A comprehensive survey on convolutional neural network in medical image analysis

Article *in* Multimedia Tools and Applications · August 2020

DOI: 10.1007/s11042-020-09634-7

4 authors:

Xujing Yao
University of Leicester
9 PUBLICATIONS **214** CITATIONS
SEE PROFILE

Xinyue Wang
2 PUBLICATIONS **27** CITATIONS
SEE PROFILE

Shuihua Wang
University of Leicester
791 PUBLICATIONS **23,820** CITATIONS
SEE PROFILE

Yu-Dong Zhang
University of Leicester
654 PUBLICATIONS **20,953** CITATIONS
SEE PROFILE

A comprehensive survey on Convolutional Neural Network in medical image analysis

Xujing Yao[1], Xinyue Wang[1], Shui-Hua Wang[1,2,3,*], Yu-Dong Zhang[1,2,*],

1 School of Informatics, University of Leicester, Leicester, LE1 7RH, UK

2 Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia

3 School of Architecture Building and Civil engineering, Loughborough University, Loughborough, LE11 3TU, UK

Email: XY (xy147@le.ac.uk), XW (fireswxy@163.com), SHW (shuihuawng@ieee.org), YDZ (yudongzhang@ieee.org)

Correspondence should be addressed to SHW & YDZ

Abstract: CNN is inspired from Primary Visual (V1) neurons. It is a typical deep learning technique and can help teach machine how to see and identify objects. In the most recent decade, deep learning develops rapidly and has been well used in various fields of expertise such as computer vision and natural language processing. As the representative algorithm of deep learning, Convolution Neural Network (CNN) has been regarded as a breakthrough of historic significance in image processing and visual recognition tasks since the astonishing results achieved on ImageNet Large Scale Visual Recognition Competition (ILSVRC). Unlike methods based on handcrafted features, CNN models can build high-level features from low-level ones in a data-driven fashion and have displayed great potential in medical image analysis among the aspects of segmentation of histological images identification, lesion detection, tissue classification, etc. This paper provides a review on CNN from the perspectives of its basic mechanism introduction, structure, typical architecture and main application in medical image analysis through analyzing over 100 references from Google Scholar, PubMed, Web of Science and various sources published from 1958 to 2020.

Keywords: Deep learning; Feedforward Neural Network; Convolutional neural network; Breast Cancer; Lung Nodule; Brain Tumor; Medical image analysis

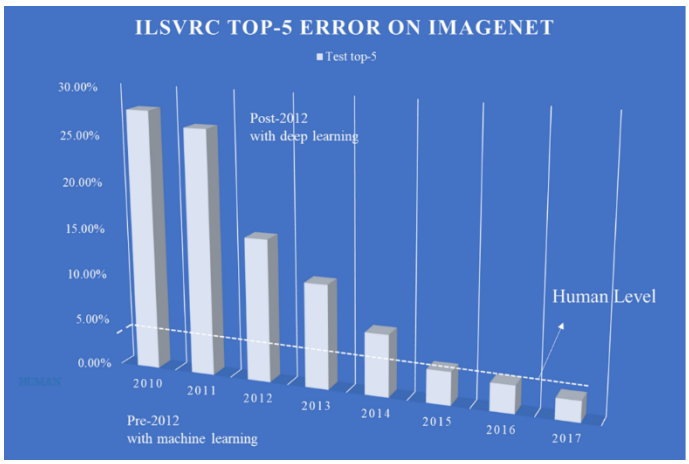
1. BACKGROUND

Deep Learning (DL) is a promising research direction in the Machine Learning (ML) area. By means of utilizing DL technique, computational models are capable to learn the features and representation of data with a process of extraction. This is an innovation that brings dramatical improvements in image recognition, object detection, voice recognition, genomic science and many other domains in real life [1, 2]

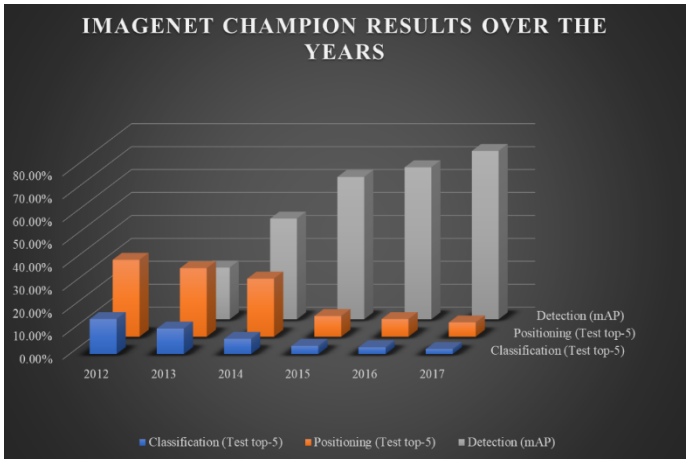
The study of machine learning is inspired in part by the exploration of human brain. Around 300B.C., the concept of understanding the brain has attracted a lot of attention of researchers, among which the celebrated philosopher Aristotle is considered as the forerunner, due to the reason that he opens the history of human trying to learn the brain [3]. In 1943, the team of McCulloch and Pitts [4] brought the earlier neural network model: MCP Neural Model into the attention of public. About one decade later, Rosenblatt [5] introduced the perceptron based on this MCP model to complete the classification task. However, just one decade after the introduction, a significant flaw was identified by Minsky and Papert [6] that the perceptron cannot handle in the nonlinear classification problems, which hindered the

subsequent progress of neural network for approximately twenty years [7]. In 1986, a book published by Rumelhart, Hinton [8] bring the turnaround for the development of neural network. The book detailedly introduced the back propagation algorithm as well as the whole derived process of utilizing back propagation algorithm and the activation function Sigmoid to deal with the nonlinear problems. This not only led to a great deal of uproar in academia but also attracted a sharp increase number of people into the research of deep learning. In 1997, the CNN model is presented [9-11]. In the same year, Hochreiter and Schmidhuber [12] discovered Long Short-Term Memory (LSTM) Networks model. After the introduction of ReLU, neural network has so far been known as the most outstanding tool to deal with learning problems [13, 14].

Because of the efforts of so many great scholars, as shown in Figure 1, deep learning is becoming more and more accurate and has evolved to perform many complex tasks and achieve loads of unimaginable breakthroughs [15-18].



(a) ILSVRC Top-5 Error on ImageNet



(b) ImageNet Champion results over the years

Figure 1 Breakthroughs in ImageNet competitions over recent years

This survey will introduce the application of CNN technology in analysing medical images. The reason why we chose to introduce CNN is that it can process image information better than deep neural network (DNN). DNN is one of the simplest neural networks. In DNN, each neuron belongs to a different layer and is connected with all neurons in the previous layer. Signal is transmitted one-way from the input layer to the output layer. While CNN is a feedforward neural network through convolution

computation, which is proposed by the biological receptive field mechanism and has translational invariance. It utilizes the convolution kernel to maximize the application of local information and retain the planar structure information. DNN is input in vector form, without considering the structure information of the plane. In the field of image processing, the plane information is very significant, so CNN performs better than DNN in medical image analysis [19].

A simple flowchart of this survey is organized as Figure 2. The purpose and contribution of this review is to: (i) introduce the theoretical basis and working principle of CNN in detail; (ii) elaborate description of the popular forms of various CNN basic structures; (iii) detailedly explain classic models and trends from the rise of CNN to the present. (iv) demonstrate the outstanding contributions of CNN models in the medical field from recent years. It can be found from our review that CNN model is increasingly being used in medical fields where it is difficult to find the basis for diagnosis, or hard to achieve high accuracy or fast diagnostic analysis by just utilizing human power. Perhaps in the next few years, humans will be able to use CNN to solve medical problems that are currently difficult to understand and analyze from a medical perspective.

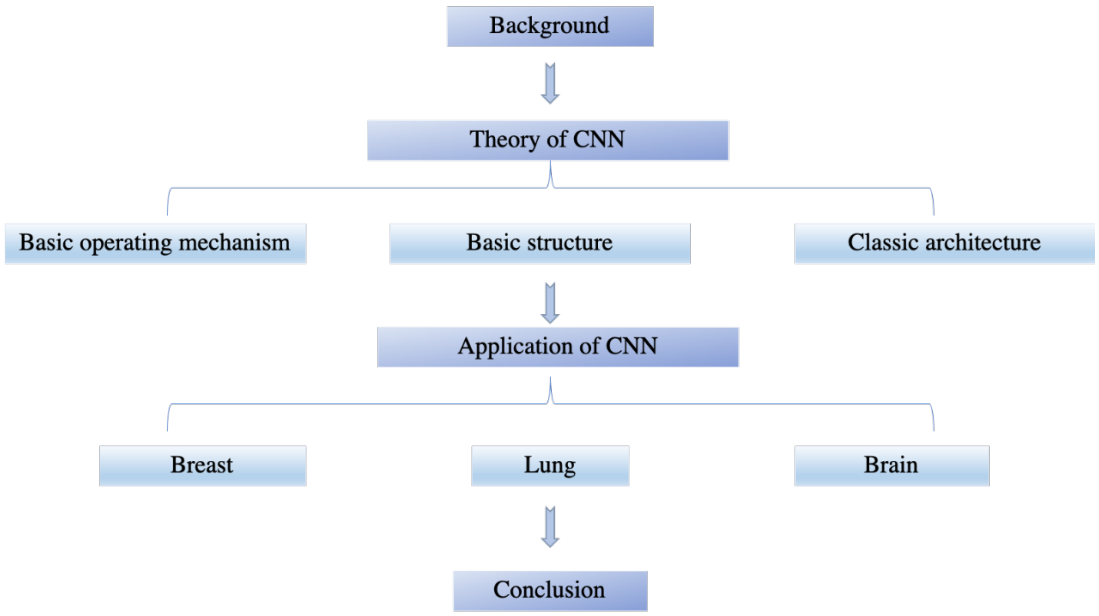


Figure 2 Simple flowchart of this survey

Section 2, 3 and 4 will explain CNN from three aspects: the basic operating mechanism, the basic structure and typical architectures of CNN. The operating mechanism section introduces the operation unit, operation thought, gradient vanishing, feedforward propagation and feedback propagation. The basic structure section describes principles and details of convolutional layer, pooling layer and fully connected layer. The classic architecture section reviews the theory and advantages of several typical architectures such as AlexNet, VGG-Net, Inception, ResNet, R-CNN, SPP-Net and so on.

2. BASIC OPERATING MECHANISM

2.1 Operation Unit: Neuron

In 1943, the MCP Model introduced by McCulloch and Pitts [4] is very similar with biological

neural network. From a biological kind of view, every neuron is linked with other neurons. When the neuron is activated, which means the potential of neuron is over a threshold, it will send chemical substances to the connected neurons to transform the potential of other neurons. MCP model has the same mechanism. The circle in Figure 3 represents neuron, which receives the input signals ($x_1 \dots x_m$) from the other neurons. These input signals will get different weights ($w_1 \dots w_m$) and bias (b) and then go to the neuron. the output $\hat{y}$ will be got through the activation function [14, 20].

$$\hat{y} = \sigma(z) = \sigma(\sum_{j=1}^{m} w_j x_j + b) \quad (1)$$

Where we assume the input of the activation function as z

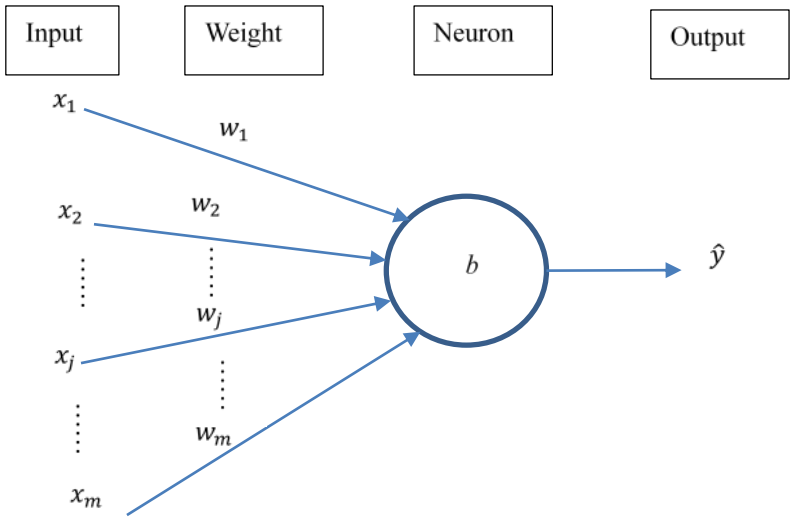
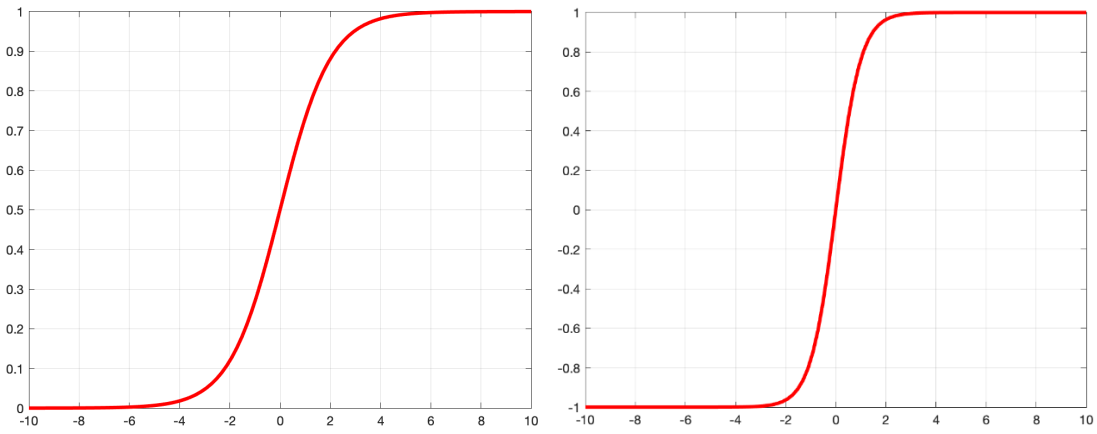


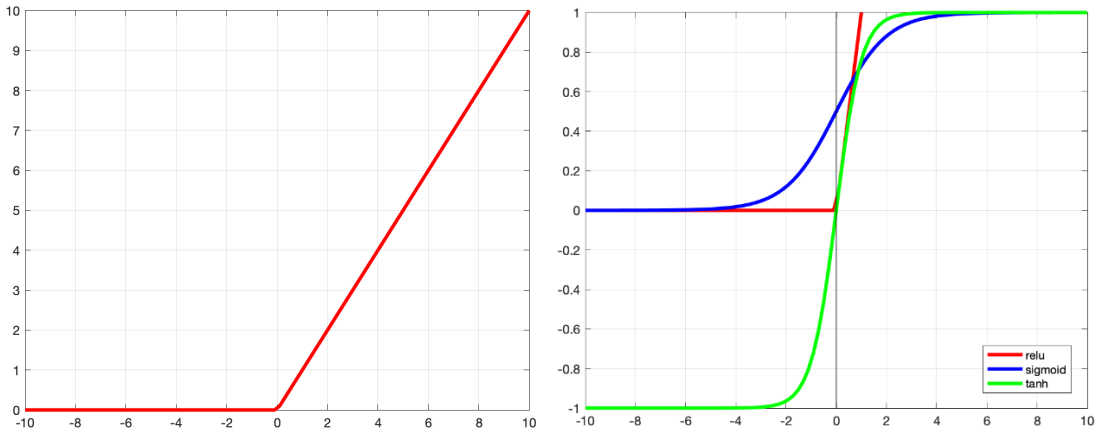
Figure 3 The McCulloch and Pitts model of a neuron

In a multi-layer neural network, each neuron node will accept the output of previous layer as its input value and then pass the value to the following layer. This function relationship between neuron nodes is defined as the activation function. If you just utilize the linear activation function or do not use the activation function, all things are about calculating the linear activation function, which is very limited, so in order to get more possibilities, the nonlinear activation function is born [21-23].

2.2 Activation Function and Loss Function



(a) Sigmoid Function (b) Hyperbolic Tangent Function



(c) Rectified Linear Unit (d) Summary Graph

Figure 4 Activation functions

2.2.1. Sigmoid Function

Sigmoid Function aims to alter the continuous real value of the input z to the output $\sigma(z)$ of a range between 0 and 1.

$$\sigma_{\text{Sigmoid}}(z) = \frac{1}{1+e^{-z}} \tag{2}$$

It's clear from Equation (2) that if the input is an extremely large negative number, the output is going to be 0. If the input value is an extremely large positive number, the output will be 1. But this function also has obvious disadvantages that it consumes loads of calculation time, and will be easy to produce gradient disappearance and gradient explosion [24]. In addition, as displayed in Figure 4(a), its data is not centralized and symmetric with (0,0.5) as the center, which will bring a negative effect on the gradient calculation of the hidden layer.

