Normalization (BN). BN is a method used to make ANNs faster and more stable through normalization of the layers' inputs by re-centering and re-scaling for each mini-batch.

In CNN, BN is achieved through a normalization step that fixes the means and variances of each layer's inputs. Ideally, the normalization would be conducted over the entire training set, but to use this step jointly with stochastic optimization methods, it is impractical to use the global information. Thus, normalization is restrained to each mini-batch in the training process. For a layer with d-dimensional input $x = \left(x^{(1)} \cdots x^{(d)}\right)$, It will normalize each dimension. And use $B = x(i \ldots m)$ to denote a mini-batch of size m of the entire training set. *BN* Transform thus be denoted as:

/mini-batch mean: $\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i$

/mini-batch variance: $\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2$

/normalize: $\hat{x}_i = (x_i + \mu_B) / \sqrt{\sigma_B^2 + \epsilon}$

/scale and shift: $y_i = \gamma \hat{x}_i + \beta = BN_{\gamma,\beta}(x_i)$

The parameters γ, β to be learned in the optimization process, ϵ is a constant added to the mini-batch variance for numerical stability.

3.2.3. InceptionV3

Inception v3 [1] mainly focuses on burning less computational power by modifying the previous Inception architectures. Its main improvements are as explained below:

- Factorized convolutions. This helps to reduce the computational efficiency as it reduces the number of parameters involved in a network. It also keeps a check on the network efficiency. This part contains the following (2) and (3).
- Smaller convolutions. replacing bigger convolutions with smaller convolutions definitely leads to faster training.
- Asymmetric convolutions. A 3×3 convolution could be replaced by a 1×3 convolution followed by a 3×1 convolution. The number of parameters is reduced by 33%.
- Grid size reduction. Grid size reduction is usually done by pooling operations. However, to combat the bottlenecks of computational cost, a more efficient technique is proposed. Say for example in the Figure 12, 320 feature maps are done by conv with stride 2. 320 feature maps are obtained by max pooling. And these 2 sets of feature maps are concatenated as 640 feature maps and go to the next level of inception module.

3.2.4. InceptionV4

The main aim of Inception V4 [107] was to reduce the complexity of the Inception V3 model [1] which made a unified choice for each Inception block. Inception blocks include Inception modules and Reduction modules as shown in Figure 13. Figure 14 for the overall architecture of the InceptionV4. All the convolutions not marked with "V" in the figures are same-padded meaning that their output grid matches the size of their input. Convolutions marked with "V" are valid padded, meaning that the input patch of each unit is fully contained in the previous layer and the grid size of the output activation map is reduced accordingly.

- The initial set of layers to which the paper refers "stem of the architecture" (Figure 14) was modified to make it more uniform. These layers are used before the Inception block in the architecture.
- This model can be trained without partition of replicas unlike the previous versions of inceptions which required different replicas to fit in memory. This architecture uses memory optimization on backpropagation to reduce the memory requirement.

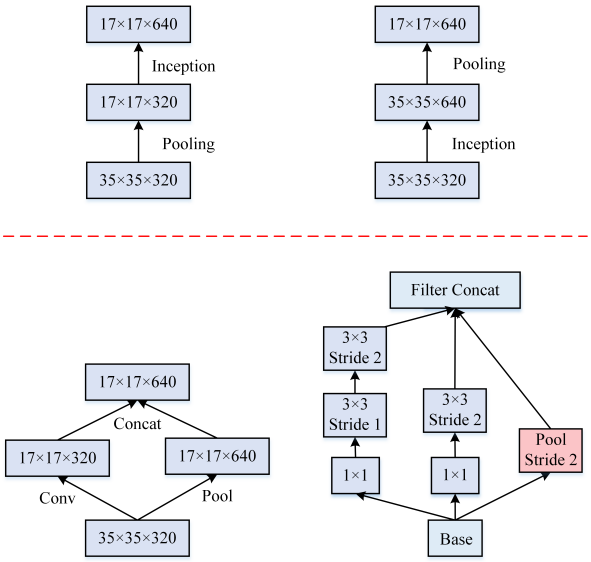


Figure 12. Two methods on the red line: the solution on the left violates the principle [1]. The version on the right is 3 times more expensive computationally. Method under the red line: an efficient grid size reduction module is both cheap and avoids the representational bottleneck as is suggested by principle [1].

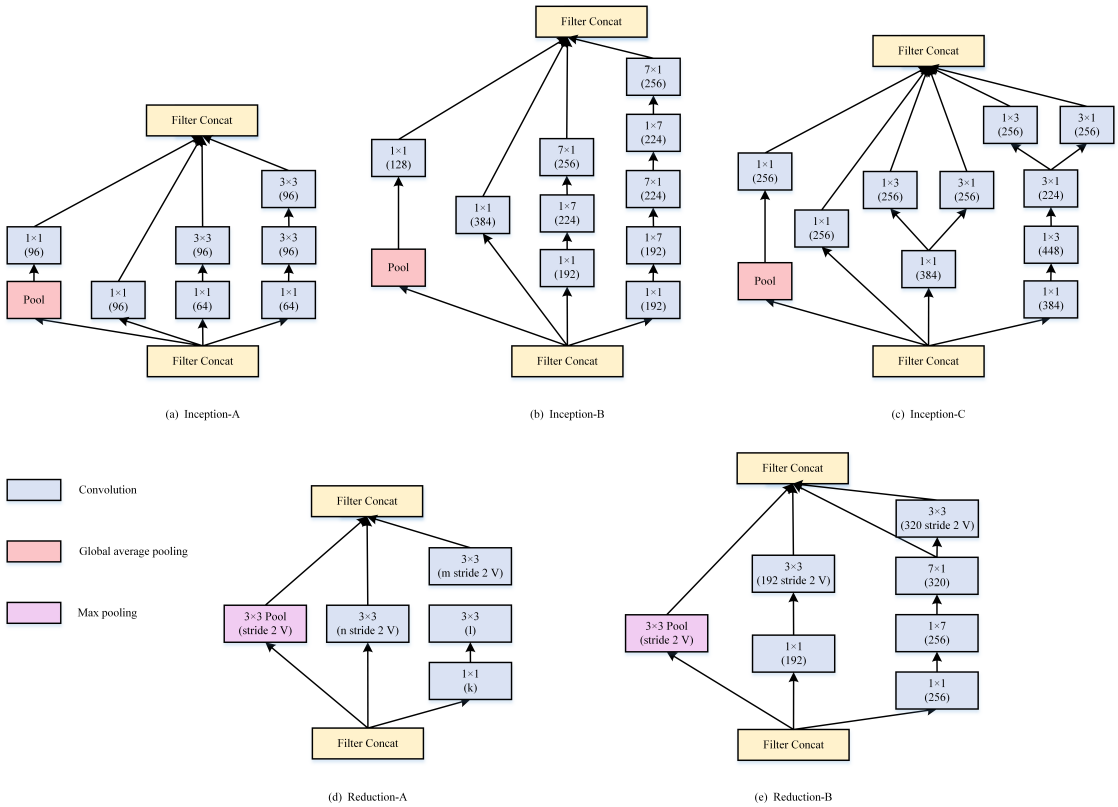


Figure 13. InceptionV4 blocks. It contains the Inception-A/B/C modules and Reduction-A/B modules.

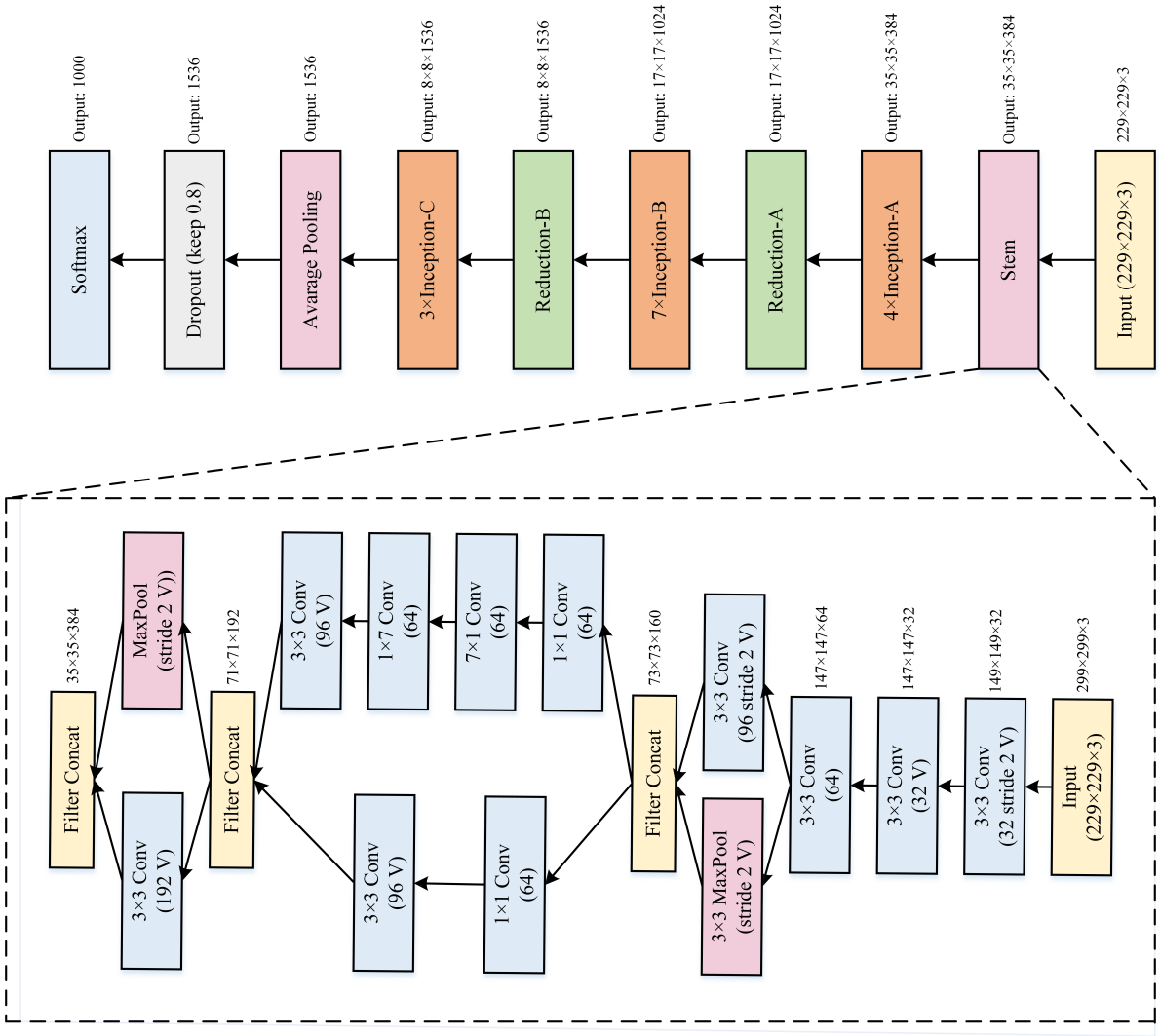


Figure 14. Overall Architecture of InceptionV4. The upper part of the picture is the overall structure, the lower part of the picture is the Stem of the architecture.

3.3. Residual Learning Networks

3.3.1. ResNet

In 2015, the deep residual network ResNet proposed by KaimingHe et al. [25] won the first prize in ILSVR2015. Looking back at the network development described earlier, deeper and deeper networks are a common development trend, that is, increasing the depth of the network will increase the network performance. However, many experiments have shown that simply increasing the network depth within a certain depth range cannot effectively improve network performance [108]. Another experiment shows that the increase in the number of network layers within 20 layers brings about the improvement of network performance, but if the deep network with more than 20 layers continues to overlay the number of network layers, the classification accuracy will decrease instead [109].

We may blindly point the finger at the problem of disappearance/exploding gradients [110,111] or the overfitting. However, networks with dozens of layers can easily converge to the backpropagation of stochastic gradient descent through initial normalization [75,110,112] and batch normalization [106]. The article [25] verifies that network degradation is not overfitting. In fact, this situation is that the deep network cannot simply be optimized well—the optimization of Stochastic Gradient Descent (SGD) [113] becomes difficult. This phenomenon that the accuracy does not increase but decreases is called "degradation". It has seriously affected the training of deep nonlinear networks. The

residual connection in ResNet is a method to break "degradation" and enable deep neural networks to achieve high accuracy [114]. Based on the residual vector coding representation of VLAD [115] and Fisher Vector [116] and the research of shortcut connection theory and practice, the residual network makes the network of the stacked layers' optimal state continue to accumulate multiple identity mapping layers. This kind of residual connection can increase the depth of the network while facilitating optimization, and the accuracy is also increasing [25].

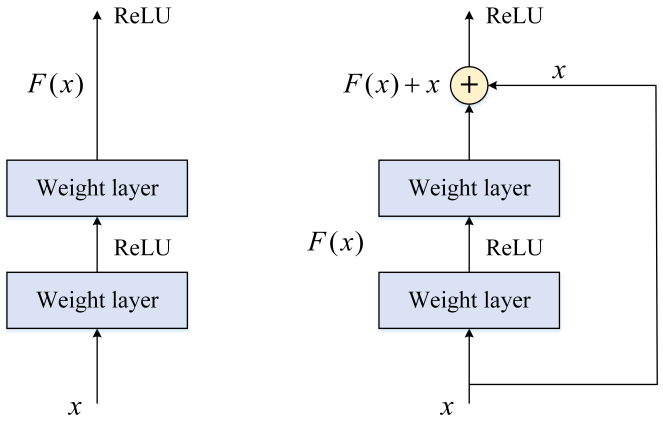
Let us focus on the residual connection layer of the residual network, as shown in Figure 15. $H(x)$ is the ideal mapping we want, the left part of Figure 15 is an ordinary CNN learning, which needs to be directly fitted to the mapping $H(x)$. The residual learning on the right is to let the residual block not directly learn the target mapping but to fit a residual mapping related to the identity mapping $F(x) = H(x) - x$. Assuming that the network of a certain depth tends to be saturated, to ensure the parameter update and gradient propagation of the next layer, only the weight and bias of $F(x)$ need to be updated to 0, and then the identity mapping $H(x) \to x$ can ensure that the input of the next layer is at least the same as the output of the previous layer. In fact, when the ideal map $H(x)$ is very close to the identity map, the residual map is also easy to capture the subtle fluctuations of the identity map. Of course, the non-linear mapping $F(x)$ is much easier to learn than the direct fitting, which allows the input value to propagate forward faster through the cross-layer data line.

The residual block contains two 3×3 convolutional layers with the same number of channels, and each convolutional layer is followed by a BN and a ReLU activation function. Another branch connects the input directly to the last ReLU by skipping the convolutional layer—a building block as in Figure 16 (left) for ResNet-34. When the network stack is deep, a 1×1 convolution layer can be added after the 3×3 convolution layer to control the number of channels—a "bottleneck" building block as in Figure 16 (right) for ResNet-50/101/152.

ResNet can be said to stand at the very gate of DCNNs in a true sense. By using ResNet, its important contributions are as follows:

Residual learning

- This method is easy to optimize, but the "plain" networks (that simply stack layers) show higher training error when the depth increases.
- It can easily gain accuracy from greatly increased depth, producing results that are better than previous networks.



(a) Ordinary CNN learning (b) Residual learning

Figure 15. Comparison of ordinary CNN learning and residual learning.

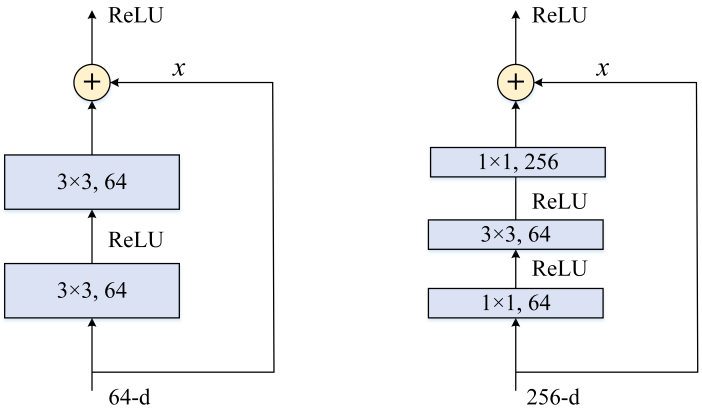


Figure 16. Two building blocks for ResNet.

3.3.2. Improvement of ResNet

CNNs with hundreds of layers or more are indeed very competitive, but very DCNNs have the challenge of difficulty in training and the risk of overfitting. In the case of limited data sets, researchers have made improvements to the ResNet's building block are as follows:

1. **ResNet with Pre-activation.** He et al. [109] proposed a pre-activation structure to pre-activate the BN and ReLU to further improve the network performance. Several experiments were carried out on the layout of BN and ReLU and the best performing structure was obtained in Figure 17(right). It can successfully train ResNet with more than 1000 layers. At the same time, they also proved the importance of identity mapping compared to other shortcut connections.
2. **Stochastic depth.** The authors of [117] pointed out that there are many layers in the ResNet network that contribute little to the output result. In the network training process, the Stochastic depth method is used, and deleting some layers can greatly shorten the training time and effectively increase the depth of ResNet, even exceeding 1200 layers. The test error and the training time on CIFAR-10/100 still has a good improvement.
3. **Wide Residual Networks (WRNs).** With the increasing depth of residual networks, the diminishing feature reuse will make the training of the network very slow [118]. To alleviate this problem, ref. [119] introduced a wide-dropout block that widens the weight layer of the original residual unit [25] Figure 15 (right) and adds dropout between the two weight layers. Compared with deeper ResNet, WRN with fewer layers greatly reduces the training time and has better performance on the CIFAR&ImageNet data set.
4. **ResNeXt [26].** Although Inception and ResNet have great performance, but these models are well-suited for several datasets. Due to the many hyperparameters and computations involved, adapting them to new datasets is no minor task. A new dimension "Cardinality *C*"—the number of paths in a block—is used to overcome this problem, and experiments demonstrate that increasing cardinality *C* is more effective than going deeper or wider when we increase the capacity. The authors compared the completely equivalent structures of the three mathematical calculations in Figure 18. The experimental results show that block Figure 18c with grouped convolution is more succinct and faster than the other two forms, and ResNeXt uses this structure as a basic block.
5. **Dilated Residual Networks (DRN).** To solve the decrease in the resolution of the feature map and the loss of feature information caused by downsampling. However, simply removing subsampling steps in the network will reduce the receptive field. So, Yu et al. [120] introduced dilated convolutions that are used to increase the receptive field of the higher layers and replaced a subset of the internal downsampling layer

based on the residual network, compensating for the reduction in receptive field induced by removing subsampling. Compared to ResNet with the same parameter amount, the accuracy of DRN is significantly improved in image classification.