2.2.2. Hyperbolic Tangent Function (Tanh)

Tanh function performs superior to the sigmoid function in hidden layer. It can be viewed as the shifted version of sigmoid as shown in Figure 4(b). The equation of Tanh is shown as:

$$\sigma_{\text{Tanh}}(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \tag{3}$$

Although it achieves a similar effect of data centralization, in practice, sigmoid and tanh functions tend to reach saturation at the end value, especially in the deep network training, which leads to a slow training speed.

2.2.3. Rectified Linear Unit Function (ReLU)

It can be seen from Figure 4(c), ReLU solves the gradient disappearance problem when there is a positive input [25]. The equation is shown below

$$\sigma_{\text{ReLU}}(z) = \max(0, z) \tag{4}$$

Moreover, ReLU costs much less computation time when compared with sigmoid and tanh function. But the gradient vanishing problem remains and the data distribution of ReLU is not centralized when

there is a negative input [26].

2.2.4. Loss Function

After the activation function, the calculated value $\hat{y}_j$ is got. However, the calculated value cannot equal to the actual value y_j due to some unexplained factors. The loss function aims to minimize the distance between each actual value and the predicted value in the training set, which refers to, the sum of the error $J(w)$ between $\hat{y}_j$ and y_j through utilizing the mean square error principle shown below:

$$J(w) = \frac{1}{2}\sum_{j=1}^{m}(\hat{y}_j - y_j)^2 \quad (5)$$

Where *w* represents weight, which is the coefficient of linear relation

However, the mean square error has many disadvantages when it is used in the back propagation algorithm which will be introduced in Section 2.41.5.1.2.4.2.

2.3 Operation Thought: gradient descent

Most deep learning algorithms involve many kinds of optimizations. Gradient descent method is one of the optimizations which used to minimize the loss function. In this method, the minimum loss function and the value of model parameters are obtained by step-by-step iteration [21].

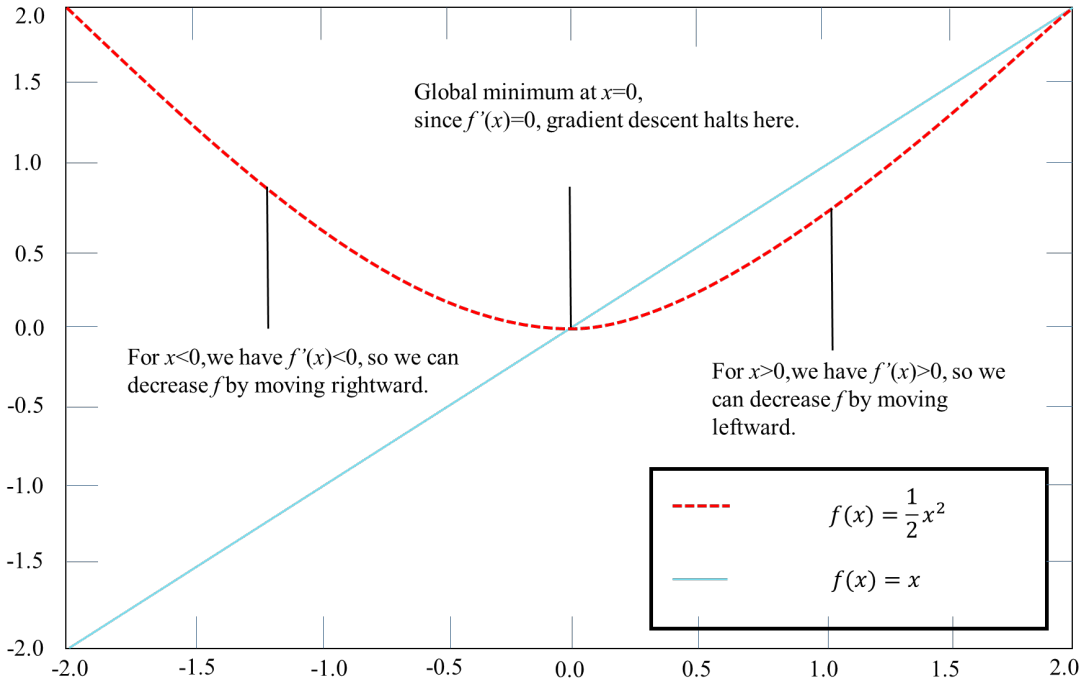


Figure 5 The relationship between a function and the derivatives of a function

In 1847, Cauchy, Augustin discovered gradient descent. Assume there is a function:

$$y = f(x) \quad (6)$$

Where both *x* and *y* are real numbers

The derivative of the function is $[dy/dx]$ or $f'(x)$, which presents the slope of *f*(*x*) at point *x*. Besides, it also shows if the input has a little change like ϵ, the output can also be represented by the

slope and the original output, the formula is shown below [27]:

$$f(x) \leftarrow f(x+\epsilon) = f(x) + \epsilon f'(x) \tag{7}$$

Where $\leftarrow$ represents the assignment operation.

With this equation, it is easy to control the output decrease. When the slope is negative the input should become bigger, and when the slope is positive the input should become smaller in order to get a smaller output value as the Figure 5 shows.

Since the loss function should be smallest, the equation will run until it meets the three possibilities as shown is Figure 6. First, the output is the minimum value within limits, which is called local minimum. Second, the output is the lowest point in the whole function, which is called global minimum. And the third possibility is that the output is a saddle point. A saddle point refers to a critical point which is neither maximum nor minimum.

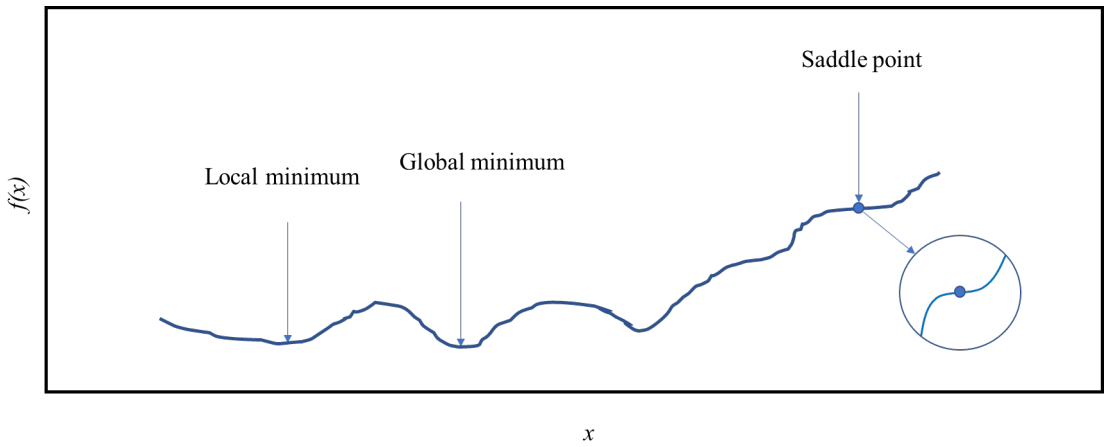
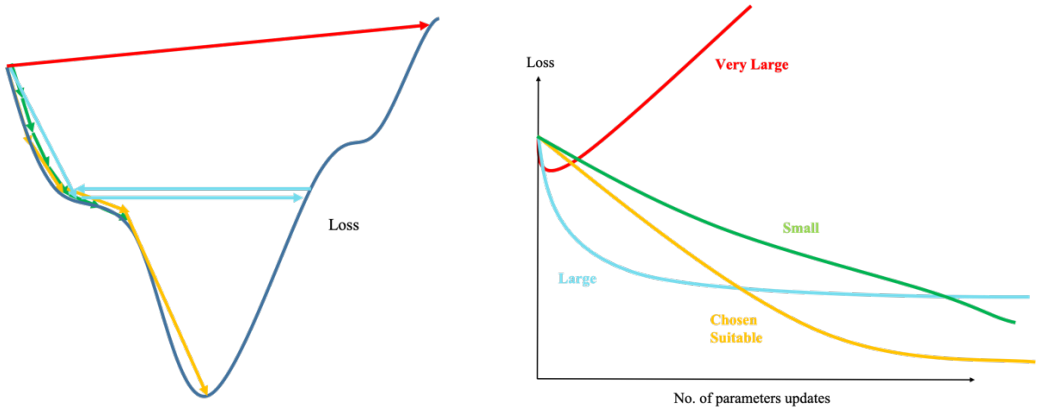


Figure 6 Three possibilities the formula will stop when it meets

In order to avoid the output which comes from the saddle point, it is important to choose the suitable learning rate ϵ. As displayed in Figure 7, if the learning rate is chosen suitable, the minimum value can be found easily as the orange line. But if ϵ is too small as the green line, the time to run the equation will increase and the output may be generated from a saddle point or local minimum. If ϵ is large as the blue line, it is impossible to find the minimum. If ϵ is too large as the red line, the equation will loss control and the output is far from the right point. Therefore, it is important to control learning rate ϵ. It is convenient and visualized to draw a graph about loss and parameters change. And from Figure 7, it is obvious that the orange line gives us the best learning rate.



(a) (b)

Figure 7 The relationship between loss and parameters change

Except choosing the suitable learning rate, different kinds of gradient descent can improve the loss function in different levels. We assume the original equation is

$$\hat{y} = b + w_1 x_1 + w_2 x_2 + \cdots w_m x_m \tag{8}$$

The following introductions and equations come from the loss function based on mean squared error:

$$J(w) = \frac{1}{2}\sum_{l=1}^{n}\sum_{j=1}^{m}(\hat{y}(x_j^l) - y^l)^2 \tag{9}$$

Where:

x_j^l the j-th neuron in the l-th layer

n the total number of layers

Three basic gradient descent variants are introduced next.

2.3.1. Batch Gradient Descent (BGD)

As shown in Equation (10), BGD requires the calculation of partial derivatives for the whole data set according to Equation (7) and (9). To get the optimum w for achieving the maximum value of $J(w)$, we will repeat the following equation until convergence:

$$w_i \leftarrow w_i + \epsilon\sum_{l=1}^{n}\sum_{j=1}^{m}\left(y^l - \hat{y}(x_j^l)\right)x_{i,j}^l \tag{10}$$

Where $\leftarrow$ represents the assignment operation

ϵ represents the learning rate

i represents the number of iterations

However, it also has obvious disadvantages. In one update, it calculates the gradient for the whole data set, so the calculation time is very long. In this way, it will be very difficult to deal with a large scale data set, and update the model in real time with new data [28].

2.3.2. Stochastic Gradient Descent (SGD)

For a large data set, there may be similar samples, so the BGD method of calculating the gradient with all the data at one time will be redundant. While SGD will update each sample with a gradient every time, which is not redundant and relatively fast, and new samples can be added in. Its equation is shown in Equation (11). However, for accuracy, SGD only uses one sample to determine the direction of the gradient, resulting in a solution that is probably not optimal. However, due to the reason that SGD is updated frequently, it may lead to serious oscillation of cost function [29, 30].

$$w_i \leftarrow w_i + \epsilon\left(y^l - \hat{y}(x_j^l)\right)x_{i,j}^l \forall j = 1, \cdots, m \wedge \forall l = 1, \cdots, n \tag{11}$$

2.3.3. Mini-Batch Gradient Descent (MBGD)

The difference between MBGD and SGD is that MBGD used a small batch of sample data consists of n samples to calculate the loss function during each update. The value range of the hyperparameter n

is generally between 50 and 256. In this method, as shown in Equation (12), when updating the parameters, the variance will be reduced, the convergence will be more stable, and the calculation speed will be faster. However, if ϵ is extremely small, the convergence rate will be slow. Besides, if the loss is extremely large, the loss function will oscillate near the minimum value. In addition, the choice of batch size is also difficult [31, 32].

$$w_i \leftarrow w_i - \epsilon \sum_{l=t}^{t+n-1} \sum_{j=1}^{m} \left(\hat{y}\left(x_j^l\right) - y^l\right) x_{i,j}^l \tag{12}$$

2.3.4. Gradient vanishing and Gradient explosion

Gradient disappearance and gradient explosion are essentially the same, both of which are due to the continuous multiplication effect in gradient back propagation brought by the too deep network layers. In other words, if the numbers in each layer are smaller than 1, then the positive gradient that you get when you multiply them is very small, and that's the gradient disappearing. On the contrary, if the numbers in each layer are bigger than 1, then the forward gradient will be extremely large, that is, the gradient explosion. There are many ways to solve these two problems, such as switching to activation functions such as ReLU, LeakyReLU, ELU, utilizing batch normalization, ResNet residual structure, LSTM structure, pretraining plus finetuning, etc. [33, 34].

2.4 Propagation Thought

2.4.1. Feedforward propagation

Like the introduced in Section 2.1, The core idea of MCP is a linear relationship plus an activation function in a small local model, which is:

$$y = \sigma(z) = \sigma_{\mathrm{ReLU}}(b + \sum w * x). \tag{13}$$

Where * represents the convolution, b represents bias, σ represents activation function, usually refers to ReLU.

And since MCP is very simple which can only solve binary classification, the Multi-Layer perceptron (MLP) is introduced which is also called deep neural network (DNN). The major difference between DNN and MCP is the number of layers. With the core idea of MCP as the feedforward propagation of DNN, the general formula is:

Assume l-1-th layer has m neurons:

$$y_j^l = \sigma_{\mathrm{ReLU}}(z_j^l) = \sigma_{\mathrm{ReLU}}\left(\sum_{j=1}^{m} w_{jk}^l \, y_k^{l-1} + b_j^l\right) \tag{14}$$

where w_{jk}^l refers to weight between the j^{th} neurons in the lth layer and the k^{th} neurons in the layer before

It can be simplified that the output of l-th layer is:

$$y^l = \sigma_{\mathrm{ReLU}}(z^l) = \sigma_{\mathrm{ReLU}}(W^l y^{l-1} + b^l) \tag{15}$$

Where W^l refers to a n x m matrix formed by linear coefficient w of layer l

y^{l-1} refers to a m x 1 vector formed by output y of layer l-1

b^l refers to a n x 1 vector formed by bias b of layer l

z^l refers to a n x 1 vector formed by linear output before activating of layer l

However, DNN has a critical defect that if the input is large, a huge amount of weights and other parameters require being trained and is likely to lead to overfitting and local optimization. Therefore, the convolutional neural network (CNN) is discovered by LeCun, Boser [35] in 1989. It has two other kinds of layer, one is convolutional layer, the other is pooling layer. The detail information will introduce in

Section 3.

2.4.2. Feedback propagation

According to the Section 2.1, loss function aims to minimize the difference between the calculated output ($\hat{y}_j$) and the real output (y_j). And the difference between them is caused by the random weights and bias. In order to get suitable weights and bias to make the calculated output correctly, the feedback propagation is used. The main concept of feedback propagation is to optimize the loss function by gradient descent. We assume W and X represent the weights matrix and input vector in one layer separately.

Since the loss function is

$$J(W) = \frac{1}{2}\sum_{l=1}^{n}(\hat{y}^l - y)^2 \tag{16}$$

Where:

$\hat{y}^l$ refers to calculated output in the l-th layer

y refers to real output in the l-th layer

n refers to total number of layers

$$\hat{y}^l = \sigma_{\mathrm{ReLU}}(z^l) = \sigma_{\mathrm{ReLU}}(b^l + \sum W^l * X^{l-1}) \tag{17}$$

The loss function's partial derivative respect to W^l is

$$\frac{\partial J(W)}{\partial W^l} = [(\hat{y}^l - y) \odot \sigma'_{ReLU}(b^l + \sum W^l * X^{l-1})](\hat{y}^{l-1})^T \tag{18}$$

and its partial derivative respect to b^l is

$$\frac{\partial J(W)}{\partial b^l} = \delta^l[(\hat{y}^l - y) \odot \sigma'_{ReLU}(b^l + \sum W^l * X^{l-1})] \tag{19}$$

Where $\odot$ represents Hadamard Product, which is defined as the product of the corresponding elements of two matrices

With the two Partial derivative formula, we utilize gradient descent to get the new weight W^l and bias of l-th layer b^l [36]. The formulas are as following:

$$W^l \leftarrow W^l - \varepsilon\frac{\partial J(W)}{\partial W^l} \tag{20}$$

$$b^l \leftarrow b^l - \varepsilon\frac{\partial J(W)}{\partial b^l} \tag{21}$$

Where ε is learning rate

Assume the layer number of the last layer which is output layer is L. Then, the equation of gradient expression which is recorded as δ of the L layer is shown below:

$$\delta^L = \frac{\partial J(W,b)}{\partial z^L} = (\hat{y}^L - y) \odot \sigma'_{ReLU}(b^L + \sum W^L * X^{L-1}) \tag{22}$$

With the above formulas, the gradient expression of layer can be calculated by the fully connected layer (FC layer) after it. Assuming the parameters and gradient expression of l-th fully connected layer are known, the gradient expression of $(l-1)$-th layer can be got by this formula:

$$\delta^{l-1} = \frac{\partial J(W,b)}{\partial z^{l-1}} = (\frac{\partial z^l}{\partial z^{l-1}})^T\frac{\partial J(W,b)}{\partial z^{l-1}} = (\frac{\partial z^l}{\partial z^{l-1}})^T\delta^l \tag{23}$$

The relationship between z^l and z^{l-1} is

$$z^l = \hat{y}^{l-1} * W^l + b^l = \sigma_{ReLU}(z^{l-1}) * W^l + b^l \tag{24}$$

The above formula about the FC layer can be shown as following:

$$\delta^l = (\frac{\partial z^{l+1}}{\partial z^l})^T \frac{\partial J(W,b)}{\partial z^{l+1}} = (W^{l+1})^T \delta^{l+1} \odot \sigma'_{ReLU}(z^l) \tag{25}$$

When the l-th layer is convolutional layer, the gradient expression of $l-1$ layer which is hidden layer can be calculated by the formula shown below:

$$\delta^{l-1} = (\frac{\partial z^l}{\partial z^{l-1}})^T \delta^l = \delta^l * \text{rot180}(W^l) \odot \sigma'_{ReLU}(z^{l-1}) \tag{26}$$

The convolution kernel has been rotated 180 degrees. So rot180() means flip up and down once, then flip left and right once. Since CNN also includes the pooling layer which compresses data, the data should be restored.

When the l-th layer is pooling layer, the calculation of gradient expression of $(l-1)$-th layer is according to equation as:

$$\delta^{l-1} = upsample(\delta^l) \odot \sigma'_{ReLU}(z^{l-1}) \tag{27}$$

This up-sample function helps accomplishing the logic of pooling error matrix amplification and error redistribution [37].

In conclusion, with the basic operating mechanism about CNN, it is relatively easy to understand the structure of the CNN introduced in the following section.

3. BASIC STRUCTURE

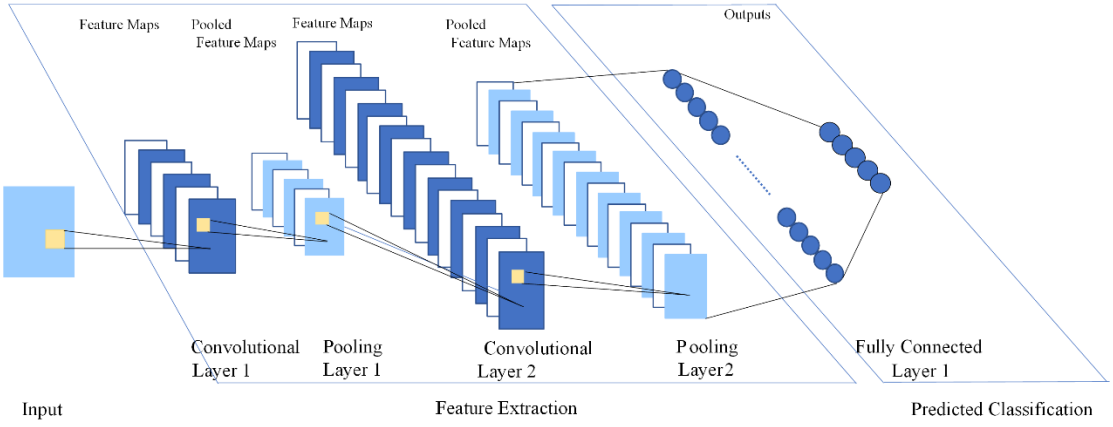


Figure 8 A sample structure of CNN

CNN can optimize the performance and generalization capability of the network by using multilayer structure [38, 39] illustrated in Figure 8. The most commonly used layers are introduced in the next.

3.1 Convolutional layer

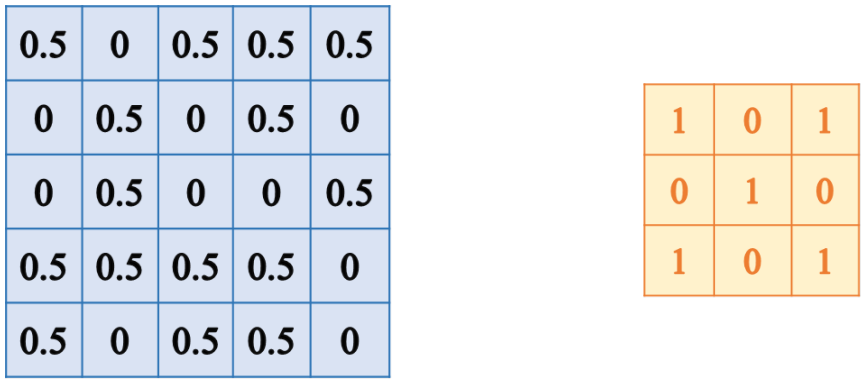
The main ability of the convolutional layer is to extract critical features for getting a good prediction from the input images, which means reducing images into forms that are easier to process [40].

3.1.1. Standard Convolution

The element utilized in conducting the convolution operation in a convolutional layer is named the

kernels whose parameters needs to be studied. The size of the kernel should be smaller than those of the input image. In the convolution process, each kernel will be convolved with the input image to compute an activation map. That is to say, the kernel will slide across the width and height of the input, calculating the dot product between the input and the kernel at each spatial position. Next, by superimposing the activation maps of each kernel along the depth dimension, the output image of the convolution layer can be achieved. It is because each kernel is smaller in width and height when compared to the input, each neuron in the activation map is associated only to a small local area of the input image, which means size of each neuron's receptive field is small, equals to size of kernel [41].

Figure 9 is considering a 5x5 input image in which matrix is composed of 0.5 and 0 and a 3x3 kernel composed of 1 and 0. The size of kernel determines the receptive field of the convolution. The most preference of selecting kernel size for a 2D input image is 3, which means 3x3 pixels.



(a) Input Image (b) 3×3 Kernel

Figure 9 Basic component of Standard Convolutional Layer.

Here the kernel will convolve over each cell position of the input image and eventually we will get a convoluted image. For every single point in it, you can take the point and the 3x3 points around it out and do the convolution. For instance, for the lower right (row 4, column 4) point, the local convolution of the 5x5 image and the 3x3 matrix could be computed as shown in Figure 10.

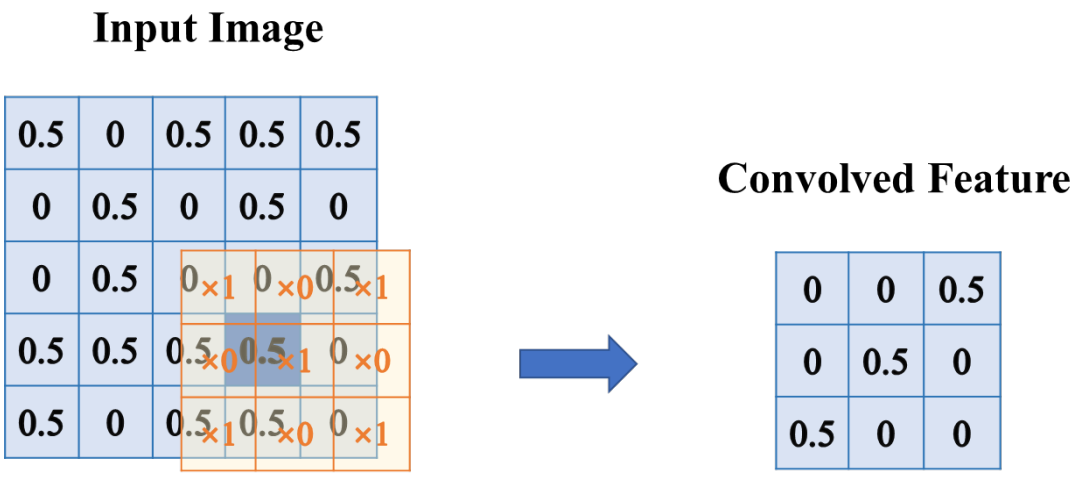


Figure 10 3×3 Convolution process

3.1.2. Dilated Convolution

One disadvantage of standard convolution is its design for up-sampling and pooling. During these processes, there will be internal data structure loss and spatial hierarchical information loss. To deal with this loss problem, one significant hyper-parameter called dilation rate is introduced to convolutional layers. The dilation rate of kernel refers to the spacing between values inside (for a standard convolution, the dilation rate equals to 1) [42]. Through the utilizing of dilated convolution, the receptive field of a 3x3 kernel with a dilation rate of 2 and no padding would as large as a 5x5 kernel as illustrated in Figure 11.

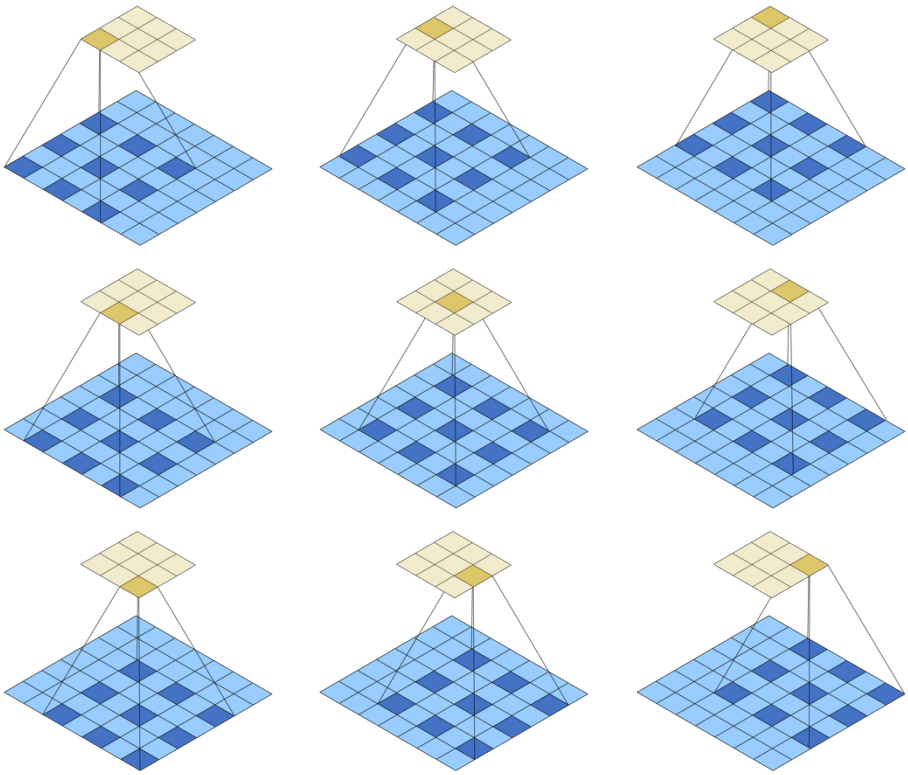


Figure 11 The process of convolution utilizing a 3x3 kernel with a dilation rate of 2

In this way, a wider reception field would be provided at the same cost of computation which makes dilated convolution a promising solution among the area of real-time segmentation. It can be of great use especially when there is a need of wider field of view but larger kernel size or multiple convolutions cannot be afforded [43].

3.1.3. Transposed Convolution

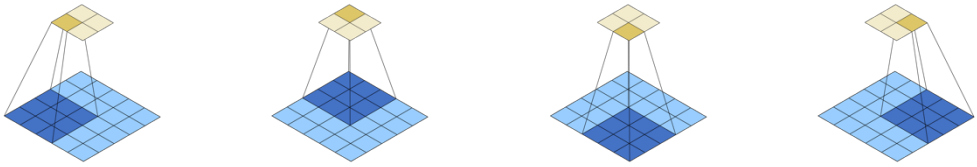


Figure 12 Part of the process of convolution utilizing a 3x3 kernel over a 5x5 input using unit-stride

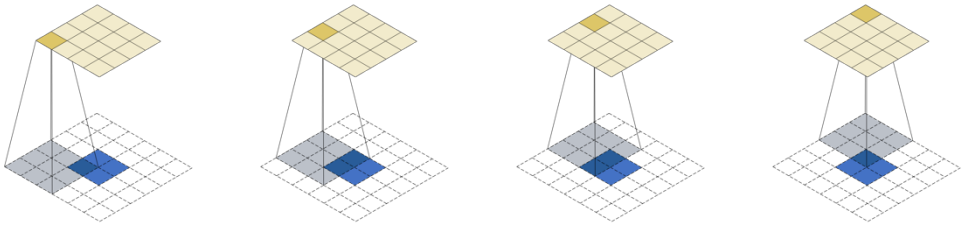


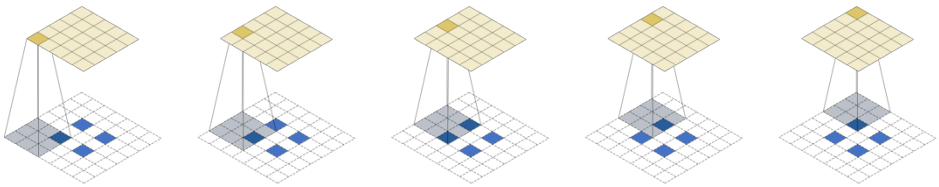
Figure 13 Part of the transpose process of convolution utilizing a 3x3 kernel over a 4x4 input utilizing unit-stride (same to convolution utilizing a 3x3 kernel over a 2x2 input with a zero padding of 2 with unit-stride)

For a conventional convolution layer, forward propagation is 'standard convolution', which maps the input feature map into the output feature map as shown in Figure 12, while reverse propagation is 'transposed convolution' [44, 45], which calculates the gradient of the input feature map according to the gradient of the output feature map and keeps the size of the gradient map same as that of the feature map illustrated in Figure 13.

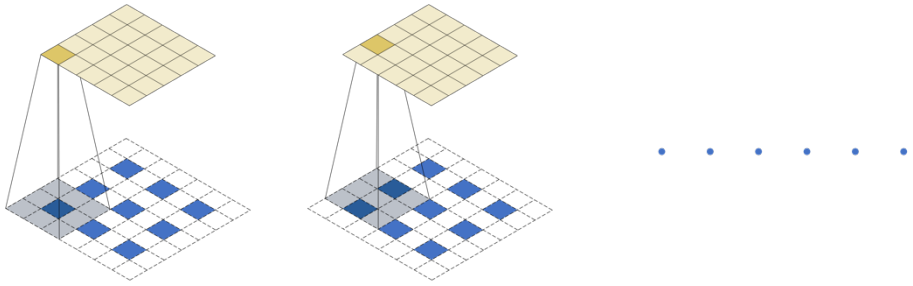
The process of transposed convolution is to some degree resemble to the deconvolution in that it produces the same spatial resolution as a deconvolutional layer. But the practical mathematical process carried out on the values is not the same. The transposed convolution guarantees that the output will be as large as before, while it is still carrying out a standard convolution process. To accomplish the goal, some fancy padding on the input is performed.

The transposed convolution step will not reverse the operation just mentioned, especially not concerning the numeric values. It only reconstructs the spatial resolution and carries out a convolution. This is very helpful for Encoder-Decoder architectures. Through utilizing this, upscaling of an image with a convolution can be achieved at one time, rather than two separate operations.

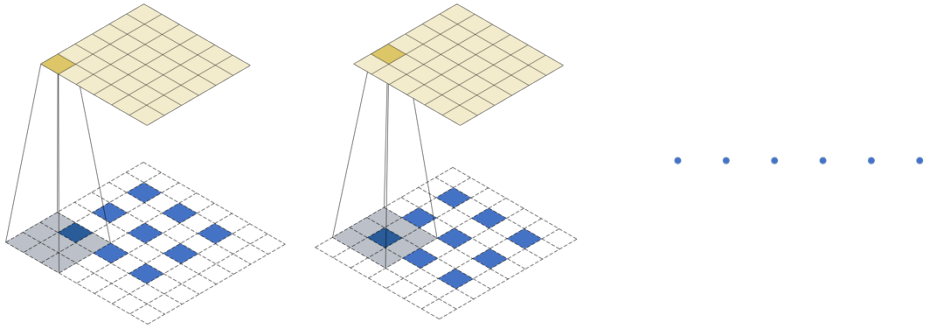
To conclude, if convolution is regarded as the mapping of the map of input size to the map of output size, transposed convolution is the mapping of the map of output size to the map of input size [46], which aims to restore the size. Some typical cases of transposed convolution are displayed in Figure 14.



(a) Part of the transpose of convolution utilizing a 3×3 kernel over a 5×5 input utilizing 2×2 strides (same to convolution utilizing a 3×3 kernel over a 2×2 input (gapped) with a zero padding of 2 utilizing unit-stride)



(b) Part of the transpose of convolution utilizing a 3×3 kernel over a 5×5 input with a zero padding of 1 utilizing 2×2 strides (same to convolution utilizing a 3×3 kernel over a 3×3 input (gapped) with a zero padding of 1 utilizing unit-stride)



(c) Part of the transpose of convolution utilizing a 3×3 kernel over a 6×6 input with a zero padding of 1 utilizing 2×2 strides (same to convolution utilizing a 3×3 kernel over a 3×3 input (gapped) with a zero padding of 1 utilizing unit-stride)

Figure 14 Typical cases of transposed convolution

3.1.4. Separable Convolution

In neural networks, separable convolution is commonly used in order to reduce the computation. The convolution process can be divided into two steps: one depth wise convolution and one point wise convolution [47]. In the depth wise convolution process, we use three convolution kernels of 5x5x1 to produce an 8x8x3 image as illustrated in Figure 15.