6. **Other models.** Veit et al. [121] drops some of the layers of a trained ResNet and still have comparable performance. Resnet in Resnet (RiR) [122] proposed a deep dual-stream architecture that generalizes ResNets and standard CNNs and is easily implemented with no computational overhead. DropBlock [123] technique discards feature in a contiguous correlated area called block, which is a regularization helpful in avoiding the most common problem data science professionals face i.e., overfitting. Big Transfer (BiT) [124] proposes a general transfer learning method to be applied to ResNet, which uses the minimal number of tricks yet attains excellent performance on many tasks. NFNet [125] proposes a ResNet-based structure without BN layer, by using adaptive gradient clipping technique to achieve amazing training speed and accuracy.

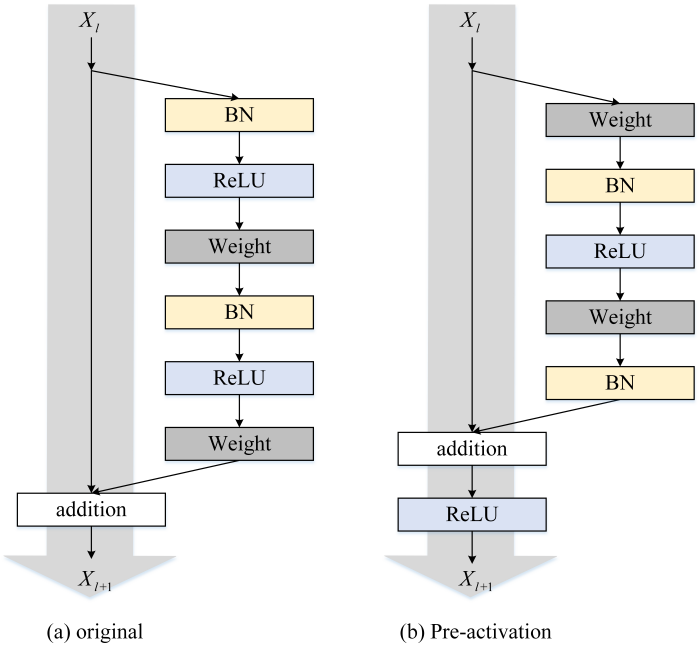


Figure 17. (**a**) original Residual Unit [25]. (**b**) Residual Unit with full pre-activate [109].

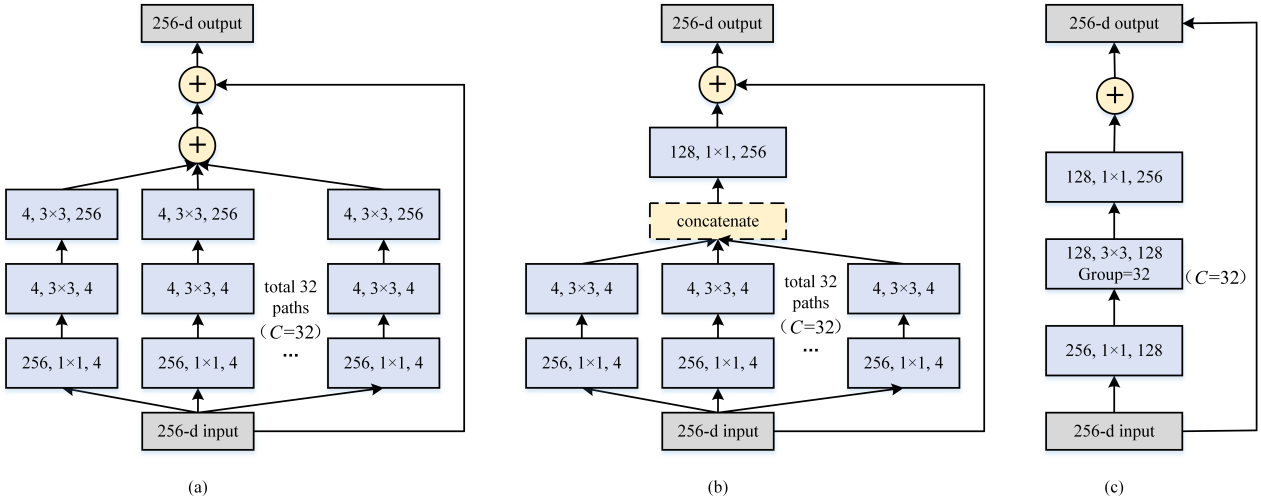


Figure 18. (**a**–**c**) Equivalent building blocks of ResNeXt.

3.3.3. ResNet with Inception

From 2014 to 2017, the residual method and the Inception method have a strong dominance in image classification tasks. Researchers combined the two structures and added new methods to achieve better performance, these classic variants deserve to be discussed as follows:

1. **Inception-ResNet [107].** Tried to combine the Inception structure with the residual structure and achieved good performance. It comes from the same paper as inceptionV4 [107], and the combination is Inception-ResNet-V1/V2 as shown in Figures 19 and 20. Inception-ResNet-V1 has roughly the computational cost of Inception-V3 [1], and it was training much faster but reached slightly worse final accuracy than InceptionV3. Inception-ResNet-V2 has roughly the computational cost of Inception-v4, and it was training much faster and reached slightly better final accuracy than InceptionV4 [107].
2. **Xception [126].** It is based on the design point of InceptionV3 [1]. The author believes that the correlation between channels and spatial correlation should be handled separately, using modified depthwise separable [127] convolution to replace the convolution operation in InceptionV3. Refs. [128,129] also show that using separable convolution can reduce the size and computational cost of CNNs. But the modification in Xception aims to improve performance. The accuracy of Xception on ImageNet is slightly higher than that of Inception-v3, while the parameters is slightly reduced. The experiment in [126] also shows that the residual connection mechanism similar to ResNet added to Xception can significantly speed up the training times and obtain a higher accuracy rate.
3. **PolyNet [130].** Many studies tend to increase depth and width in image classification tasks to obtain higher performance. But very deep networks will have trouble that is a diminishing return and increased training difficulty. A quadratic growth in both computational cost and memory demand is caused by a widening network. This method explores the structural diversity of Inception and ResNet that a new dimension beyond just depth and width, which introduced a better-mixed model from the perspective of polynomials.

3.3.4. DenseNet

DCNNs often face the dilemma of gradient disappearance and degradation, and network training has become a problem. The improved residual network described above proposes some solutions, but here we must mention the excellent work DenseNet [131]. It has the same direction as ResNet [25] and the Highway network [118]. The connection of traditional convolution is between each layer and the next layer. In DenseNet each layer connects to every other layer in a feed-forward fashion. In this way, each layer has direct access to the gradients from the loss function and the original input signal, leading to implicit deep supervision.

Dense Blocks are shown in Figure 21, each layer obtains additional inputs from all preceding layers, and passes its own feature maps to all subsequent layers. Simply expressed as $x_l = H_l([x_0, x_1, \cdots, x_{l-1}])$, and the residual connection is $x_l = H_l(x_{l-1}) + x_{l-1}$. It also setup growth rate k indicates the added number of input channels when pass through a layer. Ref. [131] pointed out "BN-ReLU-Conv(1×1)-BN-ReLU-Conv(1×1)" called "Bottleneck layers" very effective for DenseNet, as DenseNet-B. In order to require the same size of feature maps, "Transition layers"—1×1 convolution reduce the number of feature-maps by $\theta \in (0, 1)$—are set between different dense blocks to achieve down sampling, as DenseNet-C. Both Bottleneck layers and Transition layers are called DenseNet-BC, of course its performance is the best.

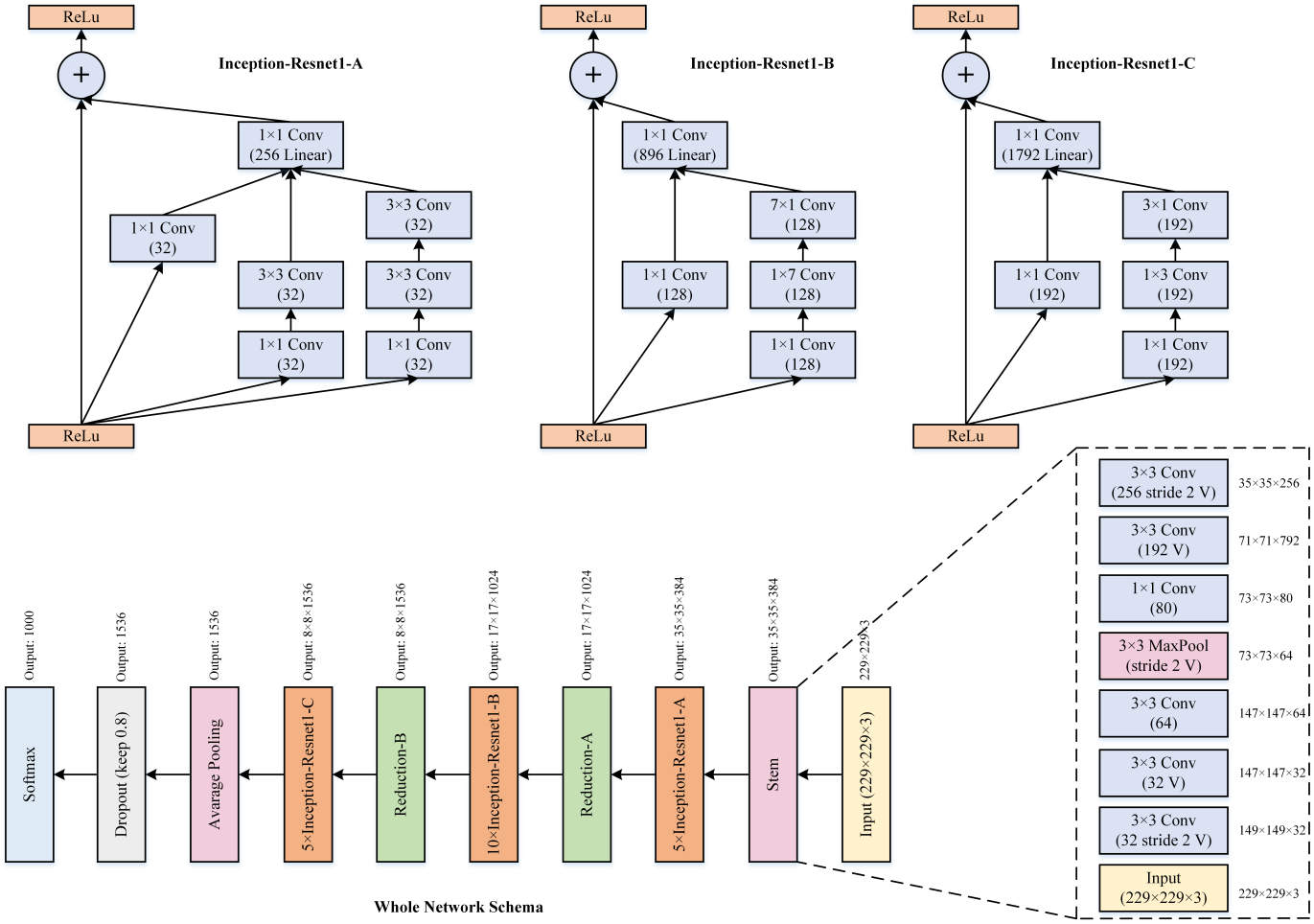


Figure 19. Inception-ResNet-V1.

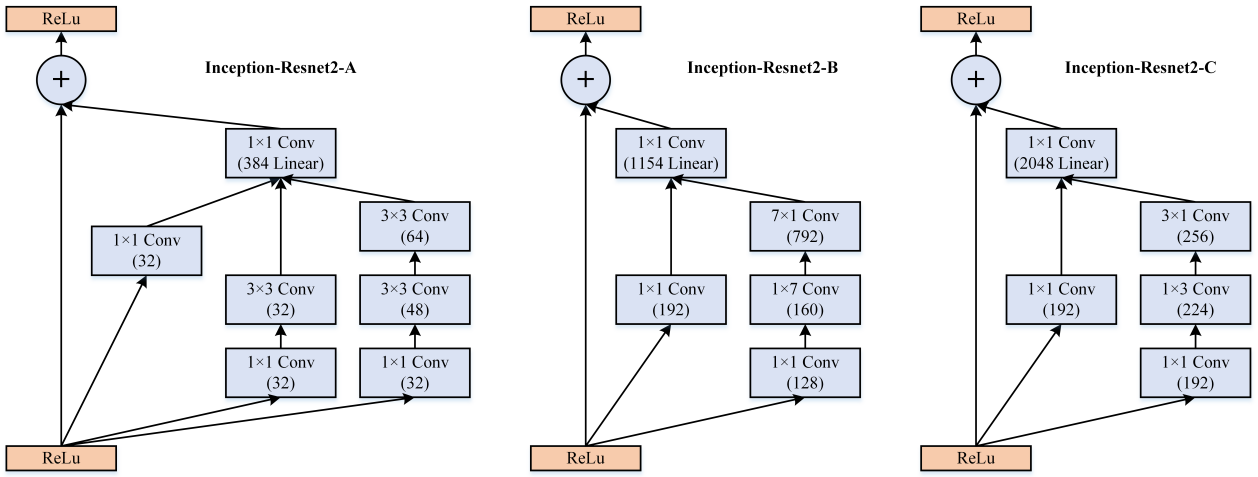


Figure 20. Inception-ResNet-V2. The number of parameters increase in some layers in comparison to Inception-ResNet-V1.

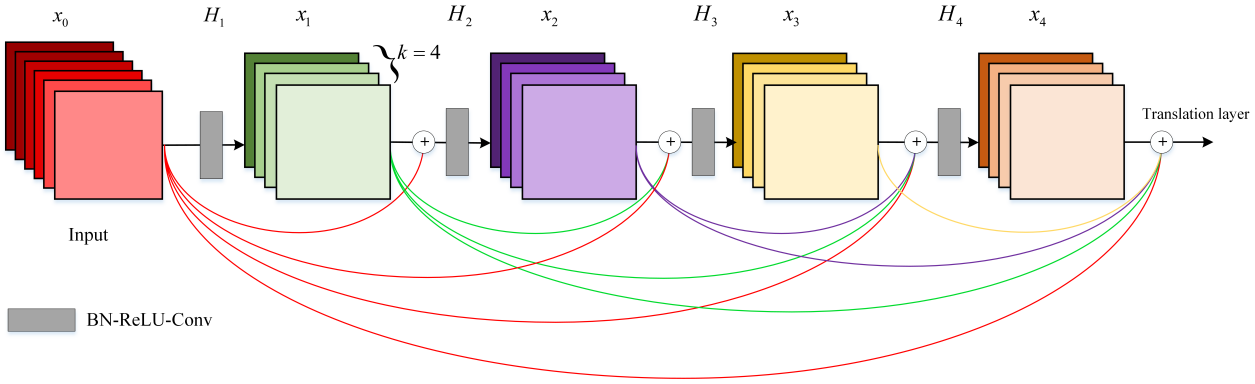


Figure 21. Dense block. 5-layer with a growth rate of $k = 4$.

It is worth mentioning that the Dual Path Networks (DPN) proposed by Yan et al. [132] explored the relationship between residual learning [25] and dense connection [131], and combined their advantages. It can also be said that the mathematical expression is unified between them by DPN. CondenseNet [133] mainly optimizes DenseNet [131] through group convolution operation and pruning during training, to achieve higher computational efficiency and fewer parameters.

3.4. Attention Module for CNNs

Attention mechanism can be explained intuitively using the human visual mechanism. Such as our visual system tends to pay attention to part of the information in the image for auxiliary judgment and ignore irrelevant information. There is a type of model that absorbs this idea to improve the performance of the CNN model, by using channel attention or spatial attention. They can be regarded as small-scale improvements of the entire model that can be transplanted to any feasible model.

3.4.1. Residual Attention Neural Network

To enable the network to learn aware features of the object, Wang et al. [134] proposed the Residual Attention Network (RAN or Attention) to incorporate the attention mechanism into CNN. The main structure of RAN is stacked by residual blocks. The overall architecture of RAN is shown in the Figure 22.

There are two branches in Residual Attention Network: Trunk branch which is the upper branch in the attention module for feature extraction can be Pre-Activation ResNet block or other blocks, with input x and output $T(x)$; Mask branch uses bottom-up top-down structure [3,135–137] to learn the same-size mask $M(x)$. The output of Attention Module H is: $H_{i,c}(x) = M_{i,c}(x) * T_{i,c}(x)$, where i ranges over the spatial locations, and c is the channel index from 1 to C. the attention mask can serve as a feature selector during forward inference, it also as a gradient update filter during backpropagation. In the soft mask branch, the gradient of mask for input feature is: $(\partial M(x,\theta)T(x,\theta))/\partial\phi = M(x,\theta)\partial T(x,\phi)/\partial\phi$, where θ are mask branch parameters and ϕ are trunk branch parameters. However, naive stacking Attention Modules will cause performance degradation. This is because dot production with mask ranges from zero to one repeatedly will degrade the value of features in deep layers, and soft mask can potentially break good property of trunk branch. A better mask is constructed as $H_{i,c}(x) = (1 + M_{i,c}(x)) * F_{i,c}(x)$, which is called Attention Residual Learning. Where $F(x)$ is the original features and $M(x))$ ranges from $[0, 1]$. '*' indicates element-wise product. A bottom-up top-down fully convolutional structure is used in soft mask branch. The activation function uses mixed attention which is simple sigmoid for each channel and spatial position. The positive effects of RAN are as follows:

- Stacking multi-attention modules has made RAN very effective at recognizing noisy, complex, and cluttered images.
- RAN's hierarchical organization gives it the capability to adaptively allocate a weight for every feature map depending on its importance within the layers.

- Incorporating three distinct levels of attention (spatial, channel, and mixed) enables the model to use this ability to capture the object-aware features at these distinct levels.