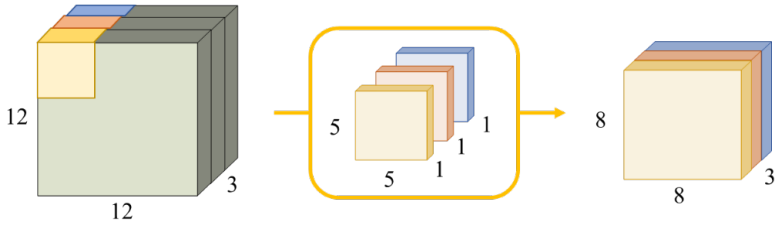


Figure 15 Depth wise convolution: transform a 12x12x3 image to an 8x8x3 image by using 3 kernels

In the pointwise convolution process, first, the convolution of 1x1x3 is used to check the calculation of each pixel, and the image of 8x8x1 is obtained. Then, 256 convolution kernels of 1x1x3 are used to produce the image of 8x8x256 as displayed in Figure 16.

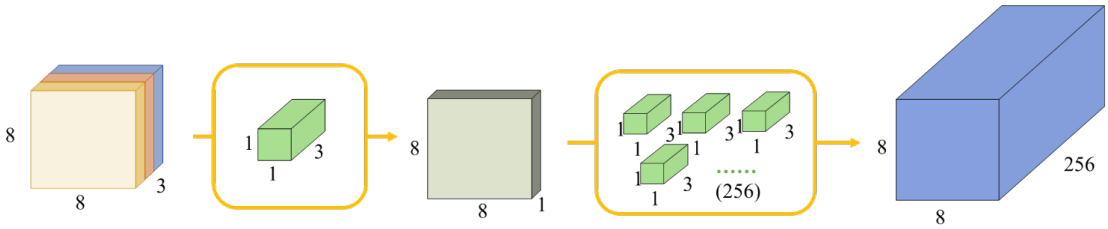


Figure 16 Pointwise convolution: transform an image from 3 channels to 1 channel, then output an image with 256 channels by using 256 kernels

Considering the change in computation, originally, 256 convolution kernels of 5x5x3 were moved 8x8 times, that is,

$$256x5x5x3x8x8=1,228,800 \quad (28)$$

times of multiplication.

During the utilization process of depth wise convolution, 3 convolution kernels of 5x5x1move 8x8 times, requiring

$$3x5x5x1x8x8 = 4,800 \quad (29)$$

times of multiplication.

During the utilization process of pointwise convolution, 256 convolution kernels of 1x1x3 move 8x8 times, requiring

$$256x1x1x3x8x8=49,152 \quad (30)$$

times of multiplication calculation, which adds up to 53,952 times of calculation Thus, the complete separable convolution process only needs 53,952 times of calculation which is much smaller than the computation of original process [48]. Separable convolution helps a lot in the application of mobile devices due to it uses the parameters efficiently [49].

3.2 Pooling layer

The function of pooling layer is reducing the resolution of feature maps. In other words, it aims to reduce the size of matrix generated by a convolutional layer. Typical categories of pooling will be introduced in the next [50].

3.2.1. Average Pooling

In a simple term, average pooling means taking the average of a small range. Figure 17 indicates an instance of average pooling utilizing a kernel of 2x2 with a stride of 2. After pooling, the average value in the 2x2 kernel will be calculated and reserved in a rectified feature map. In this way, the 4x4 input is compressed to 2x2.

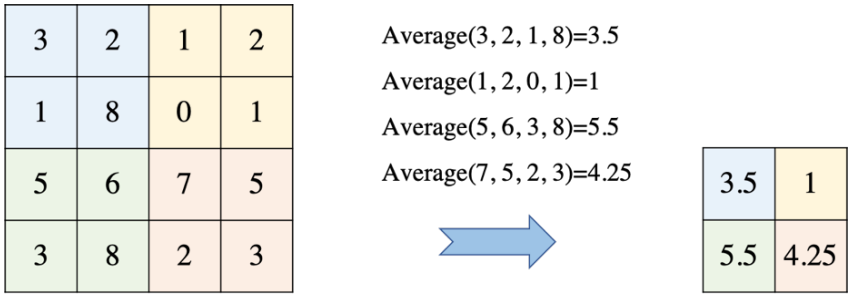


Figure 17 Average Pooling using a 2x2 kernel with a stride of 2

Error of feature extraction mainly caused by two sources. On the one hand, the increasing variance of the estimated value led from the limitation of neighborhood size. On the other hand, the parameter error from the convolutional layer will result in the deviation of the estimated mean value. Generally, average pooling to some degree solve the first problem by retaining more background information of the input image. Average pooling emphasizes the sub-layer sampling of the overall feature information, reducing the parameter dimension while ensuring the integrity of the information transferred. In a large model such as Dense Net, average pooling is frequently utilized to shrink the dimension and at the same time facilitate the transfer of information to the following module for extracting features [51].

3.2.2. Max Pooling

In a simple term, max pooling means taking the maximum of a small range. Figure 18 indicates an instance of max pooling utilizing a kernel of 2x2 with a stride of 2. After pooling, the maximum value in the 2x2 kernel will be left in a rectified feature map. Eventually, the 4x4 input image is shrunk to 2x2.

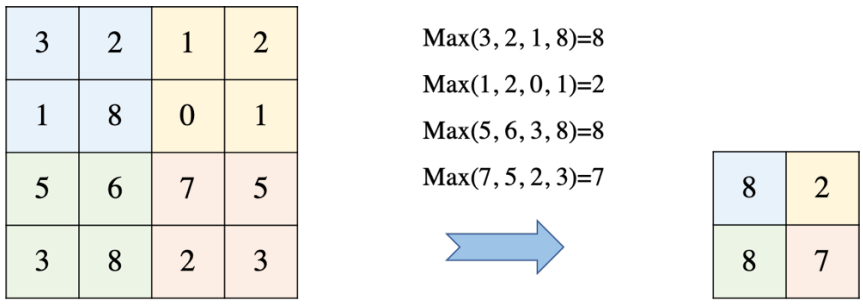


Figure 18 Max Pooling using a 2x2 kernel with a stride of 2

Generally speaking, max pooling is more effective and commonly used in image processing. As it feels more like doing a feature selection. It selects features with better classification and identification as well as providing nonlinearity. Besides, max pooling is able to deal with the second error mentioned in

the average pooling paragraph, which means it can preserve more texture information.

For instance, if the feature we need to find is a dog, then as long as there is a dog in one area of the image, it means that there is a dog in the whole image. Therefore, we should take the maximum matching degree between all areas in the image and the characteristics of the dog, that is, maximum pooling should be utilized.

3.2.3. Stochastic Pooling

In stochastic pooling, the pool mapping response is obtained by sampling the multinomial distribution formed by the activation of each pool area. To be more exact, we compute the probabilities p for each region j via normalizing the activations within the region using:

$$p_i = \frac{a_i}{\Sigma_{k \in R_j} a_k} \tag{31}$$

Where a refers to value of element

k refers to the index of element in region of feature map

We then sample from the multinomial distribution according to p to choose a location within the region. The procedure is in Figure 19.

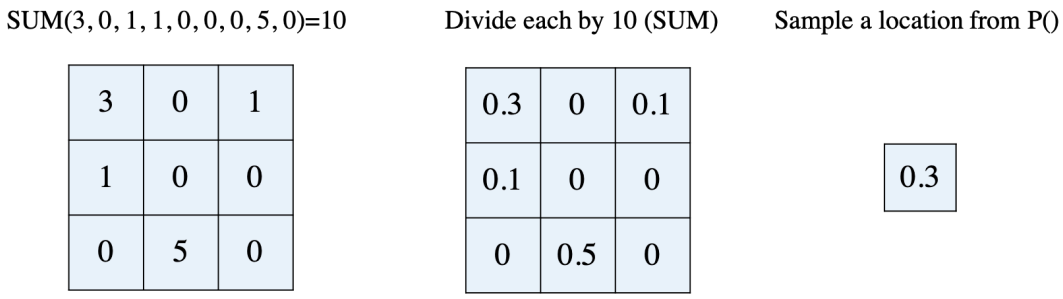


Figure 19 The transformation process of stochastic pooling

The samples for each pooling region in each layer for each training example are drawn independently to one another. When back propagation is carried out over the network, the gradient is guided back to the pool area using the same selected location, similar to using the maximum pool for back propagation. When passing information through the network, max pooling only focuses on the strongest activation of each region. While, sometimes non-maximal activations should also be taken into consideration. Stochastic pooling does well in this as those additional activations could be taken into account.

In a word, stochastic pooling is between the average pooling and max pooling. Probability is assigned to pixel points according to the value size, and then sub-sampling is conducted based on the probability. In the average meaning, it obeys average pooling criterion, and in the local meaning, it is similar to max pooling [52].

Although pooling operation has little effect on improving the overall accuracy, it plays an obvious role in decreasing parameters, reducing overfitting, optimizing model performance and saving computational force, so it is an essential process in neural network design [53].

3.3 Fully connected layer

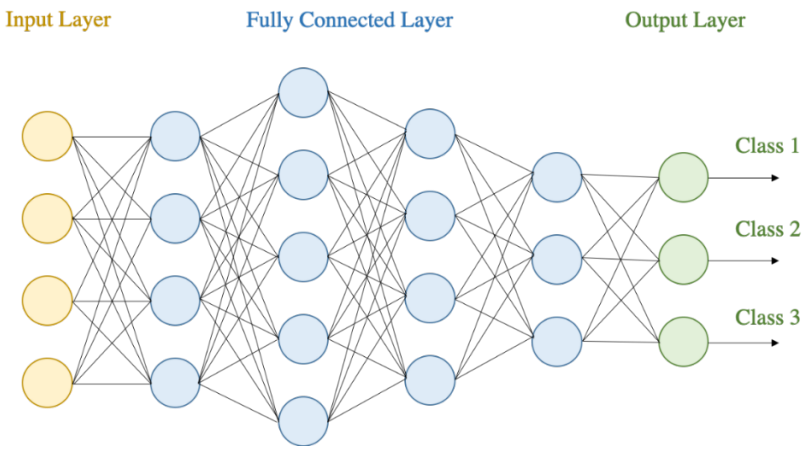


Figure 20 A sample structure of FC layer

Fully connected (FC) layers often appear near the end of CNN system and are utilized to sum the features of the previous design. If we consider convolution and pooling in the front as process of feature engineering, local amplification and extraction of local features. The FC layer in the back can be regarded as feature weighting. A sample structure of FC layer as displayed in Figure 20 is usually a fast method for learning nonlinear combinations of the high-level properties generated from convolutional layer. In that space, the FC layer will learn about a possible nonlinear function.

The basic procedure of learning is as follow. Firstly, flatten the image that has already been converted to a form suitable for a multilevel perceptron into a column vector and feeding it back to the feedforward neural network. Then, apply the propagation after planarization to each iteration of training. In this way, the model will have the ability to distinguish between the main features and some low-level features in the image and classify them through classification techniques such as SoftMax [54].

In some circumstances, FC layer can be realized by conducting convolution operation, that is, we can utilize a convolutional layer with 1x1 kernel size to replace the FC layer that is fully connected to the front layer. Within normal FC layer, the important feature weight is increased to obtain the category of the identified object. While in convolution operation, like an intentional weakening of fully connection, the weak influence outside the local area is directly erased to zero on the basis of the inspiration of local vision. There is also a bit of enforcement that the parameters utilized in different parts are the same. The weakening reduces the parameters, saves the calculation, and specializes in local areas rather than seek for more. While the enforcement can help further reduce the parameters.

After introducing the basic structure of CNN, the next section will focus on some classic CNN architectures.

4. CLASSIC ARCHITECTURE

In the early stage, because of the lack of training data and computational ability, it is impossible to train high performance CNN under no over-fitting circumstance. However, in the past few years, with the introduction of ReLU and dropout, along with the historical opportunity achieved by GPU and big data, the research on CNN has experienced several historic breakthroughs, began to obtain first-class results and attracted attention of the world. Figure 21 displays several classic architectures of CNN in its

development history [55].

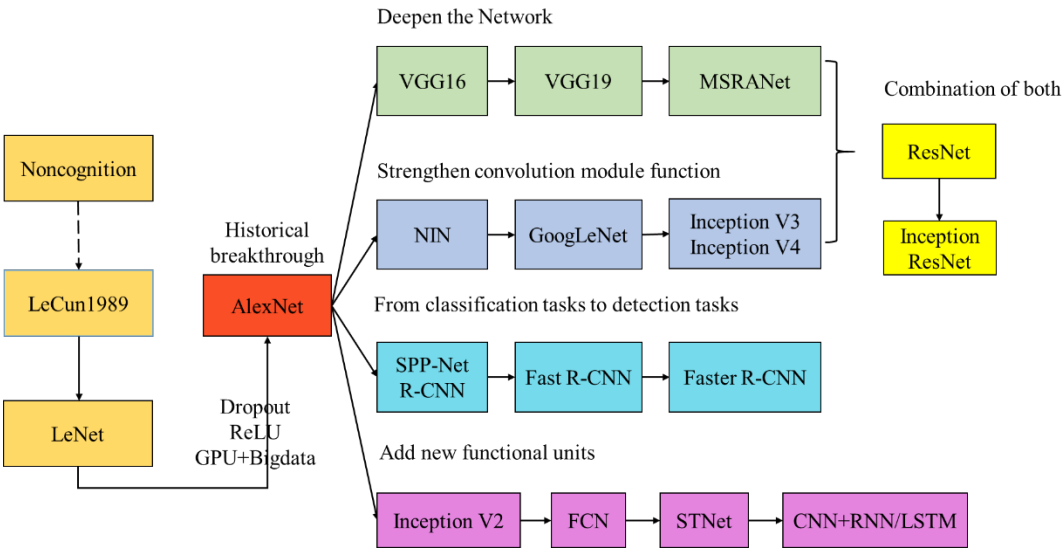


Figure 21 Classic CNN architecture in development history

4.1 AlexNet

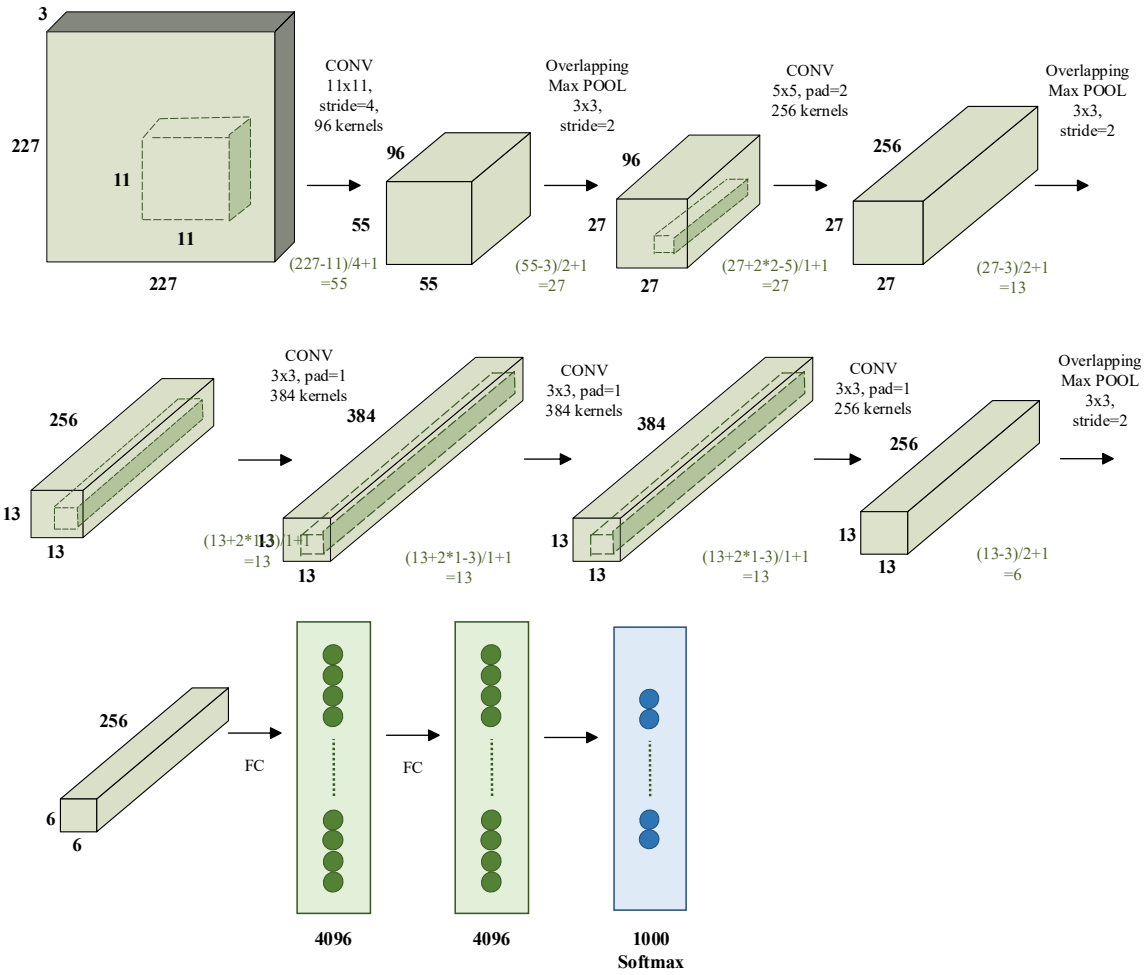


Figure 22 A sample structure for AlexNet

AlexNet came from a team led by Krizhevsky, Sutskever [16] in 2012 and its brief structure is illustrated in Figure 22. Due to the limited GPU capacity at that time, dual-GPU training was used, and the network was cut into two parts accordingly. However, with the development of GPU, single GPU can easily train AlexNet, so the two parts can be combined as shown in the figure. (Note that this leads to more connections and more computation) From this figure, it is obvious that the entire network structure of AlexNet consists of 5 convolutional layers and 3 FC layers, with a total depth of 8 layers [56, 57].

There is a clear process of direct conversion between layers and a simple interpretation of each layer is displayed in Figure 23. And then during a dropout process, where you randomly drop some information, you will get 4096 new neurons. The use of dropout can reduce overfitting and enhance the generalization of the model because it makes the model depend less on some local features. The dropout rate usually equals to 0.5.

CONV1	
Input image specification: 224x224x3 (RGB image), will actually be changed to 227x227x3 after preprocessing.	Image ↓ 11x11 conv, 96/ReLu ↓ Local Response Norm ↓ Max pool ↓
96 filters with size and specification of 11x11x3 are used for feature extraction. Use ReLU to ensure that the value of feature graph is within a reasonable range. Data after convolution is 55x55x96.	
Down sampling operation pool1 (Pool1 kernel: 3x3 step size: 2) Data is reduced to 27x27x96.	
CONV2	
Input data 27x27x96.	5x5 conv, 256/ReLu ↓ Local Response Norm ↓ Max pool ↓
256 5x5 filters with step size 1 were utilized to further extract features from the 27x27x96 feature graphs.	
Down sampling operation pool2 (Pool1 kernel: 3X3 step size: 2)	
CONV3	
Get 384 new feature maps of 13x13 (kernel 3x3, step size 1)	3x3 conv, 384 ↓
CONV4	
Same as CONV3.	3x3 conv, 384 ↓
CONV5	
Output data 13x13x256.	

Pool3 for down sampling operation (Pool3 kernel: 3x3 step size: 2) Data reduced to 6x6x256 [(13-3)/2+1=6]		3x3 conv, 256 ↓ Max pool ↓
FC6	A full connection was made with 4096 neurons to 256 feature maps of size 6 by 6.	fc 4096 ↓
FC7	Similar to FC6.	fc 4096 ↓
FC8	A thousand neurons were used to fully connect the 4,096 neurons in FC7, which then went through a gaussian filter to get 1,000 float values, which is what we consider as predictive probability.	fc 1000

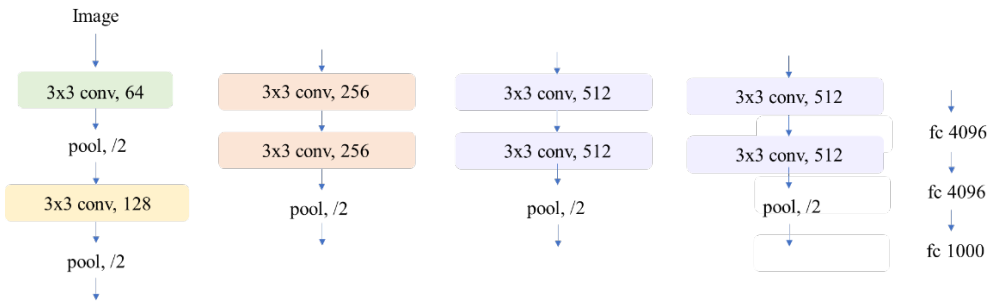
Figure 23 Interpretation of each layer in AlexNet

4.2 VGG-Net

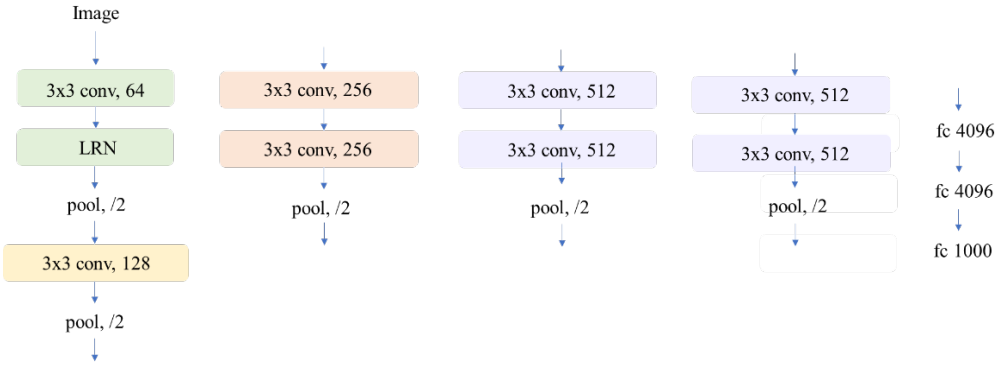
VGG-Net is a well-known deep CNN developed by Simonyan and Zisserman [58]. It writes about the relationship between the depth and performance of CNN [59]. Through repeatedly stacking the convolutional layer and the maximum pooling layer, VGG-Net creates a 16~19 layers CNN which achieves the runner-up in ILSVRC 2014 and the champion in positioning, with an error rate of 7.5% on the top5. So far, VGG-Net has still been a widely used network to extract image features.

The innovation of VGG-Net is it uses 3x3 convolution kernel and 2x2 pooling kernel to improve performance through deepening network structure. The growing number of network layers does not result in an explosion in the number of parameters due to the reason that the number of parameters is concentrated in the final three FC layers. Meanwhile, the series of two 3x3 convolutional layers can be regarded as one 5x5 convolutional layer, and the series of three 3x3 convolutional layers can be regarded as one 7x7 convolutional layer, that is, the size of the field of three 3x3 convolutional layers is equivalent to one 7x7 convolutional layer. But the number of convolutional parameters of 3x3 is only about half of 7x7, and the former can have 3 nonlinear operations while the latter only has 1 nonlinear operation, which makes the former have relatively stronger learning ability for features than the latter.

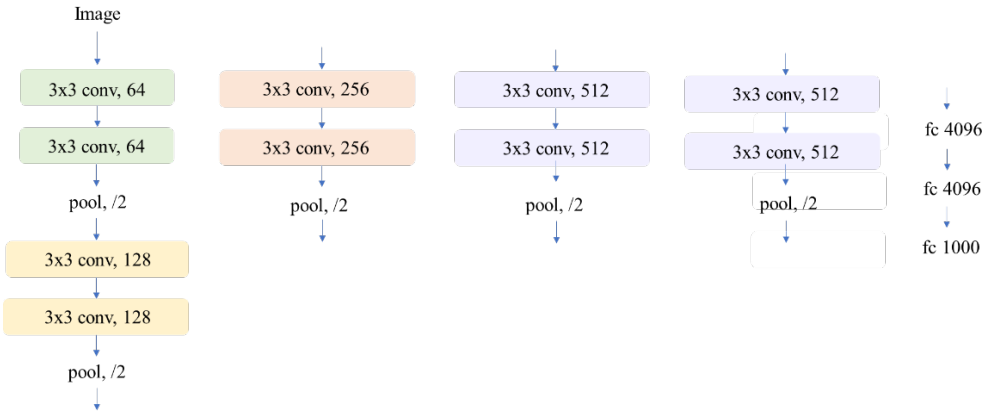
During training, VGG-Net first trains network of level A, and then initializes the later complex models by repeatedly using the weight of network A, in this way, the convergence speed improves. The configuration of VGG-Net is shown in Figure 24.



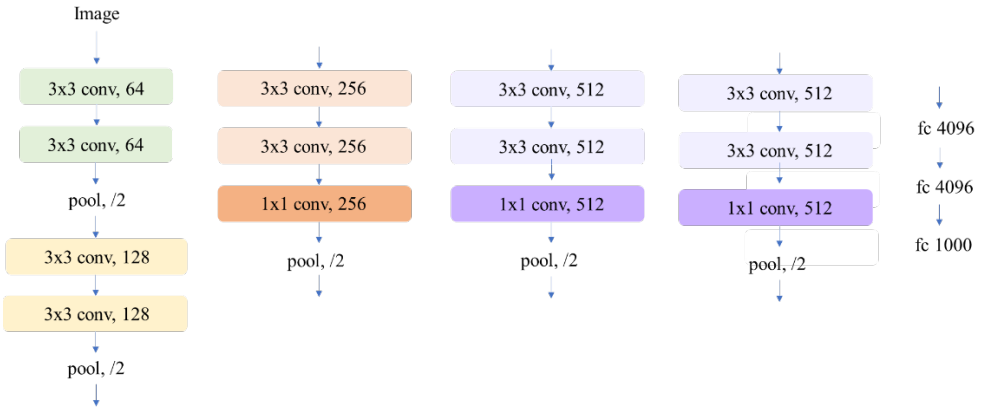
(a) A: 11 weight layers



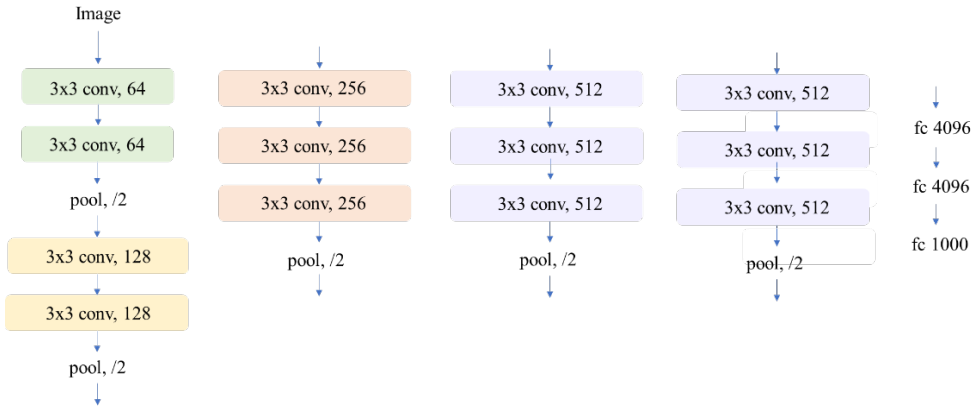
(b) A-LRN: 11 weight layers



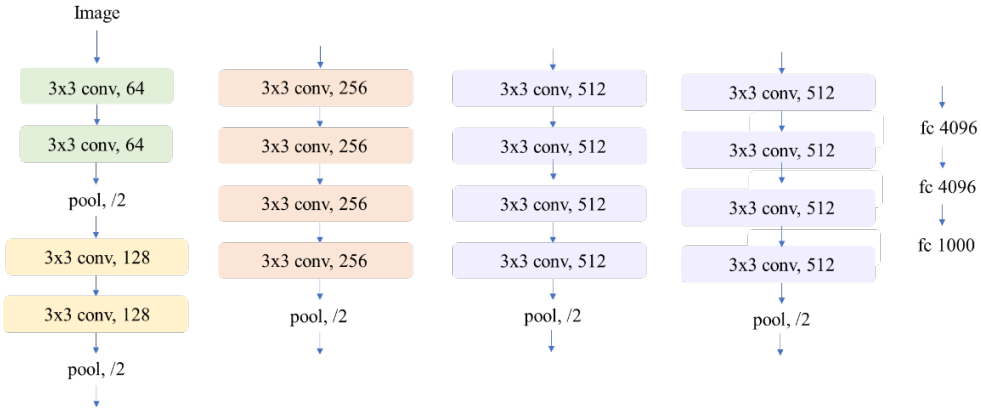
(c) B: 13 weight layers



(d) C: 16 weight layers



(e) D: 16 weight layers (VGG-16)



(f) E: 19 weight layers (VGG-19)

Figure 24 Configuration of VGG-Net

The authors of VGG-Net conclude that LRN layer has little effect. The experiment confirms that, when the network is going deeper, performance of network would be improved as well. The convolution of 1 by 1 is also efficient, allowing linear transformations of pixels without changing the number of output channels and the parameters of the neural network. But it doesn't work as well as the convolution of 3 by 3, because the convolution of 3 by 3 learns more about spatial features.

4.3 Inception (GoogLeNet)

Inception (also known as GoogLeNet) is a classic deep learning structure proposed by Szegedy, Ioffe [17]. Before this, many structures mentioned before all utilize adding the depth of the network to achieve better training performance, however, utilizing more layers will at the same time bring loads of negative effects, for instance, overfitting, gradient disappearance, gradient explosion, etc. Inception improves the training results by proposing a new way which is making more efficient use of computing resources, in other words, extracting more features for the same amount of computation [60, 61].

The basic structure is illustrated in Figure 25. The entire inception structure is made up of several such inception modules in series. The inception structure has two main contributions: one is using 1x1 convolution to raise and lower the dimension, the other is carrying out convolution and reintegration on multiple scales at the same time.

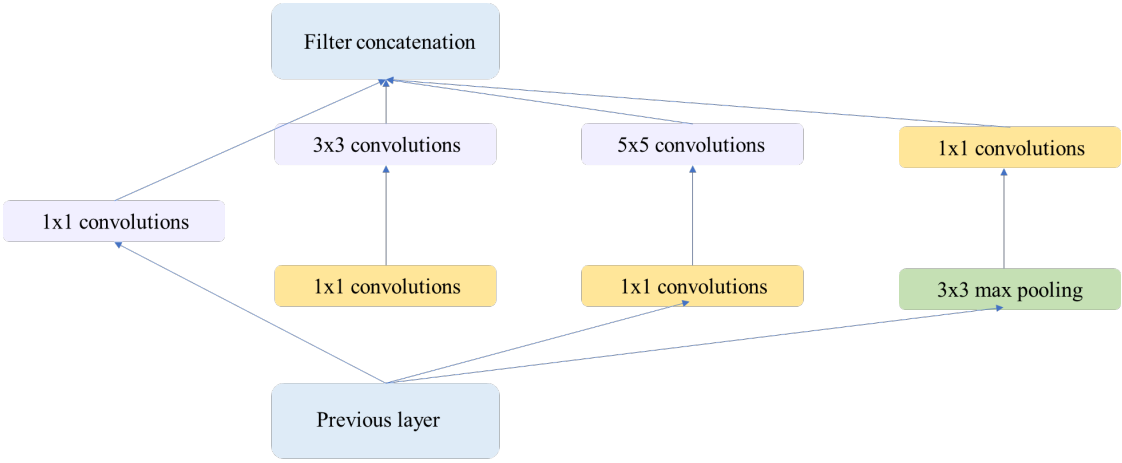


Figure 25 Basic structure of the inception module

There are multiple yellow 1x1 convolution blocks in the figure, which have two effects.

The first effect is they can help extracting richer features by adding more convolution to the same size of the feeling field. This idea comes from the Network in the Network (NIN) which indicates that if you put two convolutions in series, you can combine more nonlinear features. The structure of NIN is likely to the multi-layer structure in the traditional neural network. The latter multi-layer spans the receptive field of different sizes by adding pooling layer between layers, so as to extract features at a higher scale. NIN structure is multi-layer in the same scale (without pooling layer in the middle), so it can extract stronger nonlinearity in the same receptive field [62].

The second effect is 1x1 convolution can be used to reduce the dimension, which helps decreasing the computational complexity. More specifically, when a convolution layer has loads of input features, the convolution operation on the input will produce much computation. However, if shrinking the dimensionality of input first, and doing the convolution after the number of features are reduced, the computation will significantly be reduced as well. Figure 26 compares the multiplication times of the two schemes before and after optimization.

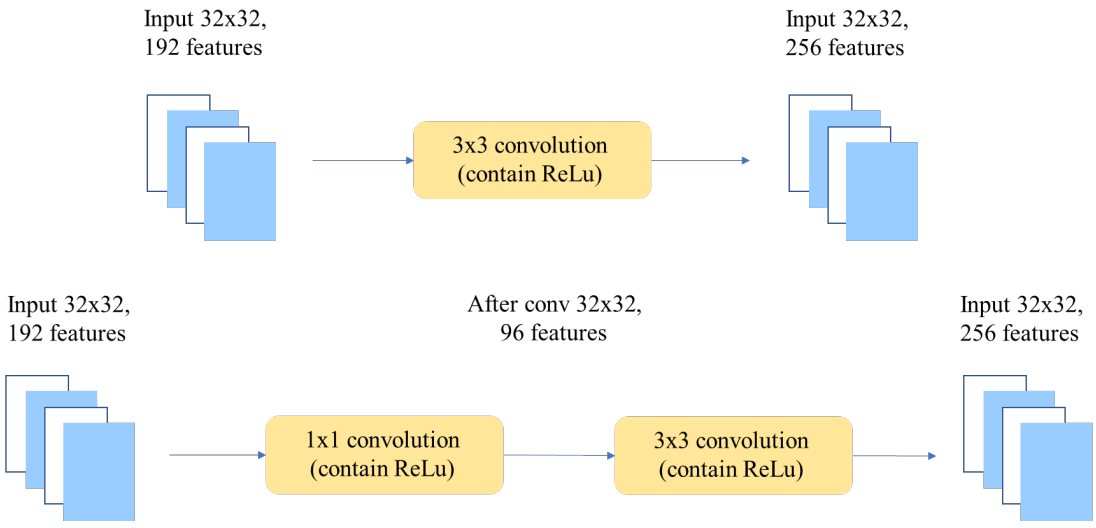


Figure 26 The comparison between the multiplication times of the two schemes before and after

optimization

With the same input of a set of 192 features, 32x32 size, and the output of 256 features, the first scheme is directly realized by 3x3 convolution, which requires

$$192\text{x}256\text{x}3\text{x}3\text{x}32\text{x}32=452984832 \text{ multiplications} \tag{32}$$

In the second scheme, the convolution of 1x1 was first utilized to decrease the number of features to 96, and then 3x3 was used to restore 256 features, which required

$$192\text{x}96\text{x}1\text{x}1\text{x}32\text{x}32+ 96\text{x}256\text{x}3\text{x}3\text{x}32\text{x}32 =245366784 \text{ multiplications} \tag{33}$$

The dimensionality reduction method of 1x1 was used to save half of the computation. Besides, if the number of features in the final output remains the same (256 sets), the dimensionality reduction in the middle is similar to the effect of compression and will not influence the result of the eventual training.