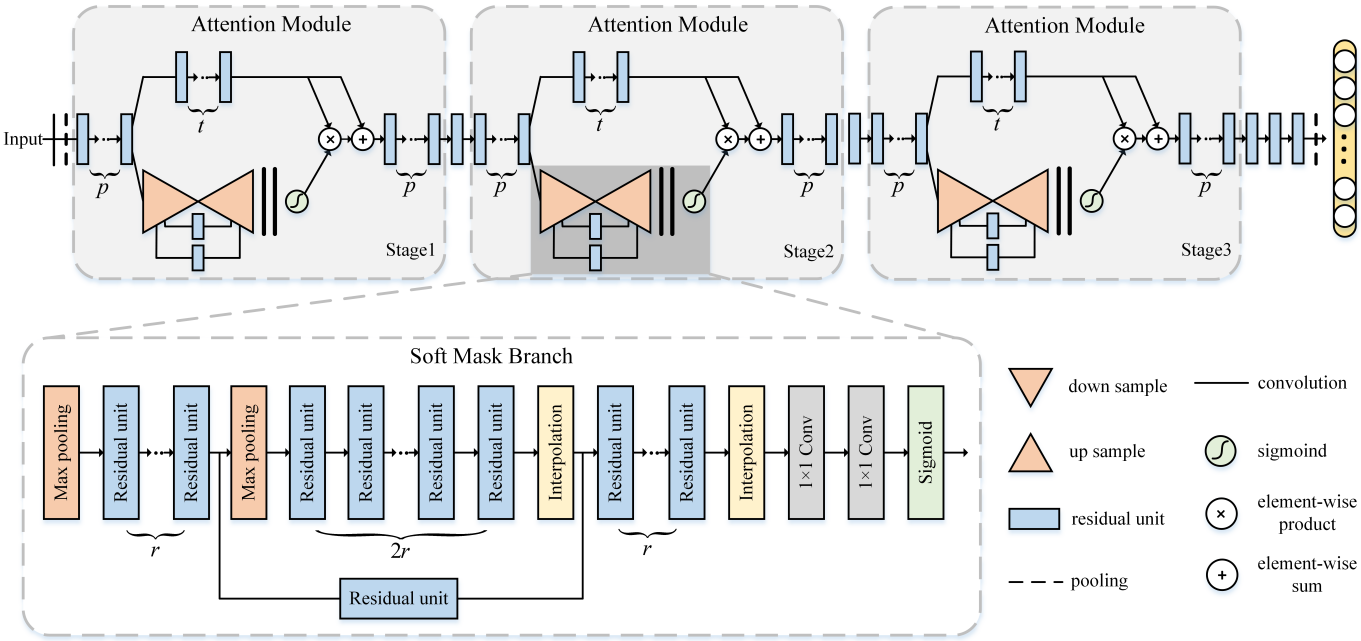


Figure 22. The architecture of Residual Attention Network. There are three hyper-parameters p, t, r. p is the number of pre-processing Residual Units before splitting into trunk branch and mask branch. t denotes the number of Residual Units in trunk branch. r denotes the number of Residual Units between adjacent pooling layer in the mask branch. Ref. [134] set the hyperparameter to $p = 1, t = 2, r = 1$.

3.4.2. SENet

In 2017, Wang et al. [27] proposed Squeeze-and-Excitation Networks (SENet) introduce a building block for CNNs that improves channel interdependencies at almost no computational cost, and it is also the champion of ILSVR2017 on the image classification task. SE module used channel-attention mechanism can be added to any baseline architecture to get an improvement in performance, with negligible computational overhead. The schematic diagram of this architecture is shown in Figure 23.

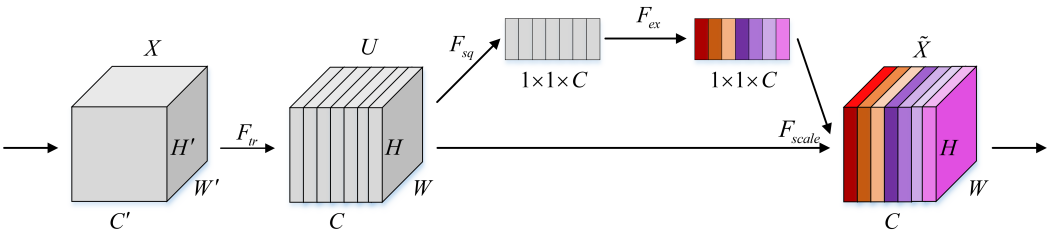


Figure 23. A Squeeze-and-Excitation (SE) block.

Usually, the network weights each of its channels equally when creating the output feature maps. SE block is all about changing this by adding an attention mechanism to weight each channel adaptively. First, the input $X \in \mathbb{R}^{H' \times W' \times C'}$ is mapped to $U \in \mathbb{R}^{H \times W \times C}$ through the transformation F_{tr}. Afterwards, they get a global understanding of each channel by *squeeze* F_{sq} the feature maps to a vector $1 \times 1 \times C$, where C is equal to the number of channels.Then channel-wise dependencies can be completely captured by *excitation* $F_{ex} : 1 \times 1 \times C \rightarrow 1 \times 1 \times C / \quad r \rightarrow 1 \times 1 \times C$, where r is reduction ratio. Finally, these C values can now be used as weights on the original features maps by F_{scale} to get $\tilde{X}$, scaling each channel based on its importance.

3.4.3. BAM and CBAM

The importance of the feature map utilization and the attention mechanism is certified via SENet [27] and RAN [134]. Woo et al. proposed Bottleneck Attention Module [138] (BAM) and Convolutional Block Attention Module [139] (CBAM) that *channel attention* module and *spatial attention* module are introduced. The two models will be introduced below because of their simple structure and similar ideas.

BAM. It gets an attention map through two separate paths: *channel attention* and *spatial attention*, as shown in Figure 24. For the given input feature map $F \in \mathbb{R}^{H\times W\times C}$, BAM infers a 3D attention map $M \in \mathbb{R}^{H\times W\times C}$. The refined feature map F' is computed as: $F' = F + F * M(F)$. To design an efficient module, they first compute the *channel attention* $M_c(F) \in \mathbb{R}^{1\times 1\times C}$ and the spatial attention $M_s(F) \in \mathbb{R}^{H\times W\times 1}$ at two separate branches, and attention map $M(F) = sigmoid(M_c(F) + M_s(F))$, $M(F) \in \mathbb{R}^{H\times W\times C}$. Finally, multiply the attention map $M(F)$ with the original feature map F to get $F * M(F)$. The above description takes the ResNet block [25] as the baseline.

CBAM. It gets an attention map through two concatenated paths: *channel attention* and *spatial attention*, as shown in Figure 25. For the given input feature map $F \in \mathbb{R}^{H\times W\times C}$, CBAM sequentially infers a 1D *channel attention* map $M_c(F) \in \mathbb{R}^{1\times 1\times C}$ and a 2D *spatial attention* map $M_s(F) \in \mathbb{R}^{H\times W\times 1}$. During multiplication, the attention values are broadcasted accordingly—*channel attention* values are broadcasted along the *spatial* dimension. The channel-refined feature map F' and channel-spatial-refined feature map F'' are summarized as: $F' = M_c(F) * F, F'' = M_s(F') * F'$.

In simple terms, BAM uses parallel connection and CBAM uses series connection for *channel attention* and *spatial attention*. However, CBAM is slightly better for image classification in actual performance.

3.4.4. GENet

In 2018, Hu et al. [140] introduced feature context to CNNs, which includes two operations: Gather ξG and Excite ξE, as shown in Figure 26. For the given input feature maps $X \in \mathbb{R}^{H\times W\times C}$, it is transformed by gather operator $\xi G : X \in \mathbb{R}^{H\times W\times C} \to \hat{X} \in \mathbb{R}^{H'\times W'\times C}$ $(H' = [H/e], W' = [W/e])$, where e is an extent ratio. Excite operator ξE uses nearest neighbor interpolation to resize $\hat{X} \to \mathbb{R}^{H\times W\times C}$, then after sigmoid, $\hat{X}$ and the original input X perform element-wise product operation: $\xi E(X, \hat{X}) = X * f(\hat{X})$, where $f : \mathbb{R}^{H'\times W'\times C} \to [0,1]\mathbb{R}^{H\times W\times C}$. This method can be added to the baseline architecture for better performance with slightly increasing parameters.

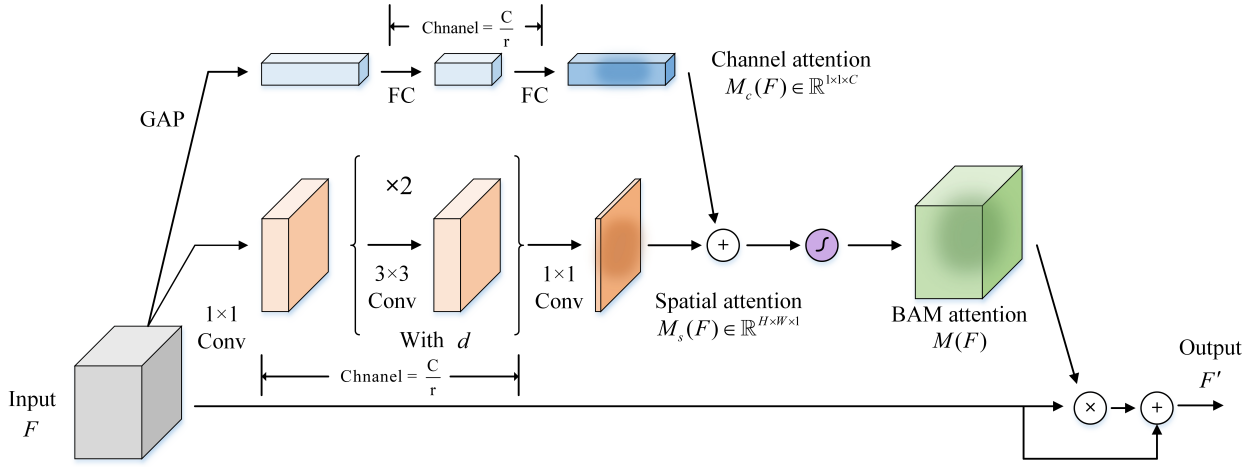


Figure 24. BAM module architecture. Two hyper-parameters d, r for this module, ref. [138] set $d = 4, r = 16$.

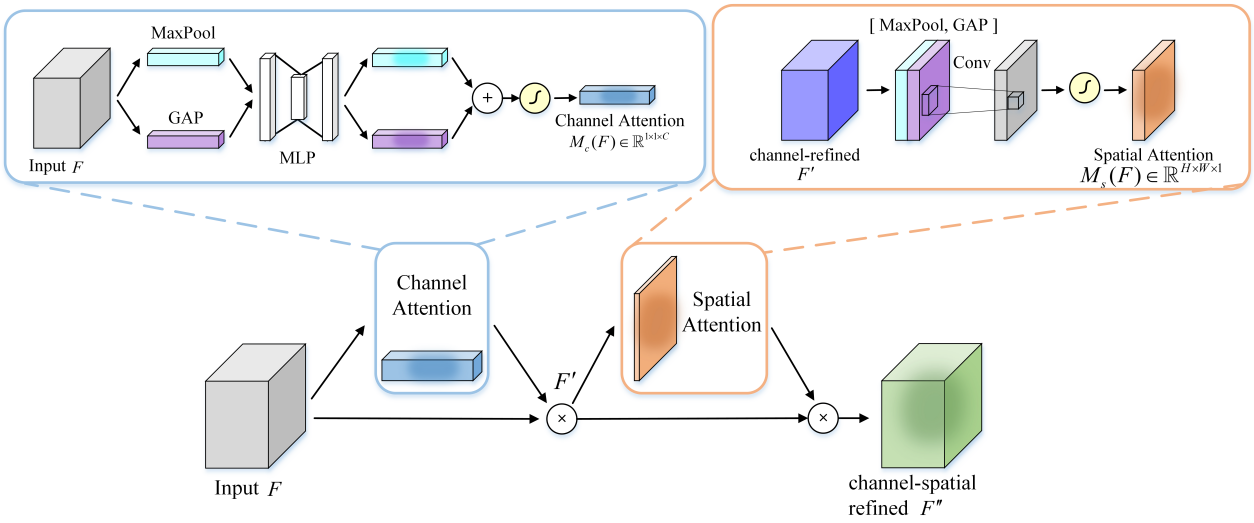


Figure 25. CBAM module architecture.

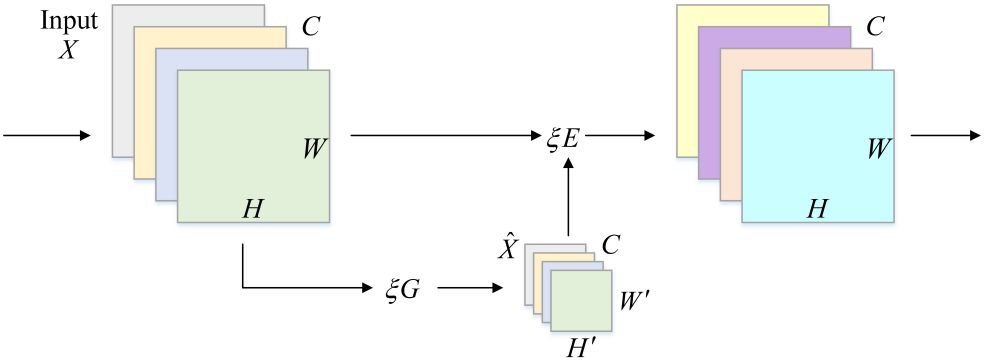


Figure 26. Gather-Excite block.

3.4.5. SKNet

In 2018, Wang et al. [141] proposed Selective Kernel Networks (SKNet), which can adaptively adjust the size of the receptive field according to multiple scales of input features. This network is mainly divided into three operations: *Split*, *Fuse* and *Select*, as shown in Figure 27.

***Split*:** For the given input feature maps $X \in \mathbb{R}^{H'\times W'\times C'}$, by default two transformations are performed $\widetilde{F} : X \to \widetilde{U} \in \mathbb{R}^{H\times W\times C}$ and $\hat{F} : X \to \hat{U} \in \mathbb{R}^{H\times W\times C}$ with 3×3 conv and 5×5 conv. $\widetilde{F}$ and $\hat{F}$ are composed of efficient grouped/depthwise convolutions, BN and ReLU function in sequence.

***Fuse*:** The two branches are fused via an element-wise summation: $U = \widetilde{U} + \hat{U}$. Afterwards, the channel-wise information is generated by using GAP: $F_{gp} : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{1\times1\times C}$, and the number of channels is changed by using two FC layers as $F_{fc} : 1 \times 1 \times C \to 1 \times 1 \times Z \to 1 \times 1 \times C, (Z < C)$. Finally, the output matrix is a and b.

***Select*:** a and b are weighted $\widetilde{U}$ and $\hat{U}$, and the final output is $V = a * \widetilde{U} + b * \hat{U}, (a + b = 1)$. $*$ indicates element-wise product.

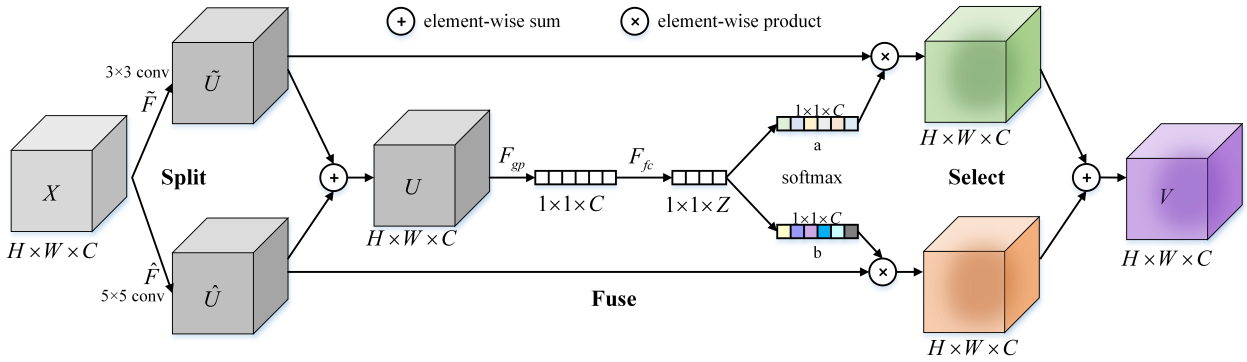


Figure 27. Selective Kernel convolution block.

3.4.6. GSoP-Net

In 2019, Gao et al. [142] proposed Global Second-order Pooling Convolutional Networks (GSoP-Net), which introduced two-dimensional GAP into the middle part of CNNs. It embodies the relationship between channels in the form of covariance. Refs. [143–147] also proved that inserting GSoP into the network can significantly improve performance, and A^2-Net [148] uses GSoP to aggregate and propagate the global features from the entire space of the input information. This method designs a simple and effective GSoP block, which has the advantages of high modularity, low memory usage, and low computational complexity. In addition, it can capture global second-order statistical information along the channel dimension or location dimension, and can also be easily inserted into the existing network architecture to further improve its performance with less overhead.

3.4.7. ECA-Net

In 2019, Hu et al. [149] proposed the Efficient Channel Attention (ECA) module to solve the irrationality of channel dimensionality reduction from SE block [27] (Figure 23 $F_e x$). Ref. [149] show avoiding dimensionality reduction is important for learning channel attention, and appropriate cross-channel interaction can preserve performance while significantly decreasing model complexity. ECA-Net proposes a local cross-channel interaction strategy without dimensionality reduction by using one-dimensional (1D) convolution, as shown in Figure 28. This method uses 1D convolution with K parameters which is adaptively determined via a mapping of channel dimension C to share all channels. Compared with more sophisticated attention modules, ECA blocks are more efficient to reduce the complexity of the model and significantly improve the performance of the model with slightly increasing parameters.