Apart from the 1x1 convolution blocks, the input layer has four branches, which were convolved or pooled with filters of different sizes, and finally integrated together on the feature dimension. Szegedy explained the benefits of this new structure from a number of perspectives.

The first explanation is the features of different scales can be extracted by convolution on multiple scales at the same time. The richer the features, the more accurate the final classification.

The second explanation is utilizing the principle of decomposition of sparse matrix into dense matrix to accelerate the convergence. For instance, the first matrix in Equation (34) is a sparse matrix (many elements are 0 and are unevenly distributed in the matrix), which is convolved with a 2x2 matrix and needs to be calculated for each element in the sparse matrix. However, if the sparse matrix is decomposed into 2 sub dense matrices and then convolved with the 2x2 matrix, the regions with more 0 in the sparse matrix can be eliminated from calculation, greatly reducing the amount of calculation. Applying this principle to Inception needs decomposing on the feature dimension. The input data of the traditional convolution layer convolved with the convolution kernel of one scale (such as 3x3) and output fixed dimensions data (such as 256 features). All 256 output features were uniformly distributed on the scale of 3x3, which can be understood as the output of a sparsely distributed feature set. While Inception module extract features on multiple scales (such as 1x1, 3x3, 5x5), thus the output features of 256 is not evenly distributed, but the features that have a strong correlation gather together (such as 96 1x1 features together, 96 3x3 features together and 64 5x5 features together). This can be understood as multiple densely distributed sub-feature sets. In this feature set, as the more relevant features cluster together, the non-relevant non-key features weaken, so that the Inception method outputs the same 256 features with less ‘redundant’ information. With such ‘pure’ feature set transferred layer by layer and eventually used as input of reverse calculation, the natural convergence rate becomes faster.

$$\begin{bmatrix} 3 & 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \Leftrightarrow \frac{\begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix} \otimes \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}}{\begin{bmatrix} 2 & 3 & 0 \\ 0 & 1 & 4 \\ 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}} \tag{34}$$

Where $\otimes$ represents Kronecker Product which is used for operations between two matrices of any size.

$\Leftrightarrow$ represents Substantively equivalent.

The third explanation is according to Hebbin's principle [63]. The Hebbin’s principle is a theory in neuroscience that explains what happens to neurons in the brain during the learning process: fire together, wire together. Hebbin believes that ‘Two neurons, or systems of neurons, that are always firing at the same time form a 'combination' in which the firing of one neuron promotes the firing of the other’. For

example, dogs drool when they see meat. After repeated stimulation, the neurons in the brain that recognize meat and those responsible for salivating will promote each other and 'entwine' with each other, making them drool faster when they see meat in the future. The usage of this principle in Inception structures is to bring together highly relevant features. It's a little bit like the second explanation above, where you separate the features of 1x1, 3x3, and 5x5. Since the ultimate goal of training convergence is to extract independent features, the convergence speed can be accelerated as long as the highly relevant features be gathered in advance.

4.4 ResNet

ResNet was proposed by He, Zhang [64] from Microsoft research institute in Dec, 2015. Through utilizing residual Unit, ResNet successfully trained the 152-layer neural network, and achieved the best result in ILSVRC 2015. Meanwhile, the number of parameters was lower than VGG-Net, with outstanding effect. The structure of ResNet is able to greatly speed up the training of ultra-DNN, as well as perfect the model accuracy.

The initial inspiration of ResNet comes from this problem: if depth of neural network is continuously increasing, a degradation problem will occur. This means the accuracy will first rise and then reach saturation, afterwards the continuous increase of depth will cause the decline of accuracy. The difference between this problem and overfitting is, in this problem, the error increases in both testing set and training set.

If a relatively shallow network reaches saturated accuracy, then adding several $y = x$ congruent mapping layers to the end will at least not increase the error, which means, the deeper network will not cause the increasing of error on training set. The idea of utilizing congruent maps to transfer the previous layer of output directly to next layer was the inspiration for ResNet. If we directly pass the input to the output as the beginning value, the target we require to study at this point will be:

$$F(x) = H(x) - x \tag{35}$$

Where x refers to input of neural network

$H(x)$ refers to expected output

The Residual Unit of ResNet is illustrated in Figure 27. ResNet altered the learning objective from learning a complete output to learning only the difference between output and input, thus simplifying the learning objectives and difficulty.

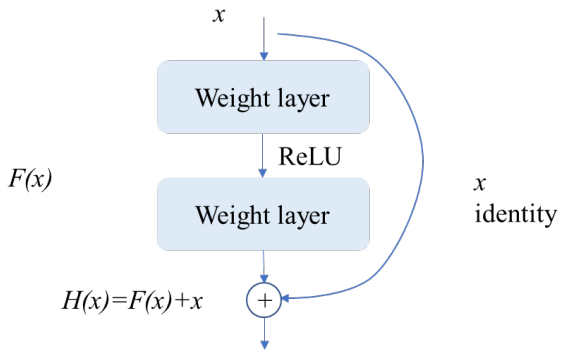
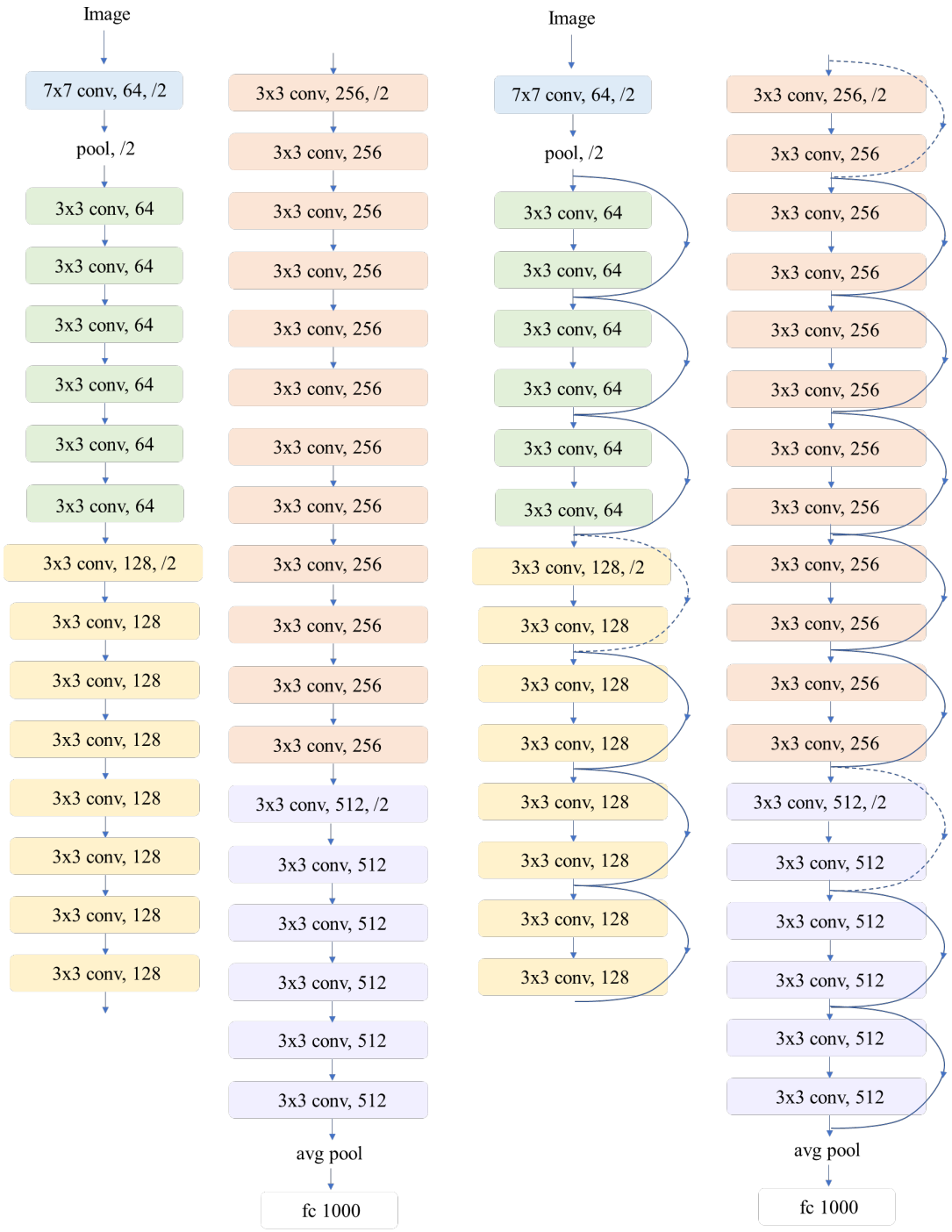


Figure 27 A sample structure of residual block

The Residual block is realized by shortcut connection as displayed in Figure 27. Through utilizing

shortcut, this block performs an element-wise superposition of the input and output. This simple addition will not increase additional parameters and computation to network. However, it could efficiently improve the training speed and training effect of the model. In addition, if the layer number of model is increasing, this structure is also able to efficiently handle in the problem of degradation.

Figure 28 shows a comparison between a 34-layer ordinary CNN and a 34-layer ResNet network. It can be seen clearly that the biggest difference between a straight-through CNN and ResNet is that ResNet has a large amount of bypass branches which transmit the input straightway to the subsequent layer, allowing the subsequent layer to directly study the residual. This category of structure is as well-known as shortcut or skip connection.



(a) (b)

Figure 28 Comparison between a 34-layer ordinary CNN and a 34-layer ResNet network.

4.5 U-Net

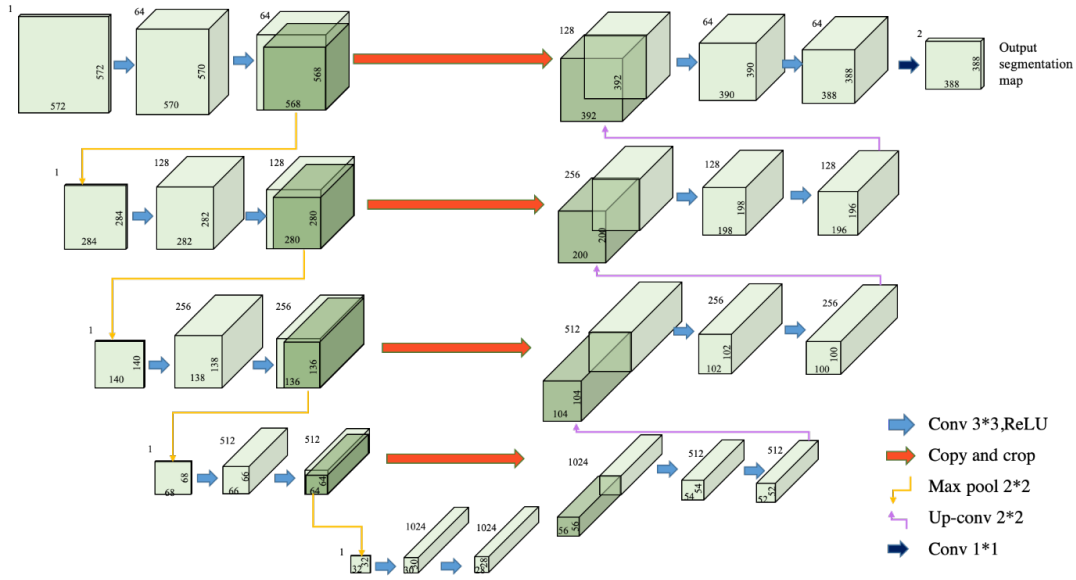


Figure 29 The basic structure of U-Net

U-Net is a classical full convolutional network. This U-shaped structure can capture semantic information in the process of down-sampling and accurately locate the corresponding up-sampling process. This network can train end-to-end with a small number of images, but the effect exceeds that of the previous best model, CNN based on sliding Windows [65]. The U-shaped structure is shown in Figure 29.

Assuming that the input image tile is of 572x572. The left side of the network composed of four blocks is called contracting path, which is a series of reduction sampling operations formed by convolution and max pooling. Finally, a feature map with a size of 32x32 is obtained.

Network on the right-hand side is named expansive path. It is as well composed of four blocks. Before the start of each block, it will use deconvolution to enlarge two times the size of the feature map, and the number in half. Then, it is merged with the feature map of the symmetric contracting path on the left due to the reason that the size of the feature map of the contracting path on the left is different from that of the expansive path on the right. The normalization of U-Net is conducted by clipping feature map of the contracting path to the same size as feature map of the expansive path (i.e., the dark green block in Figure 29). Effective convolution operation is still used in convolution operation of extension path, and the final size of feature map is 388x388.

U-Net mainly uses SoftMax cross entropy loss to deal with the problem that similar target boundaries of medical images are not easy to be recognized. Besides, weights are added to each pixel in the objective function to make the network more able to distinguish the boundaries. The equations are as follows:

$$E = \sum_{x\in\Omega} w(x)\log(p_{l(x)}(x)) \tag{36}$$

$$w(x) = w_c(x) + w_0 \cdot \exp\left(-\frac{(d_1(x)+d_2(x))^2}{2\sigma^2}\right) \tag{37}$$

Where $p_{l(x)}(x)$ is loss function of SoftMax

l refers to the label value of a pixel

w refers to the weight value of a pixel

w_c is the weight that balances the proportion of the category

d_1 is the distance from the pixel to the nearest cell

d_2 is the distance from the pixel to the second closest cell.

w_0 and σ are constants.

4.6 R-CNN Series

4.6.1. R-CNN

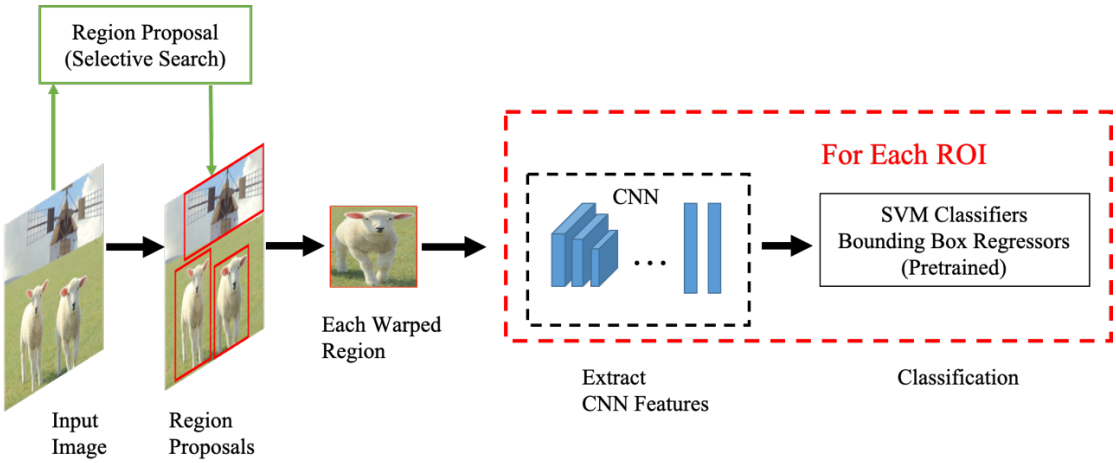


Figure 30 The basic structure of R-CNN

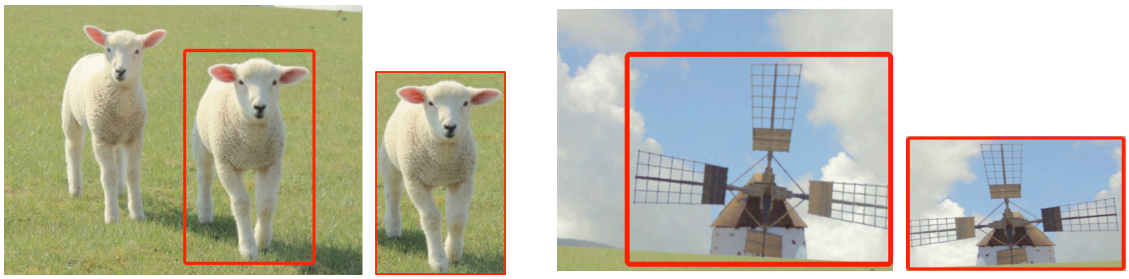
The general procedure for target detection of R-CNN is illustrated in Figure 30: (i) Extract regional proposals from input images using selective search algorithm and warp them to a fixed size. Selective search synthesizes the methods of both continues search and segmentation. It aims at identifying possible target locations for object recognition. (ii) Input the normalized regional proposals into the CNN network and extract features. (iii) SVM classification is used for identification of these features, and linear regression is applied for fine-tuning the border position and size, with each category separately training a bounding box regression. By using less and higher quality region of interest (ROI), R-CNN performs better than the system using traditional sliding window approach in the evaluation of accuracy and speed [66].

However R-CNN framework still exist the following problems: (i) it is very time-consuming and has lots of repeated computation as each candidate box needs to go through CNN alone; (ii) its training is divided into several stages and the steps are tedious; (iii) it takes up disk space: the convolved characteristics of the data needs to be kept separately [67].

4.6.2. SPP-Net

In order to solve the problems of R-CNN, SPP-Net made the following improvements.

One of the main ideas of SPP-Net is to remove the operations such as crop/ warp on the original image and replace them with the spatial pyramid pooled layer (SPP) on the convolution feature. In R-CNN, the proposal was warped before it was sent to the net. Part of the reason for the introduction of SPP layer is that different levels of crop/ warp on images in R-CNN will not only take time and effort but also cause some problems as shown in Figure 31. For example, the crop in the figure will cause the object to be incomplete, and warp will cause serious deformation after the object is stretched [68].



(a) (b)

Figure 31 (a) Crop; (b) Warp

However, the more important reason is that in R-CNN, each regional proposal is subject to a CNN feature extraction operation, which brings a lot of double calculations. In SPP-Net, we only performed CNN feature extraction on the whole image once, and then map the proposal to the feature map generated by the convolutional network, which is equivalent to feature map obtained by image in a proposal going through the layer by layer convolution process.

4.6.3. Fast R-CNN

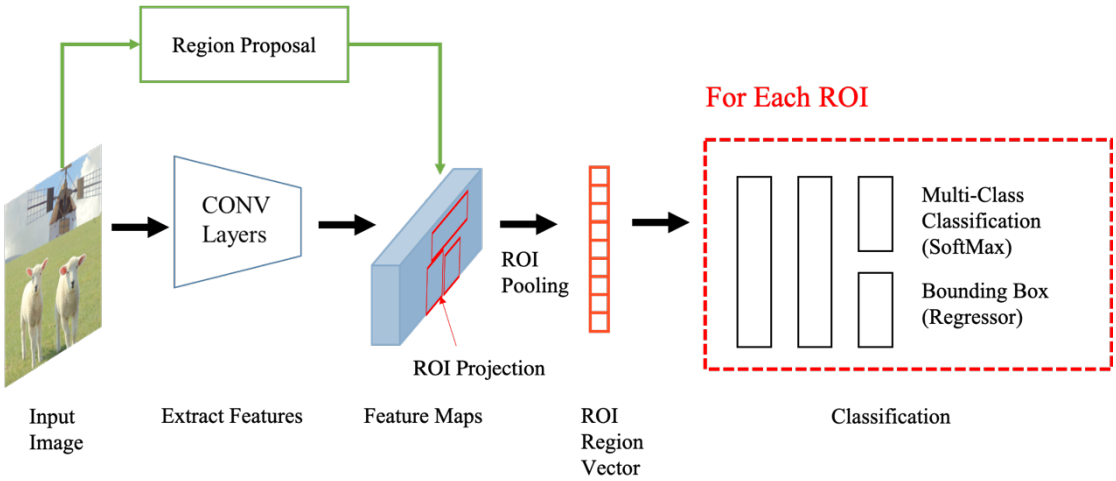


Figure 32 The basic structure of fast R-CNN

It could be obviously seen in Figure 32 that, fast R-CNN is also a method that makes some perfections based on R-CNN. Similar to SPP-Net, it only conducts CNN feature extraction for the whole image once, and then utilizes an ROI pooling layer. While, due to the reason that it is not a fixed size input, the size of the pooling grid shall be adjusted dynamically each time to achieve regional normalization. Fast R-CNN uses SoftMax to replace SVM for classification, as well as multi-task loss

function to carry out border regression and classification together [69].

The main steps are as follows: (i) Take the whole picture as the input and use CNN to get the feature layer of the image. (ii) Extract regional candidate boxes from the original image by selective search and projected onto the final feature layer one by one. (iii) ROI pooling operation is conducted for each area candidate box on the feature layer to achieve feature representation of fixed size. ROI pooling is able to accelerate the speed of training and testing. (iv) After two FC layers, utilize SoftMax to conduct multi-target classification respectively, and apply regression model in fine-tuning the border position and size.

The disadvantage of fast R-CNN is that region proposal is extracted by selective search, and most of the target detection time is spent on it (normally, if a region proposal is proposed 2~3s once, the feature classification will be proposed only 0.32s once), which could not reach the requirements of real-time application.

In general, fast R-CNN uses feature CNN as extractors to extract features of the whole image first. And then it applies the method of creating candidate regions directly to the extracted feature map. For example, fast R-CNN chose feature map output by convolutive layer Conv5 in VGG-16 to generate ROI. These ROI would then be combined with corresponding feature graphs to cut out feature blocks and used in target detection tasks. We use ROI pooling to convert feature blocks to fixed sizes and feed them to the FC layer for classification and positioning. Because fast R-CNN has no repeated feature extraction operation, it saves lots of processing time. The training speed of fast R-CNN is 10 times faster than R-CNN, and the predicting speed is 150 times that of R-CNN.

4.6.4. Faster R-CNN

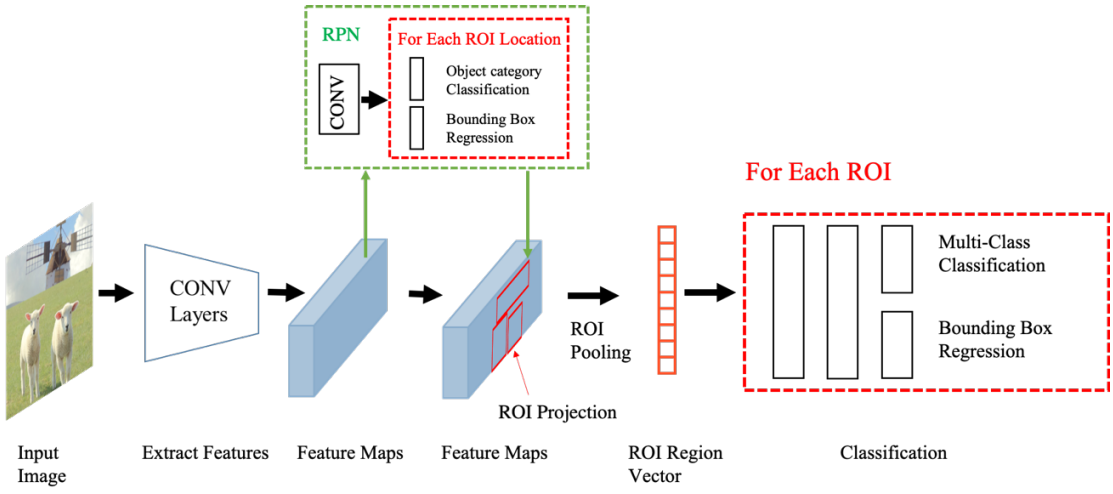


Figure 33 The basic structure of faster R-CNN

As displayed in Figure 33, faster R-CNN utilizes the proposal achieved by Region Proposal Network (RPN) to replace the proposal achieved by selective search.

The function of RPN network is outputting a batch of rectangular candidate regions with the input of an image, which is somewhat alike to the selective search in the previous target detection. The network structure is according to a CNN, but includes the SoftMax and Bounding box regression multi-task models [70].

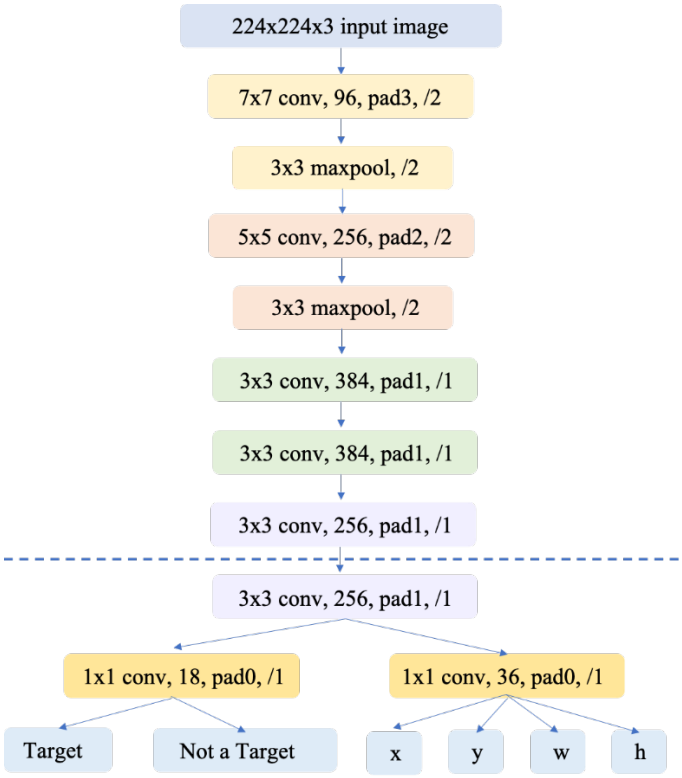


Figure 34 The basic structure of RPN Network

In Figure 34, above the dashed line is the structure in front of the last convolutional layer of ZF network. Under the dashed line is the particular structure of RPN network. For the particular structure, the convolution is 3*3, and then the output is divided into two ways through 1*1 convolution, one way of which outputs the probability of target and non-target, and the other way outputs the four parameters related to box.

The procedure of faster R-CNN is: (i) Same as fast R-CNN. (ii) K different Anchor boxes are used in the final convolution feature layer to conduct a region proposal, that is, object/non-object binary classification is carried out for the corresponding area of each anchor box, and k regression models are used for fine-tuning the location and size of the candidate box. Regional proposal (via RPN) is generated on the feature map, with about 300 suggestion windows per image. iii) Same as fast R-CNN. iv) Classify the objects and do border regression.

4.7 YOLO

YOLO series algorithm is a typical one-stage target detection algorithm. Its basic structure is illustrated in Figure 35.

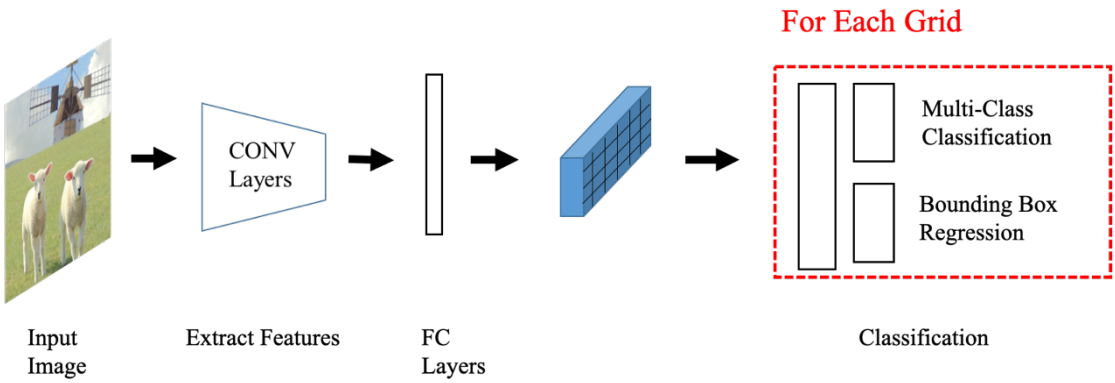


Figure 35 The basic structure of YOLO

2.3.6.1 YOLOv1

YOLOv1 is an end-to-end target detection approach, and the determination of position and category is accomplished by direct regression without need for intermediate regional proposal to find the target. If two small targets fall into a cell at the same time, the model can only predict one. Large object intersection over union (IOU) error and small object IOU error are close to the loss contribution value in network training. Thus, for small object, small IOU error will also bring a great influence on the network optimization, reducing positioning accuracy of object detection. YOLO uses multiple lower sampling layers, and the object features obtained from network are not precise, which will also affect the detection effect [71].

Compared with faster R-CNN, YOLOv1 has more obvious advantages. Firstly, faster R-CNN needs an RPN network to find the candidate region, while for YOLOv1, after the input image has been divided, it directly takes the region as the candidate region. YOLOv1 utilizes the end-to-end regression method rather than use those regional proposal steps, so direct regression accomplishes the determination of position and category. Therefore, compared with fast R-CNN, the main advantage of YOLOv1 lies in (i) its speed; (ii) global processing makes background errors less than regional based methods; (iii) good generalization performance. However, YOLOv1 performs not very well in target positioning process, which causes low detection accuracy of YOLO.

To sum up, YOLOv1 proposes a new idea of a single stage. Compared with the two-stage method, YOLOv1 has obvious advantages in speed and impressive real-time characteristics. However, YOLOv1 also has some problems, such as rough grid division and the number of boxes that could be generated by each grid, which limits the detection of small-scale objects and objects that are close to each other.

2.3.6.2 YOLOv2

The differences between YOLOv2 and YOLOv1 are as follows:

1) The dropout layer was removed, and Batch Normalization was added in each convolutive layer, with a 2% improvement in Mean Average Precision (mAP).

2) High-resolution classifier. Input data of high-resolution changed from 224*224 to 448*448, and mAP improved by 4%.

3) YOLOv2 deletes a pooling layer to get high resolution data. Use anchor to replace FC layer.

4) Anchor box is determined by k-mean clustering and distance calculation of clustering is used

instead of Euclidean distance.

5) YOLOv2 uses anchor box to predict the object category rather than use grid to predict the probability of each grid in YOLOv1.

6) The bounding Box coordinate continues YOLOv1's prediction of Bounding boxes' coordinates with respect to the upper left corner of the grid. To ensure that the center of bounding Box falls within the grid (i.e. its coordinates fall between 0 and 1), Bounding Box center is sent into the activation function.

7) YOLOv2 adds a passthrough layer to improve the detection of small objects. This layer conducts information fusion of high-resolution feature and low-resolution feature. Through the stacked in adjacent features rather than the spatial location of different channel, it integrates characteristic of high and low resolution, making the mAP value increased by 1%.

8) YOLOv2 only uses convolutional layer and pooling layer, thus the size of input image could be dynamically adjusted. Besides, the accuracy and speed can be adjusted according to own requirements.

4.8 SSD

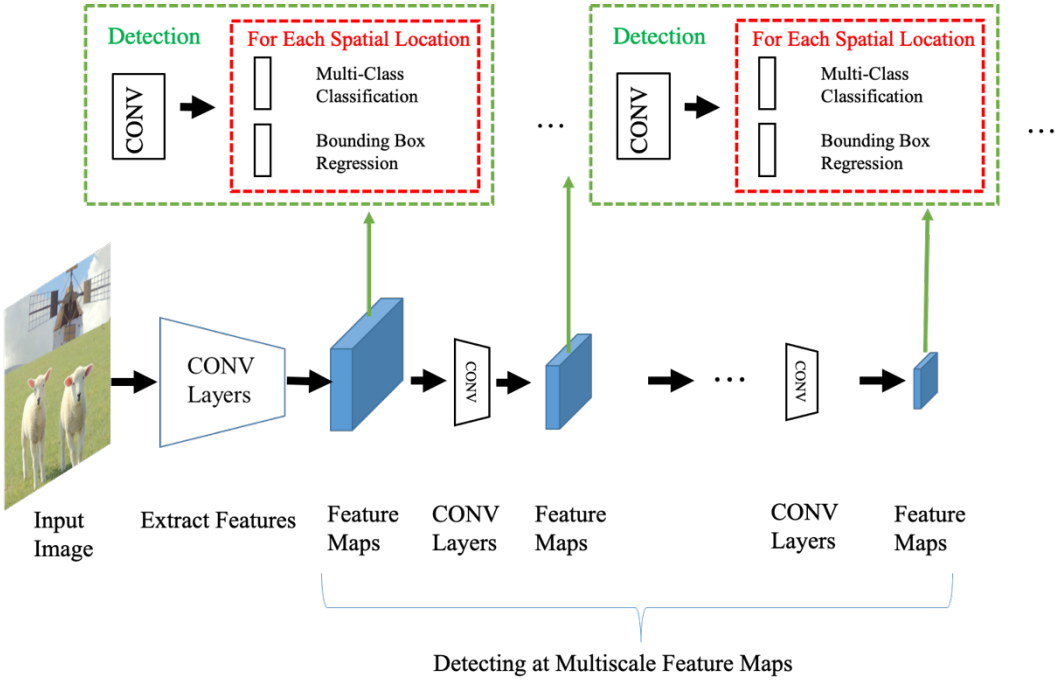


Figure 36 The basic structure of SSD

YOLO predicts a location utilizing full-image features, SSD predicts a location utilizing features around that location [72].