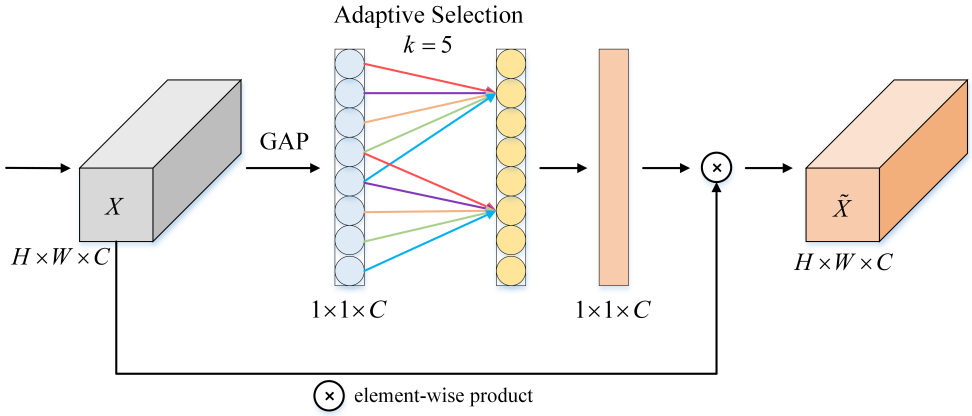


Figure 28. Efficient Channel Attention (ECA) module.

3.4.8. Coordinate Attention

In 2021, Hou et al. [150] proposed the Coordinate Attention (CA) module for efficient mobile networks, which uses two 1D GAP to capture feature codes in two spatial directions (positional information), CA block as shown in Figure 29. For the given input feature maps $X \in \mathbb{R}^{H \times W \times C}$. First, encoded each channel along the horizontal coordinate direction and the vertical coordinate direction by X GAP and Y GAP. Secondly, coordinate attention is generated by Concat + Conv2d, then after the BN + activation function, it is split into two feature maps with spatial direction features. Finally, the original input X is Re-weighted, r is the reduction ratio. The CA block captures not only cross-channel but also direction-aware and position-sensitive information, which helps models to more accurately locate and recognize the objects of interest. It is a lightweight module designed for mobile networks and can be flexibly inserted into classic mobile networks to improve performance with almost no computational overhead.

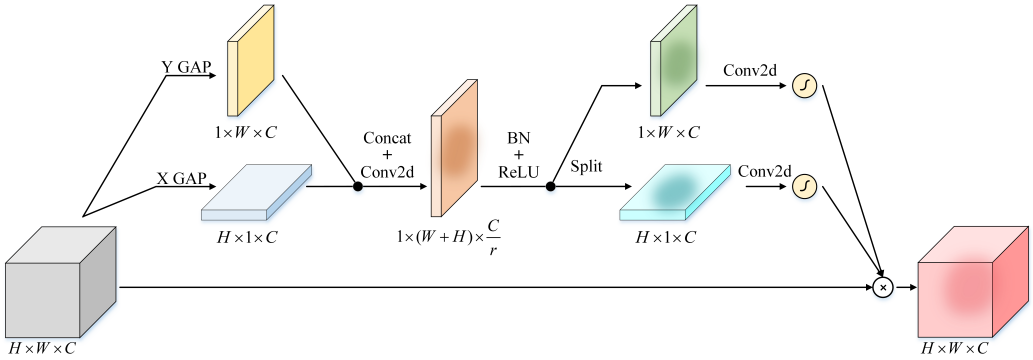


Figure 29. Coordinate Attention (CA) block.

3.4.9. Other Attention Modules and Summary

SENet [27] is one of the most successful examples of using the attention mechanism for image classification. Not only the above BAM [138], CBAM [139], GENet [140], SKNet [141], ECA-Net [149] and CA [150], but also GALA [151], AA-Net [152] and TA [153] developed its ideas by adopting different spatial attention mechanisms or designing advanced attention blocks. In addition, GSoP-Net [142], A^2Net [148], NLNet [154], GCNet [155], SCNet [156] and CCNet [157] serve as typical examples of non-local/self-attention networks are recently very popular due to their capability of building spatial or channel-wise attention. These attention networks are very helpful in a variety of computer vision tasks, by inserting into a large network or mobile network. Please note that NLNet [154] and CCNet [157] are not used for image classification.

3.5. Smaller or More Efficient Network

There is a trend to make deeper and more complicated networks to achieve higher accuracy. However, in many real-world applications such as mobile phones, robotics, and self-driving cars, these advances to improve accuracy are not necessarily making networks more efficient with respect to size and speed. Here we will introduce smaller and more efficient models that are dedicated to carrying out the recognition tasks in a timely fashion on a computationally limited platform.

3.5.1. SqueezeNet

In 2016, F. N. Iandola et al. [158] proposed a miniaturized network model structure SqueezeNet. This network introduced the fire module (Figure 30), which reduces the network parameters by replacing 3 × 3 filters with 1 × 1 filters and decreasing the number of input channels to 3 × 3 filters. It places down sampling late in the network so that convolution layers have large activation maps. Meanwhile, the model is combined with Deep Compression [159] to scale the volume of the model. Compared with AlexNet, SqueezeNet reduces the number of parameters by nearly 50 times while ensuring that there

is no loss of accuracy, and the model volume is compressed to about 510 times the original. After that, SqueezeNext [160] considered the hardware based on SqueezeNet.

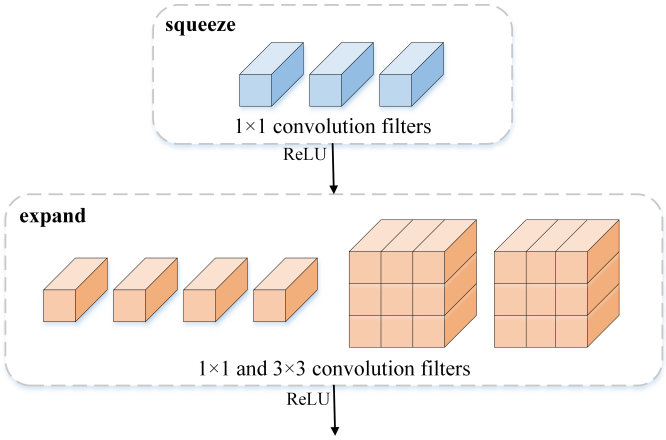


Figure 30. Fire protection module in SqueezeNet architecture.

3.5.2. MobileNet V1 to V3

In 2017, Google proposed MobileNetV1 [161], which is a lightweight network that focuses on mobile or embedded devices. This network uses depthwise separable convolution consisting of depthwise convolution and pointwise convolution (1×1 conv) to instead of standard convolution, as shown in Figure 31. It can greatly reduce calculation cost and parameters. At the same time, MobileNetV1 also provides two hyperparameters (width multiplier α and resolution multiplier α) to effectively balance calculation and accuracy.

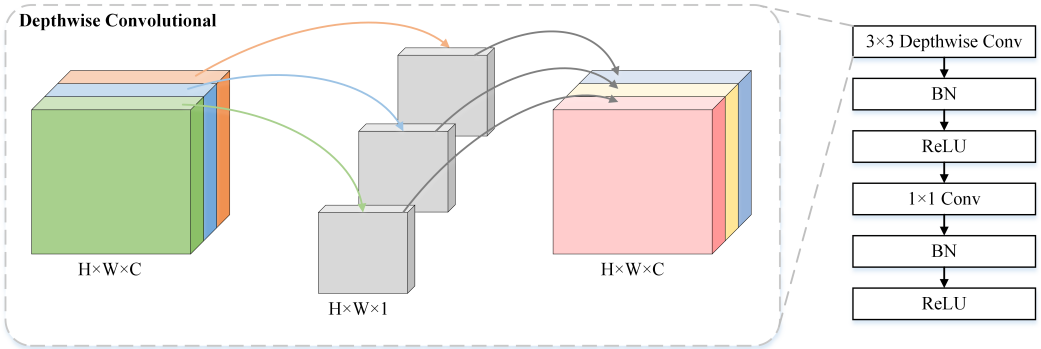


Figure 31. Depthwise Separable convolutions.

In 2018, MobileNetV2 [162] introduced Inverted Residuals and Linear Bottlenecks to solve the problem that most of the convolution kernel parameters in the depthwise separable convolution are zero. The reason is that the feature information is easily damaged by ReLU after being mapped from high-dimensional space to low-dimensional space [162]. The bottleneck residual block of MobileNetV2 is shown in Figure 32. Different from standard residual block [25] 1×1(compression)$\rightarrow 3 \times 3 \rightarrow 1 \times 1$(expansion), the inverted residual block is 1×1(expansion)$\rightarrow 3 \times 3 \rightarrow 1 \times 1$(compression). And this block replaces the last ReLU connected behind 1×1conv with linear transformation, to avoid information loss.

In 2019, based on the previous work, MobileNetV3 [163] employs the SE block [27] and Neural Architecture Search (NAS) [164,165] technology to achieve more efficiency and accuracy. SE block is used to build channel-wise attention. The platform-aware NAS for block-wise search approach [164] is used to find the global network structures, and then NetAdapt [165] for layer-wise search approach is used to fine-tune individual layers in a sequential manner. This model also uses the h-swish activation function that modifies the sigmoid of the swish function to improve accuracy.

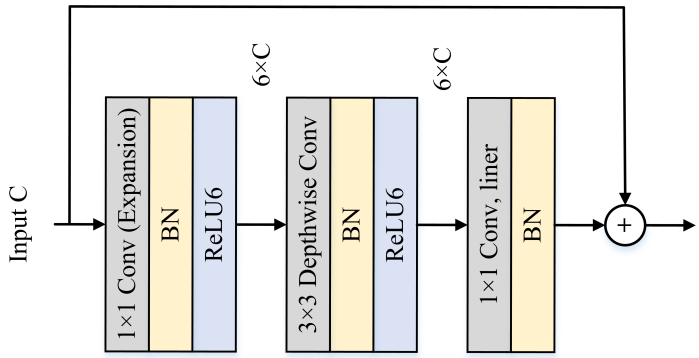


Figure 32. The bottleneck residual block of MobileNetV2. C is the number of channels, and expansion ratios are 6.

3.5.3. ShuffleNet V1 to V2

In 2017, ShuffleNetV1 [166], by Face++, focuses on common mobile platforms (e.g., drones, robots, and smartphones), using Pointwise Group convolution [19,26,126] and Channel shuffle to improve residual block [25]. The former is to solve the problem that the expensive pointwise convolutions result in a limited number of channels to meet the complexity constraint that the accuracy might be significantly damaged. The latter is to solve the problem that the group convolution blocks information flow between channel groups and weakens representation. ShuffleNet unit (stride = 1) replaces the first 1×1 conv with pointwise group convolution followed by a channel shuffle operation (see Figure 33c), as shown in Figure 33a. ShuffleNet unit (stride = 2) make two modifications that a 3×3 GAP is added to the shortcut path and the channel concatenation is replaced by element-wise addition, as shown in Figure 33b. The multiple versions of mobileNetV1 have experimented with a different number of groups for convolutions (g groups) and different scaling for the number of filters (s).

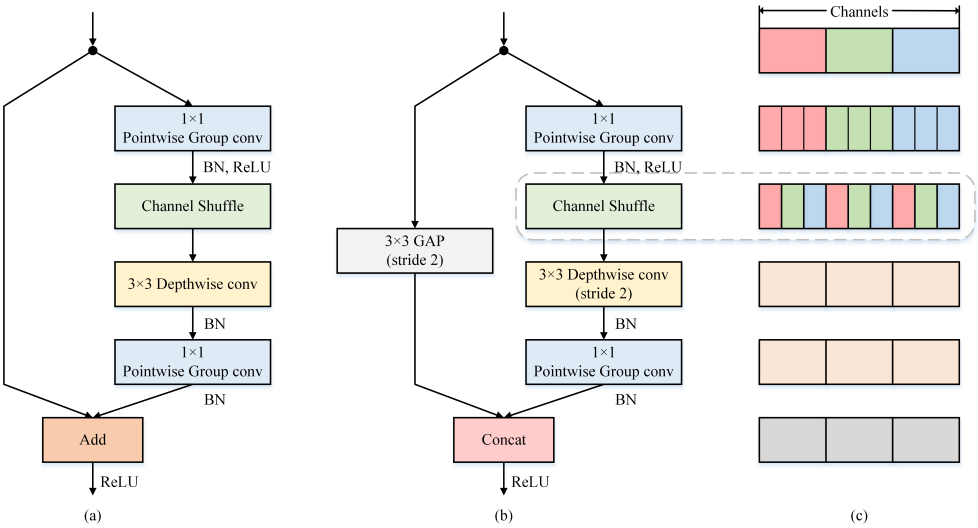


Figure 33. (**a**) ShuffleNetV1 unit with stride = 1. (**b**) ShuffleNetV1 unit with stride = 2 (down sampling, 2). (**c**) Channel shuffle operation.

In 2018, ShuffleNetV2 [167] has higher requirements for speed and accuracy. It not only considers computational complexity (floating-point operations per second, FLOPs) but also takes into account other factors such as memory access cost (MAC) and platform features. According to the experiment, four guidelines are given in [167]: (G1) Equal channel width minimizes MAC; (G2) Excessive group convolution increases MAC; (G3) Network fragmentation reduces the degree of parallelism, i.e., such fragmented structure could decrease efficiency because it is unfriendly for devices with strong parallel computing

powers like GPU. (G4) Element-wise operations are non-negligible because it also has a high MAC/FLOPs ratio. The ShuffleNetV2 unit avoids violating the above guidelines, as shown in Figure 34.

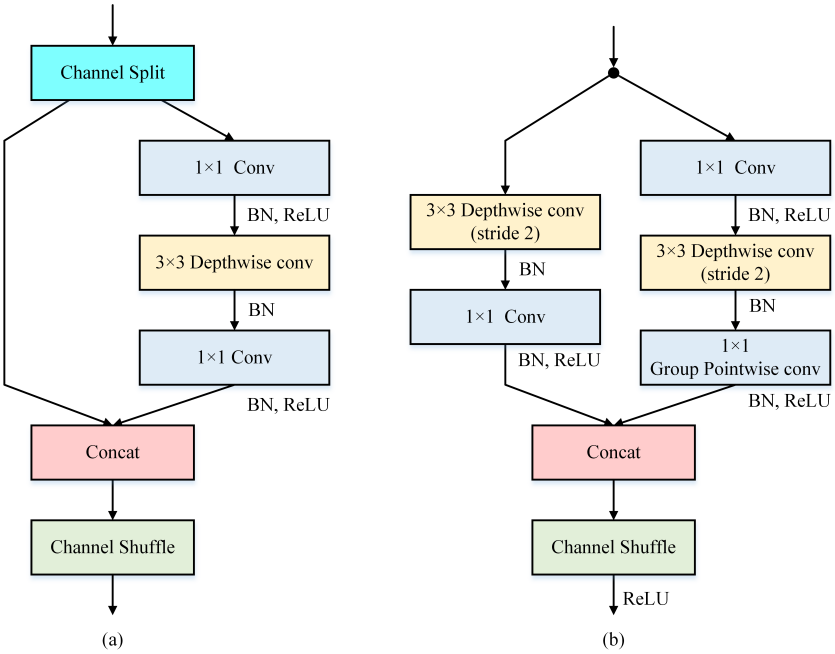


Figure 34. (**a**) ShuffleNetV2 unit. (**b**) ShuffleNetV1 unit with stride = 2 (down sampling, 2).

3.5.4. PeleeNet

In 2018, PeleeNet [168] makes improvements based on DenseNet [131], which is an efficient architecture for embedded platforms. It uses a steam block (see in Figure 35a) to downsample the input image for the first time. The original dense layer is divided into two paths to get different scales of receptive fields, called a 2-way dense layer (see in Figure 35b). It keeps the number of output channels the same as the number of input channels in transition layers because the compression factor proposed by DenseNet hurts the feature expression. Another improvement is that the number of channels in the bottleneck layer varies according to the input shape instead of fixed 4 times of growth rate, which can save up to 28.5% of the computational cost compared to DenseNet. Finally, it uses post-activation (Conv→BN→→ReLU) instead of pre-activation [109] to improve actual speed.

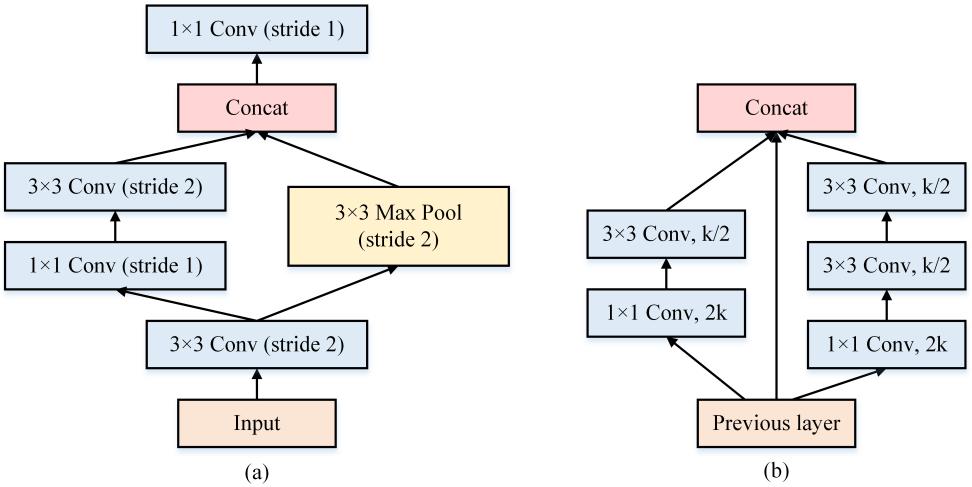


Figure 35. (**a**) Structure of stem block. (**b**) Structure of 2-way dense layer.

3.5.5. MnasNet

In 2018, MnasNet [164], by Tan et al., is an automated mobile neural architecture search network that is used to build mobile models using reinforcement learning (RL). It incorporates the basic essence of CNN and thereby strikes the right balance between enhancing accuracy and reducing latency, to depict high performance when the model is deployed onto a mobile. MnasNet explicitly incorporates the speed information into the main reward function of the search algorithm and directly measures model speed by executing the model on a particular platform. This architecture, in general, consists of two phases are as follows:

Factorized hierarchical search space: The search space supports diverse layer structures to be included throughout the network. The CNN model is factorized into various blocks wherein each block has a unique layer architecture. The connections are chosen such that both the input and output are compatible with each other, and henceforth yield good results to maintain a higher accuracy rate. Figure 36 shows a schematic diagram of the search space in MnasNet.

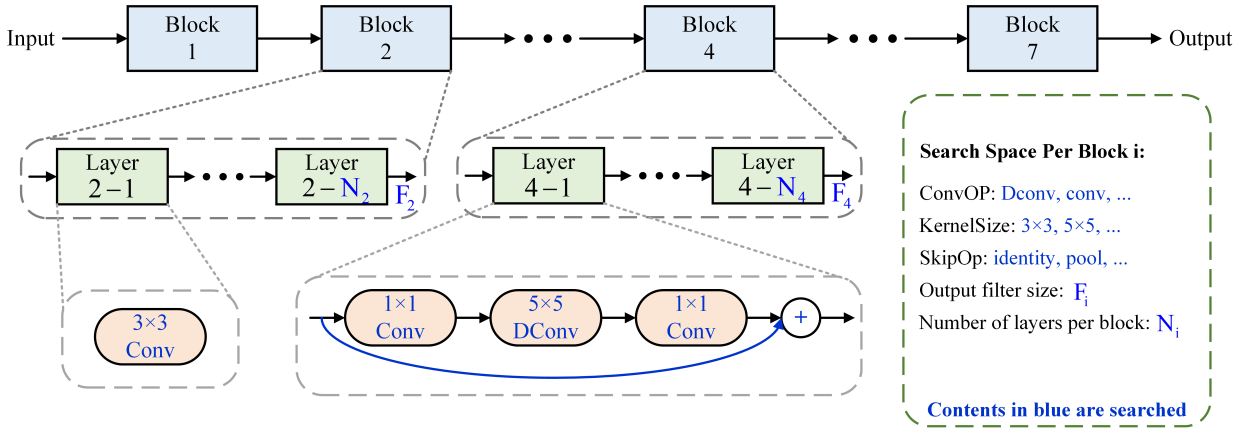


Figure 36. Factorized Hierarchical Search Space. Convolutional ops (ConvOP): regular conv (conv), depthwise conv (dconv), and mobile inverted bottleneck conv with various expansion ratios. Convolutional kernel size (KernelSize). Skip operations (SkipOp): max or average pooling, identity residual skip, or no skip path.

Reinforcement search algorithm: It employs a reinforcement learning approach where the rewards are maximized (multi-objective reward) to achieve two major objectives (latency and accuracy). Each CNN model as defined in the search space would be mapped to a sequence of actions that are to be performed by an RL agent. The controller is a Recurrent Neural Network (RNN), and the trainer trains the model and outputs the accuracy. The model is deployed onto a mobile phone to estimate the latency. Both accuracy and latency are consolidated into a multi-objective reward. This reward is sent to RNN using which the parameters of RNN are updated, to maximize the total reward. An overview of Platform-Aware NAS for Mobile is shown in Figure 37.

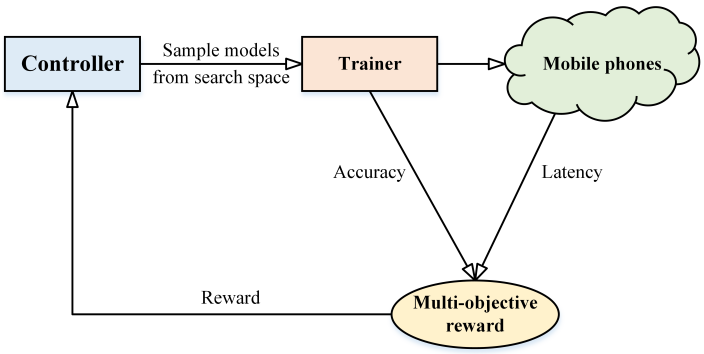


Figure 37. An Overview of Platform-Aware NAS for Mobile.

3.5.6. More Backbone Networks for Real-Time Vision Tasks

The above-mentioned smaller and more efficient networks are used to embed platforms. This attribute promotes more models as a backbone to be used in real-time vision tasks (e.g., object detection or instance segmentation). We have selected several influential network model series for a brief introduction, as follows:

1. **CSPNet.** Wu et al. [169] proposed Cross Stage Partial Network (CSPNet) to solve the problem of heavy inference computations, which is caused by the duplicate gradient information within network optimization. A large amount of gradient information in DenseNet [131] is reused for updating weights of different dense layers, which will result in different dense layers repeatedly learning copied gradient information. The network modifies the equations of the feed-forward pass and weight updating that the gradients coming from the dense layers are separately integrated and the feature map that did not go through the dense layers is also separately integrated. It preserves the advantages of DenseNet's feature reuse characteristics but at the same time prevents an excessive amount of duplicate gradient information by truncating the gradient flow. It also designed partial transition layers is to maximize the difference of gradient combination. CSPNet can be easily applied to DenseNet [131], ResNet [25] and ResNeXt [26].
2. **VarGNet.** In 2019, Variable Group Network (VarGNet) [170], by Zhang et al. makes a compromise between the lightweight models and the optimized hardware-side configurations methods. This embedded-system-friendly network is well suited to the targeted computation patterns and the ideal data layout because the computation patterns of a chip in an embedded system is strictly limited. The question of the State-Of-The-Art (SOTA) network is so complex that some layers can be accelerated by hardware design while others cannot. VarGNet sets the channel numbers in a group in a network to be constant, to balance the computation intensity. Later, VarGFaceNet [171] introduced Variable Group Convolution into the task of face recognition.
3. **VoVNet/OSANet.** In 2019, Lee et al. [172] proposed VoVNetV1 that the One-shot aggregation (OSA) module is designed which is more efficient than Dense Block [131]. Reducing FLOPs and model sizes does not always guarantee the reduction of GPU inference time and real energy consumption. Dense connections induce high MAC which is paid by energy and time, and the use of bottleneck structure Figure 21 which harms the efficiency of GPU parallel computation. The redundant information is also generated. VoVNet proposed an OSA module to aggregate its feature in the last layer at once that the MAC is much smaller than dense blocks and the GPU is more computationally efficient. It is also named as OSANet and further discussed in Scaled-YOLOv4 [173]. In 2020, the residual connection [25] and SE modules [27] are used in VoVNetV2 [174].
4. **Lite-HRNet.** The HRNet proposed by Wang et al. [175] maintains high-resolution representations by connecting high-to-low resolution convolutions in parallel and strengthens high-resolution representations by repeatedly performing multi-scale fusions across parallel convolutions, which is a model with a powerful performance in multiple visual tasks. Later, in 2021, Lite-HRNet [176] applies efficient shuffle blocks [166,167] to HRNet [175]. It introduces a lightweight unit, conditional channel weighting, to replace costly 1×1 pointwise convolutions in shuffle blocks.

3.5.7. EfficientNet V1 to V2

The conventional practice for model scaling is to arbitrarily increase the CNN depth [25] or width [119], or to use larger input image resolution for training and evaluation [177]. Refs. [119,178,179] shows that there is a certain relationship between network depth and width. While these methods do improve accuracy, they usually require tedious manual tuning, and still often yield suboptimal performance. In 2019, Tan et al. [180] proposed EfficientNetV1 that uses a simple yet highly effective compound coefficient ϕ to scale up

CNNs in a more structured manner. Unlike the above approaches that arbitrarily scale network dimensions (such as width w, depth d, and resolution r). This model scaling method uses ϕ to uniformly scales network w, d, and r, following [180]:

$$\begin{aligned} depth : d &= \alpha^{\phi} \\ width : w &= \beta^{\phi} \\ resolution : r &= \gamma^{\phi} \\ s.t \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\ \alpha \geq 1, \beta &\geq 1, \gamma \geq 1 \end{aligned} \tag{6}$$

where α, β, γ are constants that can be determined by a small grid search. ϕ is a user-specified coefficient that controls how many more resources are available for model scaling. Finally, it develops a baseline network by leveraging a multi-objective NAS [164]. The main building block is mobile inverted bottleneck MBConv [162,164] (see in Figures 32 and 36) and the SE modules [27] are also be added.

In 2021, EfficientNetV2 [181] proposed a smaller model and a faster training method. First, EfficientNetV2 applies FixRes [182] to solve the problem that the EfficientNetV1's [180] large image size results in significant memory usage [183], by using a smaller image size for training, but no fine-tuning of any layers after training. Ref. [125] also pointed out that using a smaller image size for training the accuracy will be slightly better. Secondly, for the depthwise convolution [127] often cannot fully utilize modern accelerators. EfficientNetV2 [181] gradually replaced MBConv [162,180] with Fused-MBConv [184] to better utilize mobile or server accelerators, that is replaced the depthwise 3×3 conv and expansion 1×1 conv in MBConv with a single 3×3 conv. Then used NAS to automatically search for the best combination of MBConv and Fused-MBConv. Finally, this network proposes a training-aware NAS to search for the best combinations. Another important point is that EfficientNetV2 uses modified progressive learning, training with different image sizes also changes the regularization intensity accordingly, to solve the problem of dropped in accuracy caused by dynamically changing the image sizes during training [185].

3.5.8. Other Technical Support

For mobile terminals and embedded devices, hand-designed CNN models are no longer the trend of the times. At present, more lightweight models are combined with other powerful algorithms. It is not difficult to see that some of the above introductions also includes impure CNN architecture design. Below we will briefly introduce several popular methods that can be combined with pure CNN:

1. **On the trained model:** Singular Value Decomposition (SVD) [186] can achieve the effect of model compression by compressing the weight matrix of the fully connected layer in the network, Low-rank filter [187] uses two $1 \times K$ conv instead of one $K \times K$ conv to remove redundancy and reduce weight parameters; The network pruning [188–192] method is to discard the connections with lower weights in the network, to reduce the network complexity; Quantization [193–199] reduces the space required for each weight by sacrificing the accuracy of the algorithm; Binarization of neural networks [195,200] can be regarded as a kind of extreme quantification, which uses a binary representation of the network weights and greatly reduces the model size; Deep Compression [159] uses three steps of Pruning, Quantization and Huffman Coding to compress the original model, and achieves an amazing compression rate without loss of accuracy. This method is of landmark significance, leading a new frenzy of miniaturization and accelerated research of CNN models.
2. **NAS Search:** A lightweight network usually needs to be smaller and faster with as high an accuracy as possible. There are too many factors to consider, which is a huge challenge to design an efficient model. To automate the architecture design process, RL was first introduced to search for efficient architectures with competitive accuracy [201–205]. A fully configurable search space can grow exponentially large and intractable. So early works of architecture search focus on the cell level structure

search, and the same cell is reused in all layers. Ref. [164] explored a block-level hierarchical search space allowing different layer structures at different resolution blocks of a network. To reduce the computational cost of search, a differentiable architecture search framework is used in [206–212] with gradient-based optimization. Focusing on adapting existing networks to constrained mobile platforms, refs. [164,165,213,214] proposed more efficient automated network simplification algorithms.

3. **Knowledge Distillation (KD):** KD [215,216] refers to the idea of model compression by teaching a smaller network, step by step, exactly what to do using a bigger already trained network. This training setting is sometimes referred to as "teacher-student", where the big one is the teacher, and the small model is the student. In the end, the student network can achieve a similar performance to the teacher network.

3.6. Competitive Methods and Training Strategy

There are much other works that outperforms pure CNN methods on image classification, each with advantages and potentials. And a strong training strategy will bring amazing improvements to the CNN model. We will discuss these representative models and strategies in this section. Image Classification on ImageNet (https://paperswithcode.com/sota/image-classification-on-imagenet (accessed on 1 July 2021)) provides detailed data for learning and reference.

3.6.1. Vision Transformer

Transformers [217] based on self-attention mechanism [218] have achieved great success in natural language processing (NLP). Thanks to its strong representation capabilities, researchers are looking at ways to apply transformer to computer vision tasks. The pure transformer architectures represented by ViT [219] performs well on the image classification task. Recently, more variant ViT models, e.g., DeiT [220], PVT [221], TNT [222], and Swin [223], have been proposed for the pursuit of stronger performance. There are also plenty of works trying to augment a pure transformer block or self-attention layer with a convolution operation, e.g., BoTNet [224], CeiT [225], CoAtNet [226], CvT [227]. Some works (such as the DETR methods [228–230]) try combining CNN-like architectures with transformers for object detection. Of course, Transformers have made achievements in more visual tasks, high/mid/low-level vision and video processing, however, we will not introduce too much here. Ref. [229] gives a more comprehensive summary if you want a more detailed understanding. Nowadays, there are still huge challenges in applying transformer models or mixed methods to computer vision tasks. More research in this direction is being explored on image classification tasks.

3.6.2. Self-Training

Self-training [231–234] has been applied successfully to improve SOTA models in image classification [235–237]. This method has three main steps: (i) Train a teacher model on labeled images. (ii) Use the teacher to generate pseudo labels on unlabeled images. (iii) Combine labeled images and pseudo labeled images to train the student model. SSL [237] proposes a semi-supervised deep learning method based on teachers-students to improve the performance of the large CNNs (like ResNet-50 [25] or ResNeXt [26]). To make students have stronger learning ability. Noisy Student [236] makes the student larger than or at least equal to the teacher, and it adds noise to the student such as RandAugment data augmentation [238], dropout [80], and stochastic depth [117]. The problem of confirmation bias in pseudo-labeling [239] can make students learn from inaccurate data if the pseudo labels are inaccurate. Meta Pseudo Labels [235] is constantly adapted through the feedback of the student's performance on the labeled dataset so that the student can learn better pseudo labels from the teacher.

3.6.3. Transfer Learning

DCNNs usually require a large amount of task-specific data and compute to obtain good performance. Applying these SOAT networks to new tasks can be prohibitively expensive. Transfer learning provides a solution that a network completes training on a large general data set and its weights to are then used to initialize subsequent tasks [124,240–242]. Many Pre-trained models, e.g., GoogLeNet [23] and ResNet [25], have been trained on large datasets such as ImageNet for image classification purposes. This method has been used as a regular solution for new tasks in image classification. In particular, BiT [124] provides a general recipe to achieve excellent performance in many tasks.

3.6.4. Data Augmentation

CNNs often face the risk of overfitting caused by limited data. Traditional data augmentation techniques [19,243–246] include a collection of methods, e.g., flipping, color space, cropping, rotation, translation, and noise injection, that can improve the attributes and size of the training data set. Not only that, but it also has the potential to significantly improve the generalization of DL models. Automated data enhancement [238,247–250] has the potential to address some weaknesses of traditional data augmentation methods, by training a CNN model with a learned data augmentation policy may significantly improve accuracy, model robustness, and performance on semi-supervised learning for image classification. Mixed Sample Data Augmentation (MSDA) [245,246,251] is to randomly mix two training samples and their labels according to a certain ratio, which can not only reduce the misidentification of some difficult samples but also improve the robustness of the model and make it more stable during training.

3.6.5. Other Training Strategies

Achieving better performance is not just about designing an excellent architecture, training-based optimization methods are also crucial. In addition to the methods described above, there are several types of reliable options:

1. **Optimizer.** The optimizer effectively minimizes the loss function to achieve ever better performance, such as SGD [113], Adam [252], PMSProp [253]. Sharpness-Aware Minimization (SAM) [254], as the best solution at present, alleviates the relationship between minimizing loss function and model generalization ability.
2. **Normalization.** BN [106] is a key component of most image classification models, which can achieve higher accuracies on both the training set and the test set. More variants also extend this idea such as layer normalization [255] and group normalization [256]. But the recent research shows that some important flaws of BN will affect the long-term development of CNN [125,257–260]. NFNet [125] trains deep model without normalization, by using core technology called Adaptive Gradient Clipping (AGC).
3. **Train-test resolution.** Data augmentation is the key to training CNNs. However, the processed images of different resolutions will have a great impact on the model. Refs. [182,261] proposes a training strategy that employs different train and test resolutions, to optimize the classifier performance.

4. Comparison of Various Image Classification Methods

4.1. Common Data Sets for Image Classification

The following are several commonly used classification data sets, with increasing difficulty in classification.

1. MNIST [262]: The image resolution of this dataset is a 28 × 28 grayscale image. Each picture has 784-pixel grayscales with an integer value of [0, 255]. It contains a training set of 60,000 examples and a test set of 10,000 examples. And it is composed of handwritten numbers (0–9) from 250 different people, see Figure 38a.
2. CIFAR-10 [263]: The image resolution of this dataset is 32 × 32 RGB images, including 60,000 images, which are divided into 10 categories and independent of each other.

Each category contains 6000 images, including 5000 training images and 1000 test images, see Figure 38b.

3. CIFAR-100 [263]: The dataset image resolution is 32 × 32 RGB images, including 60,000 images, divided into 100 categories and independent of each other. Each category includes 500 training images and 100 test images. Compared with the data set CIFAR-10, this dataset divides 100 classes into 20 super classes, see Figure 38c.
4. ImageNet [101]: The dataset has approximately 1.5 million annotated images, at least 1 million images provide border annotations, and contain more than 20,000 categories, and each category has no less than 500 images. Beginning in 2010, the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) held every year will end after 2017. Competition items include image classification, target positioning, target detection, video target detection, scene classification, and scene analysis. The data used in ILSVRC is only a part of the ImageNet dataset, see Figure 38d.

Figure 38. Visual image classification datasest (small part). ImageNet's original image does not have a fixed size.

4.2. Comparison and Results

The comparison results of various image classification methods are shown in Tables 3 and 4. The indicators and accuracy considered by the lightweight model are different from those of the large model, so here is a separate list. In Table 1, the main comparative factors include model name, year, depth/version, data set, FLOPs (Billion), model parameters, the accuracy of ImageNet (Top-1) and CIFAR 10/100, main characteristics, and approach (such as quotation of ideas, training strategy, non-pure CNN methods). In Table 2, the main comparative factors of the lightweight model include model name, year, depth/baseline, data set, FLOPs (Million), model parameters, accuracy of ImageNet (Top-1), CPU latency, main characteristics, and approach. The FLOPs in the table are only for the ImageNet dataset. * indicates that the model uses extra training data.

Table 3. Comparison of Various Image Classification Methods.

Model	Year	Depth/Version	Data Set	BFLOPs	Params	Accuracy(%) ImageNet, Top-1	Accuracy(%) CIFAR 10	Accuracy(%) CIFAR 100	Characteristics	Approach
AlexNet	2012	8	ImageNet	-	60M	63.3	-	-	ReLU and Dropout	CNN
NIN	2013	3	CIFAR 10/100	-		-	91.19	64.33	"mlpconv" layer and GAP	CNN
VGGNet	2014	16	ImageNet	-	138M	74.4	-	-	Small filter size, Blocks of layers	CNN
InceptionV1 (GoogLeNet)	2014	22	ImageNet	1.45	6.8M	-	-	-	Inception block with different filter size	CNN
InceptionV2	2015	-	ImageNet	1.94	11.2M	74.8	-	-	BN, Small filter size	CNN
InceptionV3	2015	48	ImageNet	5.73	24M	78.8	-	-	Using small filter size to burn less computational power	CNN
ResNet	2015	50, 101 110, 1202	ImageNet CIFAR 10/100	3.86, 7.57	25.6M, 44.5M 1.7M, 10.2M	77.15, 78.25	93.57, 92.07	74.84, 72.18	Residual learning	CNN
ResNet (pre-activation)	2016	200 164, 1001	ImageNet CIFAR 10/100	-	64.7M 1.7M, 10.2M	79.9	93.63, 95.38	-, 77.29	pre-activation structure	CNN
Stochastic Depth	2016	110, 1202	CIFAR 10/100	-	1.7M, 10.2M	-	94.77, 95.09	75.09, -	Stochastic delete some layers	CNN
WRN	2016	50 16, 28	ImageNet CIFAR 10/100	-	68.9M 11.0M, 36.5M	78.1	95.74, 96.00	79.57, 80.75	Wider and shallower	CNN
DRN-A	2016	50	ImageNet	-	25.6M	78.7	-	-	Using Dilated convolutions to increase the receptive field	CNN
DenseNet	2016	264 190	ImageNet CIFAR 10/100	6	- 25.6M	77.85	96.54	82.82	Each layer connects to every other layer	CNN
Inception-ResNet-v2	2016	164	ImageNet	11.75	55.8M	80.1	-	-	Combined residual connection and Inception	CNN
Xception	2016	71	ImageNet	8.4	22.8M	79	-	-	Combined residual connection and Inception Depthwise separable convolutions	CNN
InceptionV4	2016	70	ImageNet	13	48M	80	-	-	Divided transform and integration concepts	CNN
ResNeXt	2016	101(32×4d) 101(64×4d)	ImageNet	7.508 32	44.18M 83.6M	78.80 80.90	-	-	Combined residual connection and Inception Grouped convolution	CNN
DropBlock	2018	50	ImageNet	3.86	25.6M	78.35	-	-	Dropout the units with a contiguous region	CNN
Attention	2017	92 92, 452	ImageNet CIFAR 10/100	10.4	51.3M 1.9M, 8.6M	80.50 -	- 95.01, 96.10	- 78.29, 79.55	Stacked attention modules based on residual connections	CNN +Attention′
PolyNet	2017	-	ImageNet	34.7	92M	81.3	-	-	Combined residual connection and Inception	CNN
SENet	2017	101 152	ImageNet	8.00 42	49.2M 146M	81.36 82.72	-	-	Channel attention Implantable lightweight module	CNN +Attention′
DPN	2017	131	ImageNet	16	-	81.38	-	-	Combination of residual learning and dense connection	CNN
NASNet-A	2017	-	ImageNet	23.8	88.9M	82.7	-	-	NASNet search space	CNN +NAS
PNASNet	2017	-	ImageNet	25	86.1M	82.9	-	-	Smaller NASNet search space Sequential Model-based Optimization	CNN +NAS

Table 3. *Cont.*

Model	Year	Depth/Version	Data Set	BFLOPs	Params	Accuracy(%) ImageNet, Top-1	Accuracy(%) CIFAR 10	Accuracy(%) CIFAR 100	Characteristics	Approach
ResNeXt (32×4d) + BAM	2018	101	ImageNet	8.05	44.6M	80.85	-	-	Channel attention and spatial attention Implantable lightweight module	CNN +Attention′
ResNeXt (32×4d) + CBAM	2018	101	ImageNet	7.519	49.2M	80.58	-	-	Channel attention and spatial attention Implantable lightweight module	CNN +Attention′
AmoebaNet-A	2018	-	ImageNet	23.1 104	86.7M 469M	82.8 83.9	-	-	NASNet search space and genetic algorithm	CNN +NAS
GPipe	2018	-	ImageNet CIFAR 10/100	-	557M	84.3	99	91.3	Using Pipeline Parallelism to train effectively	CNN
ResNet + GE	2019	101	ImageNet CIFAR 10/100	7.59	58.4M -	79.26 -	- 95.07	- 79.15	Feature context exploitation Implantable lightweight module	CNN +Attention′
ECA-Net	2019	101	ImageNet	7.35	42.45M	78.65	-	-	No channel dimensionality reduction Portable lightweight modul	CNN +Attention′
SKNet	2019	101 29	ImageNet CIFAR 10/100	8.46	48.9M 27.7M	79.81 -	- 96.53	- 82.67	Adaptively adjust receptive field Implantable lightweight module	CNN +Attention′
GSoP-Net	2019	50	ImageNet	6.56	58.65M	78.81	-	-	Global Second-order Pooling Implantable lightweight module	CNN +Attention′
EfficientNetV1	2019	B0 - B7, L2	ImageNet	0.39, 0.70, 1.0, 1.8, 4.2, 9.9, 19, 37, -	5.3M, 7.8M, 9.2M 12M, 19M, 30M 43M, 66M, 480M	77.1, 79.1, 80.1, 81.6, 82.9, 83.6, 84.0, 84.3, 85.5	98.9 (B7)	91.7 (B7)	Utilizes effective compound coefficient to scale up CNNs	CNN +NAS
BiT	2019	-L	ImageNet CIFAR 10/100	-	-	87.54*	99.37	93.51	Transfer learning Better training and fine-tuning strategies	CNN Transfer-learning
NoisyStudent	2019	-L2	ImageNet	-	480M	88.4*	-	-	Self traning / Semi-Supervised add noise to the student	CNN Semi-Supervised
FixEfficientNet	2020	-B7 -L2	ImageNet	-	66.4M 480M	85.3, 87.1* 85.7, 88.5*	-	-	A training strategy that employs different train and test resolutions	CNN
ViT-H/14	2020	-L, -H	ImageNet CIFAR 10/100	-	307M, 632M	87.76*, 88.55*	99.42, 99.50	90.54, 90.72	Pure transformer model for Computer vision	Vision Transformer
Meta Pseudo Labels	2020	-L2	ImageNet	-	480M	90.2*	-	-	Self traning / Semi-Supervised Update teacher based on student performance	CNN Semi-Supervised
EfficientNetV2	2021	-S, -M, -L	ImageNet CIFAR 10/100	8.8, 24, 53	22M, 54M, 120M 24M, 55M, 121M	83.9, 85.1, 85.7	98.7, 99.0, 99.1	91.5, 92.2, 92.3	FixRes, Fused-MBConv, NAS	CNN +NAS
NFNet	2021	F6+SAM, F4+	ImageNet	337.28, -	438.4M, 527M	86.5, 89.2*	-	-	Adaptive gradient clipping technique, Without Normalization	CNN
ViT-G/14	2021	-G/14	ImageNet	-	1843M	90.45*	-	-	Pure transformer model for Computer vision	Vision Transformer
CoAtNet	2021	-7	ImageNet	2586	2440M	90.88*	-	-	Mixed methods, Convolutional + Transformer	CNN + Transformer

Table 4. Comparison of Various Image Classification Lightweight Methods.

Model	Year	Version/Baseline	Data Set	MFLOPs	Params	Accuracy(%) Top-1	CPU Latency	Characteristics	Approach
SqueezeNet	2016	-	ImageNet	1700	1.25M	57.5	-	Decreased 3×3 conv, deep compression	CNN
MobileNetV1	2017	1.0	ImageNet	569	4.2M	70.6	113ms	Depthwise separable convolution	CNN
ShuffleNetV1	2017	2×(g=3), +SE	ImageNet	527	-	75.3	-	Group pointwise convolution, channel shuffle	CNN
PeleeNet	2018	-	ImageNet	508M	2.8M	72.6	-	Improved DenseNet	CNN
MobileNetV2	2018	1.0 1.4	ImageNet	300 585	3.4M 6.9M	72.0 74.7	75ms 143ms	Inverted residual block	CNN
ShuffleNetV2	2018	2×(g=3), +SE	ImageNet	597	-	75.4	-	More consideration for MAC	CNN
MnasNet	2018	A3	ImageNet	391	5.2M	76.7	103ms	Factorized hierarchical search space Reinforcement search algorithm	CNN +NAS
ECA-Net	2019	MobileNetV2-1.0	ImageNet	319	3.34M	72.5	-	Channel Attention Implantable lightweight module	CNN +Attention
MobileNetV3	2019	1.0 Large	ImageNet	219	5.4M	75.2	61ms	SE block, NAS, h-swish activation function	CNN +NAS
MobileNeXt	2020	1.0 1.4	ImageNet	300 590	3.4M 6.1M	74.0 76.1	211ms -	Improved inverted residual block Proposed sandglass block	CNN
CA	2021	MobileNetV2-1.0 MobileNeXt-1.0	ImageNet	310 330	3.95M 4.09M	74.3 75.2	-	Positional information + channel attention Implantable lightweight module	CNN +Attention

In many cases, the reasonable setting of hyperparameters and regularization is also crucial in the successful training of the model. Table 5 provides several representative model settings during the training process for reference.

Table 5. Model settings during training. '32×(50)' indicates that the distributed machine learning system using 50 replicas running each on a GPU with batch size 32.

Model	Initial LR	Batch Size	Epochs	Optimizer	Regularization
VGG	0.01	256	74	SGD + Momentum 0.9	weight decay 0.5×10^{-3} Scale augmentation Dropout 0.5
ResNet	0.1	256	85	SGD + Momentum 0.9	weight decay 0.1×10^{-2} Scale augmentation BN
InceptionV3	0.045	32×(50)	100	RMSProp with decay 0.9	Label Smoothing Gradient clipping BN
DenseNet	0.1	256	90	SGD + Nesterov momentum 0.9	weight decay 0.1×10^{-3} Augmentation BN
ResNeXt	0.1	256	100	SGD + Momentum 0.9	weight decay 0.1×10^{-3} Scale augmentation BN
SENet	0.6	1024	100	SGD + Momentum 0.9	Label Smoothing Random horizontal flipping BN
SENet for mobile network	0.1	256	400	SGD + Momentum 0.9	Label Smoothing Random horizontal flipping BN
MobileNetV3	0.1	4096	120	RMSProp with decay 0.99; momentum 0.9; batch norm momentum 0.99	weight decay 0.1×10^{-4} Exponential moving average Dropout BN
EfficientNetV2	0.256	4096	350	RMSProp with decay 0.9; momentum 0.9; batch norm momentum 0.99	weight decay 0.1×10^{-4} Exponential moving average RandAugment and Mixup Dropout andstochastic depth BN

5. Conclusions

In this survey, we are not limited to a systematic summary of mainstream CNN models (such as architecture and characteristics), but also related introductions to non-pure CNN methods, mixed models, and training strategies. These methods are shining points in the development of image classification field. Next, we are committed to enabling readers to find inspiration in their work or research, it is mandatory for the inclusion of three brief discussions:

Summarizing our review

- The classic models from 2012 to 2017 provided the basis for the structural design of the CNN-based image classification method, so that many later studies have been established on their basis.
- The attention mechanism is introduced into CNN to form an embedded module, which can be easily and quickly inserted into any network to improve performance. For example, many models currently have SE blocks implanted.
- The networks designed for mobile platforms have smaller and more efficient network structures, which are generally in the extreme use of characteristics. It is the best choice to consider their characteristics comprehensively on a resource-constrained platform.
- The choice of hyperparameters has a great impact on the performance of CNN. Many works will reduce the amount of hyperparameters and replace them with other composite coefficients.
- Manually designing a network to achieve higher performance often requires more effort. NAS search can make this job much easier. It is a good choice to use NAS as the main tool or auxiliary tool to design the network.

- The CNN model relies on sizeable datasets to predict unlabeled data. Transfer learning and data augmentation can not only alleviate it effectively but also can increase the performance of the model.
- Not only designing efficient networks can improve performance, but training strategies can also help CNN models gain huge benefits.

The challenges of the CNN model

- Lightweight models often need to sacrifice accuracy to compensate for efficiency. Currently, the efficiency of using CNN is still being explored in embedded and limited systems.
- Although some models have achieved great success in semi-supervised learning, most CNN models have not transitioned to semi-supervised or unsupervised learning to manage data. In this regard, the NLP field is doing better.

The future directions

- Vision Transformer's achievements in image classification tasks cannot be underestimated. How to effectively combine convolution and Transformer has become one of the current hot spots. They have their own advantages and can complement each other such as the current SOTA network CoAtNet. This type of mixed model also needs further exploration.
- There are some stereotypical components in CNN may become factors hindering development, such as activation functions, dropout, or batch normalization. Various studies have achieved amazing results after breaking the convention, and such ideas are also worthy of further study.

In addition, Table 6 provides a list of acronyms frequently used in this paper for easy readership.

Table 6. Acronyms in the Paper.

Convolutional Neural Networks	CNNs	Multilayer Perceptron	MLP
Artificial Neural Networks	ANNs	Convolutional	Conv
Deep Convolutional Neural Networks	DCNNs	Fully Connected	FC
Reinforcement Learning	RL	Global Average Pooling	GAP
Natural Language Processing	NLP	Batch Normalization	BN
State-Of-The-Art	SOTA	Neural Architecture Search	NAS

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

Funding: This research was partially supported by National Key R&D Program of China: No. 2020YFB1713300, No. 2018AAA0101803; Higher Education Project of Guizhou Province: No. [2020]005, No. [2020]009; Science and Technology Project of Guizhou Province: No. [2019]3003.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 12 December 2016; pp. 2818–2826. [CrossRef]
2. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv* **2013**, arXiv:1311.2524.
3. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*, 640–651.
4. Toshev, A.; Szegedy, C. DeepPose: Human Pose Estimation via Deep Neural Networks. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 25 September 2014; pp. 1653–1660. [CrossRef]
5. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-Scale Video Classification with Convolutional Neural Networks. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 25 September 2014; pp. 1725–1732. [CrossRef]
6. Wang, N.; Yeung, D.Y. Learning a Deep Compact Image Representation for Visual Tracking. In Proceedings of the 26th International Conference on Neural Information Processing Systems—Volume 1, Lake Tahoe, NV, USA, 5–10 December 2013; NIPS'13. Curran Associates Inc.: Red Hook, NY, USA, 2013; pp. 809–817.
7. Dong, C.; Loy, C.C.; He, K.; Tang, X. Learning a Deep Convolutional Network for Image Super-Resolution. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 184–199.
8. Bhattacharyya, S. A Brief Survey of Color Image Preprocessing and Segmentation Techniques. *J. Pattern Recognit. Res.* **2011**, *1*, 120–129. [CrossRef]
9. Vega-Rodríguez, M.A. Review: Feature Extraction and Image Processing. *Comput. J.* **2004**, *47*, 271–272. [CrossRef]
10. D, Z.; Liu, B.; Sun, C.; Wang, X. Learning the Classifier Combination for Image Classification. *J. Comput.* **2011**, *6*, 1756–1763.
11. Mcculloch, W.S.; Pitts, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. *J. Symb. Log.* **1943**, *9*, 49–50. [CrossRef]
12. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. [CrossRef]
13. Duffy, K.R.; Hubel, D.H. Receptive field properties of neurons in the primary visual cortex under photopic and scotopic lighting conditions. *Vis. Res.* **2007**, *47*, 2569–2574. [CrossRef]
14. Werbos, P.J. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Ph.D. Thesis, Harvard University, Cambridge, MA, USA, 1974.
15. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
16. Zhou, J.; Zhao, Y. Application of convolution neural network in image classification and object detection. *Comput. Eng. Appl.* **2017**, *53*, 34–41.
17. Hinton, G.E.; Osindero, S.; Teh, Y.W. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef]
18. Cireşan, D.C.; Meier, U.; Masci, J.; Gambardella, L.M.; Schmidhuber, J. High-Performance Neural Networks for Visual Object Classification. *arXiv* **2011**, arXiv:1102.0183.
19. Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
20. Zeiler, M.D.; Fergus, R. Visualizing and Understanding Convolutional Networks. *arXiv* **2013**, arXiv:1311.2901.
21. Lin, M.; Chen, Q.; Yan, S. Network In Network. *arXiv* **2013**, arXiv:1312.4400.
22. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv* **2013**, arXiv:1312.6229.
23. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 15 October 2015; pp. 1–9. [CrossRef]
24. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
25. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
26. Xie, S.; Girshick, R.; Dollar, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Los Alamitos, CA, USA, 2017; pp. 5987–5995. [CrossRef]
27. Hu, J.; Shen, L.; Sun, G.; Albanie, S. Squeeze-and-Excitation Networks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*; IEEE: Piscataway, NJ, USA, 2019. [CrossRef]
28. Zhang, L.; Zhang, L.; Du, B. Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art. *IEEE Geosci. Remote Sens. Mag.* **2016**, *4*, 22–40. [CrossRef]
29. Zhu, X.X.; Tuia, D.; Mou, L.; Xia, G.S.; Zhang, L.; Xu, F.; Fraundorfer, F. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geosci. Remote Sens. Mag.* **2017**, *5*, 8–36. [CrossRef]

30. Cheng, G.; Yang, C.; Yao, X.; Guo, L.; Han, J. When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 2811–2821. [CrossRef]
31. Zhang, F.; Du, B.; Zhang, L. Scene classification via a gradient boosting random convolutional network framework. *IEEE Trans. Geosci. Remote Sens.* **2015**, *54*, 1793–1802. [CrossRef]
32. Zhong, Y.; Fei, F.; Zhang, L. Large patch convolutional neural networks for the scene classification of high spatial resolution imagery. *J. Appl. Remote Sens.* **2016**, *10*, 025006. [CrossRef]
33. Cheng, G.; Li, Z.; Yao, X.; Guo, L.; Wei, Z. Remote sensing image scene classification using bag of convolutional features. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 1735–1739. [CrossRef]
34. Yu, Y.; Gong, Z.; Wang, C.; Zhong, P. An unsupervised convolutional feature fusion network for deep representation of remote sensing images. *IEEE Geosci. Remote Sens. Lett.* **2017**, *15*, 23–27. [CrossRef]
35. Liu, Y.; Zhong, Y.; Fei, F.; Zhu, Q.; Qin, Q. Scene classification based on a deep random-scale stretched convolutional neural network. *Remote Sens.* **2018**, *10*, 444. [CrossRef]
36. Zhu, Q.; Zhong, Y.; Liu, Y.; Zhang, L.; Li, D. A deep-local-global feature fusion framework for high spatial resolution imagery scene classification. *Remote Sens.* **2018**, *10*, 568. [CrossRef]
37. Hu, F.; Xia, G.S.; Hu, J.; Zhang, L. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sens.* **2015**, *7*, 14680–14707. [CrossRef]
38. Penatti, O.A.; Nogueira, K.; Dos Santos, J.A. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Boston, MA, USA, 26 October 2015; pp. 44–51.
39. Marmanis, D.; Datcu, M.; Esch, T.; Stilla, U. Deep learning earth observation classification using ImageNet pretrained networks. *IEEE Geosci. Remote Sens. Lett.* **2015**, *13*, 105–109. [CrossRef]
40. Chaib, S.; Liu, H.; Gu, Y.; Yao, H. Deep feature fusion for VHR remote sensing scene classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 4775–4784. [CrossRef]
41. Li, E.; Xia, J.; Du, P.; Lin, C.; Samat, A. Integrating multilayer features of convolutional neural networks for remote sensing scene classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 5653–5665. [CrossRef]
42. Yuan, Y.; Fang, J.; Lu, X.; Feng, Y. Remote sensing image scene classification using rearranged local features. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 1779–1792. [CrossRef]
43. He, N.; Fang, L.; Li, S.; Plaza, A.; Plaza, J. Remote sensing scene classification using multilayer stacked covariance pooling. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 6899–6910. [CrossRef]
44. Lu, X.; Sun, H.; Zheng, X. A feature aggregation convolutional neural network for remote sensing scene classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 7894–7906. [CrossRef]
45. Minetto, R.; Segundo, M.P.; Sarkar, S. Hydra: An ensemble of convolutional neural networks for geospatial land classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 6530–6541. [CrossRef]
46. Wang, Q.; Liu, S.; Chanussot, J.; Li, X. Scene classification with recurrent attention of VHR remote sensing images. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 1155–1167. [CrossRef]
47. Castelluccio, M.; Poggi, G.; Sansone, C.; Verdoliva, L. Land use classification in remote sensing images by convolutional neural networks. *arXiv* **2015**, arXiv:1508.00092.
48. Liu, Y.; Suen, C.Y.; Liu, Y.; Ding, L. Scene classification using hierarchical Wasserstein CNN. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 2494–2509. [CrossRef]
49. Liu, Y.; Zhong, Y.; Qin, Q. Scene classification based on multiscale convolutional neural network. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 7109–7121. [CrossRef]
50. Fang, J.; Yuan, Y.; Lu, X.; Feng, Y. Robust space–frequency joint representation for remote sensing image scene classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 7492–7502. [CrossRef]
51. Xie, J.; He, N.; Fang, L.; Plaza, A. Scale-free convolutional neural network for remote sensing scene classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 6916–6928. [CrossRef]
52. Sun, H.; Li, S.; Zheng, X.; Lu, X. Remote sensing scene classification by gated bidirectional network. *IEEE Trans. Geosci. Remote Sens.* **2019**, *58*, 82–96. [CrossRef]
53. Chen, G.; Zhang, X.; Tan, X.; Cheng, Y.; Dai, F.; Zhu, K.; Gong, Y.; Wang, Q. Training small networks for scene classification of remote sensing images via knowledge distillation. *Remote Sens.* **2018**, *10*, 719. [CrossRef]
54. Zhang, B.; Zhang, Y.; Wang, S. A lightweight and discriminative model for remote sensing scene classification with multidilation pooling module. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *12*, 2636–2653. [CrossRef]
55. He, N.; Fang, L.; Li, S.; Plaza, J.; Plaza, A. Skip-connected covariance network for remote sensing scene classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 1461–1474. [CrossRef] [PubMed]
56. Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* **2017**, *29*, 2352–2449. [CrossRef] [PubMed]
57. Wang, W.; Yang, Y.; Wang, X.; Wang, W.; Li, J. Development of convolutional neural network and its application in image classification: A survey. *Opt. Eng.* **2019**, *58*, 040901. [CrossRef]
58. Dhruv, P.; Naskar, S. Image Classification Using Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN): A Review. In *Machine Learning and Information Processing*; Springer: Singapore, 2020; pp. 367–381.

59. Oquab, M.; Bottou, L.; Laptev, I.; Sivic, J. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 25 September 2014. [CrossRef]
60. Zagoruyko, S.; Komodakis, N. Learning to compare image patches via convolutional neural networks. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 15 October 2015; pp. 4353–4361. [CrossRef]
61. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G. Recent Advances in Convolutional Neural Networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
62. Tuytelaars, T.; Mikolajczyk, K. Local Invariant Feature Detectors: A Survey. *Found. Trends Comput. Graph. Vis.* **2007**, *3*, 177–280. [CrossRef]
63. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
64. Dumoulin, V.; Visin, F. A guide to convolution arithmetic for deep learning. *arXiv* **2016**, arXiv:1603.07285.
65. Hawkins, D. The Problem of Overfitting. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1–12. [CrossRef]
66. Liu, W.; Wang, Z.; Liu, X.; Zeng, N.; Liu, Y.; Alsaadi, F. A survey of deep neural network architectures and their applications. *Neurocomputing* **2016**, *234*, 11–26. [CrossRef]
67. Gulcehre, C.; Cho, K.; Pascanu, R.; Bengio, Y. Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks. *arXiv* **2013**, arXiv:1311.1780.
68. Yu, D.; Wang, H.; Chen, P.; Wei, Z. Mixed Pooling for Convolutional Neural Networks. In *International Conference on Rough Sets and Knowledge Technology*; Springer: Cham, Switzerland, 2014; pp. 364–375.
69. Zeiler, M.; Fergus, R. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. *arXiv* **2013**, arXiv:1301.3557.
70. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 346–361.
71. Gong, Y.; Wang, L.; Guo, R.; Lazebnik, S. Multi-scale Orderless Pooling of Deep Convolutional Activation Features. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 392–407.
72. Boureau, Y.L.; Ponce, J.; Lecun, Y. A Theoretical Analysis of Feature Pooling in Visual Recognition. 2010, pp. 111–118. Available online: https://dl.acm.org/doi/10.5555/3104322.3104338 (accessed on 1 June 2021).
73. Nair, V.; Hinton, G. Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. 2010, Volme 27, pp. 807–814. Available online: https://dl.acm.org/doi/10.5555/3104322.3104425 (accessed on 1 June 2021).
74. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. 2013. Available online: https://www.mendeley.com/catalogue/a4a3dd28-b56b-3e0c-ac53-2817625a2215/ (accessed on 1 June 2021).
75. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *IEEE Int. Conf. Comput. Vis. (ICCV 2015)* **2015**, *1502*, 1026–1034. [CrossRef]
76. Xu, B.; Wang, N.; Chen, T.; Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv* **2015**, arXiv:1505.00853.
77. Zeiler, M.; Fergus, R. Visualizing and Understanding Convolutional Neural Networks. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2013; Volume 8689.
78. Sainath, T.N.; Mohamed, A.r.; Kingsbury, B.; Ramabhadran, B. Deep convolutional neural networks for LVCSR. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 8614–8618. [CrossRef]
79. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.
80. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
81. Sainath, T.N.; Kingsbury, B.; Saon, G.; Soltau, H.; Rahman Mohamed, A.; Dahl, G.; Ramabhadran, B. Deep Convolutional Neural Networks for Large-scale Speech Tasks. *Neural Netw.* **2015**, *64*, 39–48. [CrossRef] [PubMed]
82. Wen, Y.; Zhang, K.; Li, Z.; Qiao, Y. A Discriminative Feature Learning Approach for Deep Face Recognition. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 499–515.
83. Liu, W.; Wen, Y.; Yu, Z.; Yang, M. Large-Margin Softmax Loss for Convolutional Neural Networks. *arXiv* **2016**, arXiv:1612.02295.
84. Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B.; Song, L. SphereFace: Deep Hypersphere Embedding for Face Recognition. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Los Alamitos, CA, USA, 2017; pp. 6738–6746. [CrossRef]
85. Wang, F.; Cheng, J.; Liu, W.; Liu, H. Additive margin softmax for face verification. *IEEE Signal Process. Lett.* **2018**, *25*, 926–930. [CrossRef]
86. Zhu, Q.; Zhang, P.; Wang, Z.; Ye, X. A new loss function for CNN classifier based on predefined evenly-distributed class centroids. *IEEE Access* **2019**, *8*, 10888–10895. [CrossRef]

87. Csurka, G.; Dance, C.; Fan, L.; Willamowski, J.; Bray, C. Visual categorization with bag of keypoints. In Proceedings of the European Conference on Workshop on Statistical Learning in Computer Vision, Prague, The Czech Republic; 2004.
88. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
89. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893. [CrossRef]
90. Ahonen, T.; Hadid, A.; Pietikainen, M. Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 2037–2041. [CrossRef]
91. Olshausen, B.A.; Field, D.J. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vis. Res.* **1997**, *37*, 3311–3325. [CrossRef]
92. Sivic, J.; Zisserman, A. Video Google: A Text Retrieval Approach to Object Matching in Videos. In Proceedings of the Proceedings Ninth IEEE International Conference on Computer Vision, Nice, France, 13–16 October 2003; Volume 2, pp. 1470–1477. [CrossRef]
93. Wang, J.; Yang, J.; Yu, K.; Lv, F.; Gong, Y. Locality-constrained Linear Coding for image classification. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 3360–3367. [CrossRef]
94. Perronnin, F.; Sánchez, J.; Mensink, T. Improving the Fisher Kernel for Large-Scale Image Classification. In *ECCV 2010—European Conference on Computer Vision*; Daniilidis, K., Maragos, P., Paragios, N., Eds.; Lecture Notes in Computer Science; Springer: Heraklion, Greece, 2010; Volume 6314, pp. 143–156. [CrossRef]
95. Cortes, C.; Vapnik, V. Support Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
96. Everingham, M.; Eslami, S.; Van Gool, L.; Williams, C.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes Challenge: A Retrospective. *Int. J. Comput. Vis.* **2014**, *111*, 98–136. [CrossRef]
97. Lin, Y.; Lv, F.; Zhu, S.; Yang, M.; Cour, T.; Yu, K.; Cao, L.; Huang, T. Large-scale image classification: Fast feature extraction and SVM training. In Proceedings of the CVPR 2011, Colorado Springs, CO, USA, 20–25 June 2011; pp. 1689–1696. [CrossRef]
98. Han, J.; Moraga, C. The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning. In *International Workshop on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 1995; Volume 930, pp. 195–201.
99. Zhou, L.; Pan, S.; Wang, J.; Vasilakos, A.V. Machine learning on big data: Opportunities and challenges. *Neurocomputing* **2017**, *237*, 350–361. [CrossRef]
100. Mangasarian, O.L.; Musicant, D.R. Data Discrimination via Nonlinear Generalized Support Vector Machines. In *Complementarity: Applications, Algorithms and Extensions*; Springer: Boston, MA, USA, 2001; pp. 233–251.
101. Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), Miami, FL, USA, 20–25 June; IEEE Computer Society: Los Alamitos, CA, USA, 2009; pp. 248–255. [CrossRef]
102. Luo, W.; Li, Y.; Urtasun, R.; Zemel, R. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. *arXiv* **2017**, arXiv:1701.04128.
103. Rosenblatt, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*; Cornell Aeronautical Lab Inc.: Buffalo, NY, USA, 1962.
104. Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef]
105. Arora, S.; Bhaskara, A.; Ge, R.; Ma, T. Provable Bounds for Learning Some Deep Representations. In Proceedings of the 31st International Conference on Machine Learning, Bejing, China, 22–24 June 2014; Volume 1.
106. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.
107. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA; 2016.
108. Wu, Z.; Shen, C.; Hengel, A. Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. *Pattern Recognit.* **2016**, *90*, 119–133. [CrossRef]
109. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity Mappings in Deep Residual Networks. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 630–645.
110. Bengio, Y.; Glorot, X. Understanding the difficulty of training deep feed forward neural networks. In Proceedings of the International Conference on Artificial Intelligenceand Statistics, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
111. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw. Publ. IEEE Neural Netw. Counc.* **1994**, *5*, 157–166. [CrossRef] [PubMed]
112. Saxe, A.M.; McClelland, J.L.; Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv* **2013**, arXiv:1312.6120.
113. Bordes, A.; Bottou, L.; Gallinari, P. SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. *J. Mach. Learn. Res.* **2009**, *10*, 1737–1754. [CrossRef]
114. Emin Orhan, A.; Pitkow, X. Skip Connections Eliminate Singularities. *arXiv* **2017**, arXiv:1701.09175.
115. Jégou, H.; Douze, M.; Sánchez, J.; Perez, P.; Schmid, C. Aggregating Local Image Descriptors into Compact Codes. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *34*, 1704–1716. [CrossRef] [PubMed]

116. Perronnin, F.; Dance, C. Fisher Kernels on Visual Vocabularies for Image Categorization. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8. [CrossRef]
117. Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; Weinberger, K. Deep Networks with Stochastic Depth. *arXiv* **2016**, arXiv:1603.09382.
118. Srivastava, R.K.; Greff, K.; Schmidhuber, J. Highway Networks. *arXiv* **2015**, arXiv:1505.00387.
119. Zagoruyko, S.; Komodakis, N. Wide Residual Networks. 2016, pp. 87.1–87.12. Available online: https://doi.org/10.5244/C.30.87 (accessed on 1 June 2021).
120. Yu, F.; Koltun, V.; Funkhouser, T. Dilated Residual Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 636–644. [CrossRef]
121. Veit, A.; Wilber, M.; Belongie, S. Residual Networks Behave like Ensembles of Relatively Shallow Networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 5 December 2016; NIPS'16. Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 550–558.
122. Targ, S.; Almeida, D.; Lyman, K. Resnet in Resnet: Generalizing Residual Architectures. *arXiv* **2016**, arXiv:1603.08029.
123. Ghiasi, G.; Lin, T.Y.; Le, Q.V. DropBlock: A Regularization Method for Convolutional Networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 3 December 2018; NIPS'18. Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 10750–10760.
124. Kolesnikov, A.; Beyer, L.; Zhai, X.; Puigcerver, J.; Yung, J.; Gelly, S.; Houlsby, N. Large Scale Learning of General Visual Representations for Transfer. *arXiv* **2019**, arXiv:1912.11370.
125. Brock, A.; De, S.; Smith, S.L.; Simonyan, K. High-Performance Large-Scale Image Recognition without Normalization. *arXiv* **2021**, arXiv:2102.06171.
126. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807. [CrossRef]
127. SIfre, L.; Mallat, S. Rigid-Motion Scattering for Texture Classification. *arXiv* **2014**, arXiv:1403.1687.
128. Jin, J.; Dundar, A.; Culurciello, E. Flattened Convolutional Neural Networks for Feedforward Acceleration. *arXiv* **2014**, arXiv:1412.5474.
129. Wang, M.; Liu, B.; Foroosh, H. Design of Efficient Convolutional Layers using Single Intra-channel Convolution, Topological Subdivisioning and Spatial "Bottleneck" Structure. *arXiv* **2016**, arXiv:1608.04337.
130. Zhang, X.; Li, Z.; Change Loy, C.; Lin, D. PolyNet: A Pursuit of Structural Diversity in Very Deep Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
131. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
132. Chen, Y.; Li, J.; Xiao, H.; Jin, X.; Yan, S.; Feng, J. Dual Path Networks. *arXiv* **2017**, arXiv:1707.01629.
133. Huang, G.; Liu, S.; Maaten, L.v.d.; Weinberger, K.Q. CondenseNet: An Efficient DenseNet Using Learned Group Convolutions. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2752–2761. [CrossRef]
134. Wang, F.; Jiang, M.; Qian, C.; Yang, S.; Li, C.; Zhang, H.; Wang, X.; Tang, X. Residual Attention Network for Image Classification. *arXiv* **2017**, arXiv:1704.06904.
135. Badrinarayanan, V.; Handa, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling. *arXiv* **2015**, arXiv:1505.07293.
136. Newell, A.; Yang, K.; Deng, J. Stacked Hourglass Networks for Human Pose Estimation. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 483–499.
137. Noh, H.; Hong, S.; Han, B. Learning Deconvolution Network for Semantic Segmentation. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1520–1528. [CrossRef]
138. Park, J.; Woo, S.; Lee, J.Y.; Kweon, I.S. BAM: Bottleneck Attention Module. *arXiv* **2018**, arXiv:1807.06514.
139. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. *arXiv* **2018**, arXiv:1807.06521.
140. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Vedaldi, A. *Gather-Excite: Exploiting Feature Context in Convolutional Neural Networks*; NIPS'18; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 9423–9433.
141. Li, X.; Wang, W.; Hu, X.; Yang, J. Selective Kernel Networks. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 510–519. [CrossRef]
142. Gao, Z.; Xie, J.; Wang, Q.; Li, P. Global Second-order Pooling Convolutional Networks. *arXiv* **2018**, arXiv:1811.12006.
143. Ionescu, C.; Vantzos, O.; Sminchisescu, C. Matrix Backpropagation for Deep Networks with Structured Layers. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 2965–2973. [CrossRef]
144. Lin, T.Y.; RoyChowdhury, A.; Maji, S. Bilinear CNNs for Fine-grained Visual Recognition. *arXiv* **2015**, arXiv:1504.07889.
145. Cui, Y.; Zhou, F.; Wang, J.; Liu, X.; Lin, Y.; Belongie, S. Kernel Pooling for Convolutional Neural Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3049–3058. [CrossRef]
146. Li, P.; Xie, J.; Wang, Q.; Gao, Z. Towards Faster Training of Global Covariance Pooling Networks by Iterative Matrix Square Root Normalization. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 947–955. [CrossRef]

147. Li, P.; Xie, J.; Wang, Q.; Zuo, W. Is Second-Order Information Helpful for Large-Scale Visual Recognition? In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2089–2097. [CrossRef]
148. Chen, Y.; Kalantidis, Y.; Li, J.; Yan, S.; Feng, J. A_2-Nets: Double Attention Networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Red Hook, NY, USA, December 2018; NIPS'18; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 350–359.
149. Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; Hu, Q. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. *arXiv* **2019**, arXiv:1910.03151.
150. Hou, Q.; Zhou, D.; Feng, J. Coordinate Attention for Efficient Mobile Network Design. *arXiv* **2021**, arXiv:2103.02907.
151. Linsley, D.; Shiebler, D.; Eberhardt, S.; Serre, T. Learning what and where to attend. *arXiv* **2018**, arXiv:1805.08819.
152. Bello, I.; Zoph, B.; Le, Q.; Vaswani, A.; Shlens, J. Attention Augmented Convolutional Networks. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 3285–3294. [CrossRef]
153. Misra, D.; Nalamada, T.; Uppili Arasanipalai, A.; Hou, Q. Rotate to Attend: Convolutional Triplet Attention Module. *arXiv* **2020**, arXiv:2010.03045.
154. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local Neural Networks. *arXiv* **2017**, arXiv:1711.07971.
155. Cao, Y.; Xu, J.; Lin, S.; Wei, F.; Hu, H. GCNet: Non-local Networks Meet Squeeze-Excitation Networks and Beyond. *arXiv* **2019**, arXiv:1904.11492.
156. Liu, J.J.; Hou, Q.; Cheng, M.M.; Wang, C.; Feng, J. Improving Convolutional Networks with Self-Calibrated Convolutions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10093–10102. [CrossRef]
157. Huang, Z.; Wang, X.; Huang, L.; Huang, C.; Wei, Y.; Liu, W. CCNet: Criss-Cross Attention for Semantic Segmentation. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 27 October–2 November 2019, Seoul, Korea; pp. 603–612. [CrossRef]
158. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
159. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2015**, arXiv:1510.00149.
160. Gholami, A.; Kwon, K.; Wu, B.; Tai, Z.; Yue, X.; Jin, P.; Zhao, S.; Keutzer, K. SqueezeNext: Hardware-Aware Neural Network Design. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, USA, 18–22 June 2018; pp. 1719–171909. [CrossRef]
161. 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.
162. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520. [CrossRef]
163. Howard, A.; Pang, R.; Adam, H.; Le, Q.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.C.; Tan, M.; Chu, G.; et al. Searching for MobileNetV3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1314–1324. [CrossRef]
164. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *arXiv* **2018**, arXiv:1807.11626.
165. Yang, T.J.; Howard, A.; Chen, B.; Zhang, X.; Go, A.; Sandler, M.; Sze, V.; Adam, H. NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications. *arXiv* **2018**, arXiv:1804.03230.
166. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *arXiv* **2017**, arXiv:1707.01083.
167. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *Computer Vision—ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 122–138.
168. Wang, R.J.; Li, X.; Ling, C.X. Pelee: A Real-Time Object Detection System on Mobile Devices. *arXiv* **2018**, arXiv:1804.06882.
169. Wang, C.Y.; Mark Liao, H.Y.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; pp. 1571–1580. [CrossRef]
170. Zhang, Q.; Li, J.; Yao, M.; Song, L.; Zhou, H.; Li, Z.; Meng, W.; Zhang, X.; Wang, G. VarGNet: Variable Group Convolutional Neural Network for Efficient Embedded Computing. *arXiv* **2019**, arXiv:1907.05653.
171. Yan, M.; Zhao, M.; Xu, Z.; Zhang, Q.; Wang, G.; Su, Z. VarGFaceNet: An Efficient Variable Group Convolutional Neural Network for Lightweight Face Recognition. *arXiv* **2019**, arXiv:1910.04985.
172. Lee, Y.; Hwang, J.w.; Lee, S.; Bae, Y.; Park, J. An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Long Beach, CA, USA, 16–17 June 2019.

173. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. Scaled-YOLOv4: Scaling Cross Stage Partial Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 13029–13038.
174. Lee, Y.; Park, J. CenterMask: Real-Time Anchor-Free Instance Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
175. Wang, J.; Sun, K.; Cheng, T.; Jiang, B.; Deng, C.; Zhao, Y.; Liu, D.; Mu, Y.; Tan, M.; Wang, X.; et al. Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 3349–3364. [CrossRef] [PubMed]
176. Yu, C.; Xiao, B.; Gao, C.; Yuan, L.; Zhang, L.; Sang, N.; Wang, J. Lite-HRNet: A Lightweight High-Resolution Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 10440–10450.
177. Huang, Y.; Cheng, Y.; Bapna, A.; Firat, O.; Chen, M.X.; Chen, D.; Lee, H.; Ngiam, J.; Le, Q.V.; Wu, Y.; et al. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. *arXiv* **2018**, arXiv:1811.06965.
178. Lu, Z.; Pu, H.; Wang, F.; Hu, Z.; Wang, L. The Expressive Power of Neural Networks: A View from the Width. *arXiv* **2017**, arXiv:1709.02540.
179. Raghu, M.; Poole, B.; Kleinberg, J.; Ganguli, S.; Sohl-Dickstein, J. On the Expressive Power of Deep Neural Networks. *arXiv* **2016**, arXiv:1606.05336.
180. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.
181. Tan, M.; Le, Q.V. EfficientNetV2: Smaller Models and Faster Training. *arXiv* **2021**, arXiv:2104.00298.
182. Touvron, H.; Vedaldi, A.; Douze, M.; Jégou, H. Fixing the train-test resolution discrepancy. *arXiv* **2019**, arXiv:1906.06423.
183. Radosavovic, I.; Kosaraju, R.P.; Girshick, R.; He, K.; Dollár, P. Designing Network Design Spaces. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 10425–10433. [CrossRef]
184. Suyog Gupta, M.T. Efficientnet-Edgetpu: Creating Accelerator-Optimized Neural Networks with Automl. 2019. Available online: https://ai.googleblog.com/2019/08/efficientnet-edgetpu-creating.html (accessed on 1 June 2021).
185. Hoffer, E.; Weinstein, B.; Hubara, I.; Ben-Nun, T.; Hoefler, T.; Soudry, D. Mix & Match: Training convnets with mixed image sizes for improved accuracy, speed and scale resiliency. *arXiv* **2019**, arXiv:1908.08986.
186. Denton, E.; Zaremba, W.; Bruna, J.; Lecun, Y.; Fergus, R. Exploiting Linear Structure within Convolutional Networks for Efficient Evaluation. 2014, Volume 2. Available online: https://dl.acm.org/doi/abs/10.5555/2968826.2968968 (accessed on 1 June 2021).
187. Wen, W.; Xu, C.; Wu, C.; Wang, Y.; Chen, Y.; Li, H. Coordinating Filters for Faster Deep Neural Networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
188. Hassibi, B.; Stork, D.G. *Second Order Derivatives for Network Pruning: Optimal Brain Surgeon*; Morgan Kaufmann: Burlington, MA, USA, 1993; pp. 164–171.
189. Cun, Y.L.; Denker, J.S.; Solla, S.A. Optimal Brain Damage. In *Advances in Neural Information Processing Systems 2*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1990; pp. 598–605.
190. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both Weights and Connections for Efficient Neural Networks. *arXiv* **2015**, arXiv:1506.02626.
191. Han, S.; Pool, J.; Narang, S.; Mao, H.; Gong, E.; Tang, S.; Elsen, E.; Vajda, P.; Paluri, M.; Tran, J.; et al. DSD: Dense-Sparse-Dense Training for Deep Neural Networks. *arXiv* **2016**, arXiv:1607.04381.
192. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning Filters for Efficient ConvNets. *arXiv* **2016**, arXiv:1608.08710.
193. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *arXiv* **2017**, arXiv:1712.05877.
194. Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv* **2018**, arXiv:1806.08342.
195. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 525–542.
196. Soudry, D.; Hubara, I.; Meir, R. Expectation Backpropagation: Parameter-Free Training of Multilayer Neural Networks with Continuous or Discrete Weights. 2014, Volume 2. Available online: https://www.mendeley.com/catalog/expectation-backpropagation-parameterfree-training-multilayer-neural-networks-real-discrete-weights/ (accessed on 1 June 2021).
197. Wu, J.; Leng, C.; Wang, Y.; Hu, Q.; Cheng, J. Quantized Convolutional Neural Networks for Mobile Devices. *arXiv* **2015**, arXiv:1512.06473.
198. Zhou, A.; Yao, A.; Guo, Y.; Xu, L.; Chen, Y. Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights. *arXiv* **2017**, arXiv:1702.03044.
199. Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; Zou, Y. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv* **2016**, arXiv:1606.06160.
200. Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or −1. *arXiv* **2016**, arXiv:1602.02830.
201. Baker, B.; Gupta, O.; Naik, N.; Raskar, R. Designing Neural Network Architectures using Reinforcement Learning. *arXiv* **2016**, arXiv:1611.02167.
202. Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.J.; Fei-Fei, L.; Yuille, A.; Huang, J.; Murphy, K. Progressive Neural Architecture Search. *arXiv* **2017**, arXiv:1712.00559.

203. Pham, H.; Guan, M.Y.; Zoph, B.; Le, Q.V.; Dean, J. Efficient Neural Architecture Search via Parameter Sharing. *arXiv* **2018**, arXiv:1802.03268.
204. Zoph, B.; Le, Q.V. Neural Architecture Search with Reinforcement Learning. *arXiv* **2016**, arXiv:1611.01578.
205. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. *arXiv* **2017**, arXiv:1707.07012.
206. Cai, H.; Zhu, L.; Han, S. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. *arXiv* **2018**, arXiv:1812.00332.
207. Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable Architecture Search. *arXiv* **2018**, arXiv:1806.09055.
208. Xie, S.; Zheng, H.; Liu, C.; Lin, L. SNAS: Stochastic Neural Architecture Search. *arXiv* **2018**, arXiv:1812.09926.
209. Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; Sun, J. Single Path One-Shot Neural Architecture Search with Uniform Sampling. *arXiv* **2019**, arXiv:1904.00420.
210. Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; Keutzer, K. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 10726–10734. [CrossRef]
211. Wan, A.; Dai, X.; Zhang, P.; He, Z.; Tian, Y.; Xie, S.; Wu, B.; Yu, M.; Xu, T.; Chen, K.; et al. FBNetV2: Differentiable Neural Architecture Search for Spatial and Channel Dimensions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
212. Dai, X.; Wan, A.; Zhang, P.; Wu, B.; He, Z.; Wei, Z.; Chen, K.; Tian, Y.; Yu, M.; Vajda, P.; et al. FBNetV3: Joint Architecture-Recipe Search using Predictor Pretraining. *arXiv* **2020**, arXiv:2006.02049.
213. Dai, X.; Zhang, P.; Wu, B.; Yin, H.; Sun, F.; Wang, Y.; Dukhan, M.; Hu, Y.; Wu, Y.; Jia, Y.; et al. ChamNet: Towards Efficient Network Design through Platform-Aware Model Adaptation. *arXiv* **2018**, arXiv:1812.08934.
214. He, Y.; Han, S. ADC: Automated Deep Compression and Acceleration with Reinforcement Learning. 2018. Available online: https://deeplearn.org/arxiv/26016/adc:-automated-deep-compression-and-acceleration-with-reinforcement-learning (accessed on 1 June 2021).
215. Buciluǎ, C.; Caruana, R.; Niculescumizil, A. Model compression. In Proceedings of the Knowledge Discovery and Data Mining, New York, NY, USA, August 2006; pp. 535–541. [CrossRef]
216. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
217. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, December 2017; NIPS'17. Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6000–6010.
218. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**, arXiv:1409.0473.
219. 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 16x16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929.
220. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jegou, H. Training data-efficient image transformers & distillation through attention. *arXiv* **2020**, arXiv:2012.12877.
221. Wang, W.; Xie, E.; Li, X.; Fan, D.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. *arXiv* **2021**, arXiv:2102.12122.
222. Han, K.; Xiao, A.; Wu, E.; Guo, J.; Xu, C.; Wang, Y. Transformer in Transformer. *arXiv* **2021**, arXiv:2103.00112.
223. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv* **2021**, arXiv:2103.14030.
224. Srinivas, A.; Lin, T.Y.; Parmar, N.; Shlens, J.; Abbeel, P.; Vaswani, A. Bottleneck Transformers for Visual Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 16519–16529.
225. Yuan, K.; Guo, S.; Liu, Z.; Zhou, A.; Yu, F.; Wu, W. Incorporating Convolution Designs into Visual Transformers. *arXiv* **2021**, arXiv:2103.11816.
226. Dai, Z.; Liu, H.; Le, Q.V.; Tan, M. CoAtNet: Marrying Convolution and Attention for All Data Sizes. *arXiv* **2021**, arXiv:2106.04803.
227. Wu, H.; Xiao, B.; Codella, N.; Liu, M.; Dai, X.; Yuan, L.; Zhang, L. CvT: Introducing Convolutions to Vision Transformers. *arXiv* **2021**, arXiv:2103.15808.
228. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In *Computer Vision—ECCV 2020*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 213–229.
229. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable {DETR}: Deformable Transformers for End-to-End Object Detection. *arXiv* **2020**, arXiv:2010.04159.
230. Dai, Z.; Cai, B.; Lin, Y.; Chen, J. UP-DETR: Unsupervised Pre-Training for Object Detection with Transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 1601–1610.

231. Lee, D.H. Pseudo-label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. Available online: https://www.kaggle.com/blobs/download/forum-message-attachment-files/746/pseudo_label_final.pdf (accessed on 1 June 2021).
232. Riloff, E. Automatically generating extraction patterns from untagged text. In Proceedings of the National Conference on Artificial Intelligence, Portland, OR, USA, August 1996; pp. 1044–1049.
233. Scudder, H. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theory* **1965**, *11*, 363–371. [CrossRef]
234. Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, Cambridge, MA, USA, June 1995; pp. 189–196.
235. Pham, H.; Dai, Z.; Xie, Q.; Le, Q.V. Meta Pseudo Labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 11557–11568.
236. Xie, Q.; Luong, M.T.; Hovy, E.; Le, Q.V. Self-Training with Noisy Student Improves ImageNet Classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
237. Yalniz, I.Z.; Jégou, H.; Chen, K.; Paluri, M.; Mahajan, D. Billion-scale semi-supervised learning for image classification. *arXiv* **2019**, arXiv:1905.00546.
238. Cubuk, E.D.; Zoph, B.; Shlens, J.; Le, Q.V. Randaugment: Practical Automated Data Augmentation with a Reduced Search Space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Seattle, WA, USA, 14–19 June 2020.
239. Arazo, E.; Ortego, D.; Albert, P.; O'Connor, N.E.; McGuinness, K. Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
240. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
241. Raghu, M.; Zhang, C.; Kleinberg, J.; Bengio, S. Transfusion: Understanding Transfer Learning for Medical Imaging. *arXiv* **2019**, arXiv:1902.07208.
242. He, K.; Girshick, R.; Dollar, P. Rethinking ImageNet Pre-Training. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
243. DeVries, T.; Taylor, G.W. Dataset Augmentation in Feature Space. *arXiv* **2017**, arXiv:1702.05538.
244. Simard, P.; Steinkraus, D.; Platt, J. Best practices for convolutional neural networks applied to visual document analysis. In Proceedings of the Seventh International Conference on Document Analysis and Recognition, Edinburgh, UK, 6 August 2003; pp. 958–963. [CrossRef]
245. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization. *arXiv* **2017**, arXiv:1710.09412.
246. Yun, S.; Han, D.; Oh, S.J.; Chun, S.; Choe, J.; Yoo, Y. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. *arXiv* **2019**, arXiv:1905.04899.
247. Cubuk, E.D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q.V. AutoAugment: Learning Augmentation Policies from Data. *arXiv* **2018**, arXiv:1805.09501.
248. Ho, D.; Liang, E.; Chen, X.; Stoica, I.; Abbeel, P. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; Volume 97, pp. 2731–2741.
249. Lim, S.; Kim, I.; Kim, T.; Kim, C.; Kim, S. Fast AutoAugment. *arXiv* **2019**, arXiv:1905.00397.
250. Zoph, B.; Cubuk, E.D.; Ghiasi, G.; Lin, T.Y.; Shlens, J.; Le, Q.V. Learning Data Augmentation Strategies for Object Detection. In *Computer Vision—ECCV 2020*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 566–583.
251. Harris, E.; Marcu, A.; Painter, M.; Niranjan, M.; Prügel-Bennett, A.; Hare, J. FMix: Enhancing Mixed Sample Data Augmentation. *arXiv* **2020**, arXiv:2002.12047.
252. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
253. Hinton, G.; Srivastava, N.; Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited* **2012**, *14*, 2.
254. Foret, P.; Kleiner, A.; Mobahi, H.; Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv* **2020**, arXiv:2010.01412.
255. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
256. Wu, Y.; He, K. Group Normalization. *arXiv* **2018**, arXiv:1803.08494.
257. Merity, S.; Keskar, N.S.; Socher, R. Regularizing and optimizing LSTM language models. *arXiv* **2017**, arXiv:1708.02182.
258. Balduzzi, D.; Frean, M.; Leary, L.; Lewis, J.; Ma, K.W.D.; McWilliams, B. The shattered gradients problem: If resnets are the answer, then what is the question? In Proceedings of the International Conference on Machine Learning, Sydney, NSW, Australia, August 2017; pp. 342–350.
259. Summers, C.; Dinneen, M.J. Four things everyone should know to improve batch normalization. *arXiv* **2019**, arXiv:1906.03548.
260. Singh, S.; Shrivastava, A. EvalNorm: Estimating Batch Normalization Statistics for Evaluation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.

261. Touvron, H.; Vedaldi, A.; Douze, M.; Jégou, H. Fixing the train-test resolution discrepancy: FixEfficientNet. *arXiv* **2020**, arXiv:2003.08237.
262. Miramontes-de Leon, G.; Moreno-Baez, A.; Magallanes-Quintanar, R.; Valdez-Cepeda, R. Assessment in Subsets of MNIST Handwritten Digits and Their Effect in the Recognition Rate. *J. Pattern Recognit. Res.* **2011**, *2*, 244–252. [CrossRef]
263. Li, H. CIFAR10-DVS: An event-stream dataset for object classification. *Front. Neurosci.* **2017**, *11*, 309. [CrossRef] [PubMed]