According to Figure 36, it seems that SSD model is somewhat alike to RPN, for instance, the Default Bounding Box in SSD is very likely to the anchor in RPN. However, SSD takes into account different scales in different feature layers, and RPN thinks about different scales in one feature layer.

SSD outputs a series of discretization boxes, produced on feature maps on different layers, with different aspect ratio.

This section introduced the classic architectures in CNN from an early stage in nearly 10 years ago till now. The CNN model has been continuously evolved and eliminated in the careful study of countless scientists, thus forming various model systems with their own merits. The following section will focus

on how different model made remarkable achievements and outstanding contributions in different medical fields.

5. APPLICATION

Since CNN have some many models and it can be used in lots of fields of image classification like identifying human face and patient condition. The following content will introduce the application of CNN for classifying the patient conditions in different body parts.

5.1. Breast

According to the statistical analysis defined from Institute for Health Metrics and Evaluation (IHME), there was a noticeable increase in breast cancer mortality in most regions of the world between 1990 and 2015, with a total mortality rate of 0.7 per 100,000. Data also indicates that the five-year survival rate of early stage breast cancer patients exceeds 95%, while the five-year survival rate of late period breast cancer patients is less than 20%. The World Health Organization (WHO) has confirmed that early diagnosis and treatment of breast cancer patients could not only decrease mortality, but also help some patients avoid psychological suffering and financial loss from mastectomy and axillary lymph node dissection [73-75]. Thus, early detection is a critical step in prevention and treatment of breast cancer.

Mammography is a specific imaging method to screen and diagnose breast cancer. Lévy and Jain [76] and his group used CNN to straightway classify pre-detected breast masses in mammograms as benign or malignant. They utilized AlexNet, GoogleNet model and achieve outstanding precision rate result as shown in Table 1. Another project focuses on applying CNN to detect microcalcifications in mammograms, since CNN shows excellent performance in classifying image data. In order to solve the problem of large class imbalance between pixels from microcalcifications and other breast tissue, the research team of Mordang, Janssen [77] applied a CNN architecture which is inspired by OxfordNet to classify the microcalcification pixels which are difficult to be identified. The researchers also used cascade classifier for comparison and find out that CNN classifier had significantly better results for different data set with a sensitivity increase of about 6%. Except the Mammography, breast cancer cytological specimen can also be analyzed in order to diagnose benign and malignant cases. To improve the diagnosis of breast cancer, team of Gao, Wu [78] presented a network in combination of a shallow CNN and a deep CNN. They developed the shallow one to achieve 'virtual' recombined images from input dataset and apply the deep CNN to extract features. Finally, the recombined and 'virtual' recombined images would help the model to classify benign or cancer category of brain circumstance. In the research of Sun, Tseng [79], the group applied CNN into a semi-supervised learning scheme for breast cancer detection, which only needs very few labelled data for training. This scheme allows the usage of unlabelled data if a small amount of labelled data is already learned by the system. The final result achieves 88.18% with unlabelled data and 82.43% with the mixed and unlabelled data, which shows a significant improvement than previous research in accuracy. In the pixel-wise region segmentation scheme proposed by Su, Liu [80], fast scanning deep CNN was applied to relieve the computational burden by reducing the redundant computations in the convolutional layers and max pooling layers. Ting, Tan [81] introduced a model for breast cancer diagnosis and named it as Convolutional Neural Network Improvement for Breast Cancer Classification (CNNI-BCC). This algorithm achieves 89.47% sensitivity

and 90.50% accuracy. Bardou, Zhang [82] tried to figure out the breast cancer classification on the basis of histology images utilizing CNN. The result reveals that CNN performs better than the handcrafted feature-based classifier with 98.33% for binary classification and 82.33% for multi-class classification.

Table 1 Results of application of convolutional neural network in breast area

Authors	Goals	Model	Results	
			Acc	Sen
D Lévy et al. [76]	Classify the mass, benign or malignant	AlexNet	0.8900	0.8680
		GoogLeNet	0.9290	0.9340
J Mordang et al. [77]	Detect microcalcifications and then classify the mass, benign or malignant	Two CNNs	/	0.7035±0.0068
F Gao et al. [78]	Classify the benign or cancer	Shallow-Deep CNN	0.90±0.06	/
W Sun et al. [79]	Allow the usage of unlabelled data in breast cancer diagnosis	A CNN model including 3 pairs of convolutional layers	0.8818	/
H Su et al. [80]	Relieve computational burden in segmentation by reducing the redundant computations	Fast scanning CNN	0.9100	/
F Ting et al. [81]	Classify incoming medical images as malignant, benign or normal patient	CNNI-BCC	0.9050	0.8947
D Bardou et al. [82]	Binary classification	CNN using an ensemble model	0.9833	/
	Multi-class classification		0.8233	

(Acc= Accuracy; Sen = Sensitivity)

5.2. Lung

Lung cancer is a major global health problem today and for decades to come, as it is the major cause of cancer deaths worldwide, far outnumbering breast cancer, accounting for 18.4% of all cancer deaths. The data show that about 70% of patients are diagnosed with advanced lung cancer, so only 15% of patients with lung cancer survive five years after being diagnosed. This means that early diagnosis and improved diagnostic accuracy are crucial to the treatment of lung cancer and this area of technology has long been the subject of research [83, 84].

Alakwaa, Nassef [85] with his team used CT scans as input and 3D-CNN as model to detect and classify the lung unmarked nodules. They first segmented out lung tissue by thresholding, and then, a modified U-Net that can determine nodule candidates in CT scans is used to locate the regions that has relatively high possibility of containing nodule candidates. Finally, these regions with nodule candidates were sent to 3D-CNN to be identified if there exists lung cancer, since the 3D-CNN could make fully usage of the spatial 3D context information of lung nodules. The accuracy of model achieved 86.6%. In another similar research displayed in Table 3 in detail, the group of Kang, Liu [86] added the concept of multi-view strategy into the usage of 3D-CNN to improve the model performance. They conducted a binary classification and a three-type classification on CT images and compared results among different methods. Compared with 2D CNN, 3D CNN is no doubt better in error rate whatever in binary classification or ternary classification. Moreover, compared with one-view-one-network strategy, the multi-view-one-network strategy also achieves a lower error rate. The final best results achieved a

superior error rate of 4.59% for the binary classification and 7.70% for the ternary classification. One breakthrough brought by Shen, Zhou [87] and his team was that, in terms of regional analysis, instead of depending on nodule segmentation, they directly modeled raw nodule patches without any prior definition of nodule morphology. A Multi-scale Convolutional Neural Networks (MCNN) model was set up to capture nodule heterogeneity and the input also comes from CT which shows both lung nodule screening and nodule annotations. The best accuracy rate achieved 86.84 % with the RF classifier as displayed in Table 2. The first model for dealing lung pattern classification problem was presented by Anthimopoulos, Christodoulidis [88] and proven to be effective when compared to previous methods on a public dataset. In the research of Hamidian, Sahiner [89], they first presented a fully convolution network (FCN) transferred from 3D CNN and used it to produce scoring results for all the volumes rather than just volumes of interest. Then, utilizing the difficult negative examples produced by FCN to train a new discriminant CNN model. The whole system was a combination of FCN and the new CNN. An excellent result in lung nodule characterization was achieved by Tumor Net in the work of Hussein, Gillies [90]. The network they proposed had nearly 6 percent improvement in accuracy when compared to methods without utilizing CNN on the same dataset.

Table 2 Results of application of convolutional neural network in lung area

Authors	Goals	Model	Results	
			Acc	Sen
W Alakwaa et al. [85]	Detect the lung unmarked nodules	U-Net and 3D CNN	0.867	0.853
G Kang et al. [86]	Binary classification	MVO-3D CNN-CNN1	0.9541	/
	Ternary classification		0.923	/
W Shen et al. [87]	Model raw nodules to detect nodule heterogeneity	Multi-scale CNN	0.8684	/
A Chris et al. [88]	Lung Pattern Classification	Self-created	0.8561	/
H Sahiner et al. [89]	Automatically detect lung nodules	3D FCN+CNN	/	0.95
H Gillies et al. [90]	Lung nodule characterization	TUMORNET	0.9231	/

(MVO = multi-view-one; Acc= Accuracy; Sen = Sensitivity)

*(CT1= CT image channel 1; CT3= CT image channel 3)

Table 3 Details in research of G Kang et al. [86]

Input data	Goals	Model			Acc	Sen	Spc
CT 1	binary classification	MVO	3D CNN	CNN1	0.9518	0.9556	0.9301
				CNN2	0.9503	0.9551	0.9299
				CNN3	0.9481	0.9845	0.8976
			2D CNN	CNN1	0.9454	0.8888	0.9967
				CNN2	0.944	0.8893	0.9972
				CNN3	0.9413	0.894	0.9874
CT 3			3D CNN	CNN1	0.9525	0.956	0.9394
				CNN2	0.9524	0.9561	0.9389

				CNN3	0.9362	0.9417	0.8973
			2D CNN	CNN1	0.9459	0.8892	1
				CNN2	0.9446	0.89	0.9976
				CNN3	0.944	0.8897	0.9968
CT		MVO	3D CNN	CNN	0.9541	/	/
		OVO		CNN1	0.9472	/	/
				CNN2	0.9439	/	/
CT 1	ternary classification	MVO	3D CNN	CNN1	0.9188	/	/
				CNN2	0.9158	/	/
				CNN3	0.8906	/	/
			2D CNN	CNN1	0.8234	/	/
				CNN2	0.8373	/	/
				CNN3	0.8479	/	/
CT 3			3D CNN	CNN1	0.9222	/	/
				CNN2	0.9175	/	/
				CNN3	0.9069	/	/
			2D CNN	CNN1	0.8184	/	/
				CNN2	0.8195	/	/
				CNN3	0.8681	/	/
CT		MVO	3D CNN	CNN	0.923	/	/
		OVO		CNN1	0.9187	/	/
				CNN2	0.9093	/	/

(MVO = multi-view-one; OVO = one-view-one; Acc= Accuracy; Sen = Sensitivity; Spc = Specificity)

*(CT1= CT image channel 1; CT3= CT image channel 3)

5.3. Brain

In the application field of brain, CNN has also made outstanding contributions as illustrated in Table 4. It is well known that there are huge differences in appearance between brain tumors, which undoubtedly poses difficult challenges for pathologists or doctors to accurately segment and identify brain tumors. Thus, making use of deep learning to segment and identify brain tumors becomes a promising way of thinking in human medicine [91-100].

Khan, Shah [101] and his team proposed a fully automated CNN-based method for segmenting brain tumors that required a combination of handcrafted features, and achieved excellent results in terms of complete, core, and enhancing tumor. More recently, a fine-tuned VGG-19 (FT-VGG-19) approach to classifying multistage brain tumors that can help radiologists better analyze MRI has yielded compelling results, according to Sajjad, Khan [102] and his team. This classification system not only made use of deep learning to segment the tumor region in the image, but also made usage of data expansion to solve

the problem of insufficient data, which showed improved performance in comparison to other existing methods. In addition, to help doctors detect gliomas at an early stage, Anaraki, Ayati [103] and his team came up with a glioma classifier that combines genetic algorithms with a CNN model (CNN-GA) and gets it right more than 90 percent of the time. This classifier is very flexible because it can be effectively classified in multiple case studies on the classification of brain tumors. Moeskops, Viergever [104] introduced an automatic segmentation method based on CNN, which aims to segment MR brain images into lots of tissue classes. This method tried to learn to recognize the significant information for classification on the basis of training data rather than just dependent on explicit features. The team applied the methods to 5 datasets and achieved good Dice similarity score (Dss) over segmented tissue classes with the highest result of 0.91, which proves that this automatic segmentation method using CNN has robustness to datasets of different age and acquisition protocol. A brain extraction tool based on the auto-context CNN invented by Salehi, Erdogmus [105] outperforms several results recently reported in brain analysis research. It was superior than other brain extraction methods not only in high Dss results, but also due to its characteristic of registration-free, which means this CNN technique relied less on accuracy of alignment between atlases and anatomy in brain and did not require making assumptions in geometry of brain image. The team of Kamnitsas, Chen [106] introduced a multi-scale CNN solution to segment brain lesion and achieved better performance than previous method on a small-scale dataset. Hussain, Anwar [107] presented several architectures based on CNN to segment Glioma tumors in brainand reaches higher results when compared to state-of-art methods. Mehta and Sivaswamy [108] introduced M-net to help segmenting human brain structure. M-net attracted attention in medical research field as it ran 3 times faster in speed when compared to traditional methods.

Table 4 Results of application of convolutional neural network in brain area

Authors	Goals	Model	Results			
			Acc	Dss		
				Com	Co	E
K Shah et al. [101]	Fully automatically segment the brain the brain tumor	TP-CNN**	/	0.81	0.76	0.73
S Khan et al. [102]	Classify multi-grade brain tumor	FT-VGG-19	0.9458	/		
A Ayati et al. [103]	Classify different grades of glioma	CNN-GA	0.909	/		
			0.942	/		
P Moeskops et al. [104]	Segment MR brain images into tissue classes	Self-created	/	0.91		
S Erdogmus et al. [105]	Brain extraction	Auto-context CNN	/	0.9773		
K Chen et al. [106]	Segment lesion	Multi-scale CNN	/	0.66		
H Anwar et al. [107]	Segment Glioma tumors in brain	Self-created	/	0.86	0.87	0.90
M and S[108]	Segment deep brain structure	M-net	/	0.8578		

(Dss= Dice Similarity Score; Acc=Accuracy; Com=Complete; Co=Core; E=Enhancing)

* (TP-CNN is a combination of CSGP+MRILP+MRIGP)

Overall, we can see that CNN does bring excellent performance when compared to other state-of-

art methods in various areas of medical image analysis. It is often possible to quickly achieve accuracy of up to 90% or more than 95% from data sets that human pathologists may expend considerable effort and time on. There is no doubt that CNN will be of great help to the diagnosis in the medical image analysis area.

6. CONCLUSION

This review presents a comprehensive view on Convolutional Neural Network in the medical field. The content including its developing history, basic operating mechanism, basic structure, classic architecture and applications in medical image analysis.

We first outlined the basic operating mechanism of convolutional neural network, with the survey in neuron, activation function, loss function, gradient descent, feedforward and feedback propagation. And then detailedly introduced 4 categories of convolutional layer, 3 categories of pooling layer, and fully connected layer. The major benefits and limitations of several classic architecture: AlexNet, VGG-Net, Inception, Resnet, U-Net, R-CNN Series, YOLO and SSD are concluded. The applications of CNN among breast, lung and brain area are finally considered from summarizing a number of relevant core journals. We hope that by reading our review, readers can understand the basic principles and popular use framework of CNN, be able to utilize popular format of convolutional or pooling layer like transposed convolutional layer, separable convolutional layer, stochastic pooling in appropriate circumstances, and feel the rapid development of CNN and outstanding contributions brought by CNN in the field of medicine.

A growing number of cases prove that CNN can explore and discover the mysteries of some unknown medical fields. It is expected that the CNN could effectively has the ability to aid diagnostic applications for evaluating more and more diseases in humans in the future.

Acknowledgement

This work was partially supported by Royal Society International Exchanges Cost Share Award, UK (RP202G0230); Medical Research Council Confidence in Concept Award, UK (MC_PC_17171); Hope Foundation for Cancer Research, UK (RM60G0680); British Heart Foundation Accelerator Award, UK; Guangxi Key Laboratory of Trusted Software (kx201901); Fundamental Research Funds for the Central Universities (CDLS-2020-03); Key Laboratory of Child Development and Learning Science (Southeast University), Ministry of Education.

Reference

1. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning.* nature, 2015. **521**(7553): p. 436-444.
2. Schmidhuber, J., *Deep learning in neural networks: An overview.* Neural networks, 2015. **61**: p. 85-117.
3. Reddy, S., *Use of artificial intelligence in healthcare delivery*, in *eHealth-Making Health Care Smarter*. 2018, IntechOpen. p. p. 81-97.
4. McCulloch, W.S. and W. Pitts, *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics, 1943. **5**(4): p. 115-133.
5. Rosenblatt, F., *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review, 1958. **65**(6): p. 386.

6. Minsky, M. and S.A. Papert, *Perceptrons: An introduction to computational geometry*. 2017: MIT press.
7. Zhang, Z. and M. Sarhadi. *A modified neuron activation function which enables single layer perceptrons to solve some linearly inseparable problems*. in *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*. 1993. IEEE.
8. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, *Learning representations by back-propagating errors.* nature, 1986. **323**(6088): p. 533-536.
9. Lawrence, S., et al., *Face recognition: A convolutional neural-network approach.* IEEE transactions on neural networks, 1997. **8**(1): p. 98-113.
10. Mikolov, T., et al. *Recurrent neural network based language model*. in *Eleventh annual conference of the international speech communication association*. 2010.
11. Schuster, M. and K.K. Paliwal, *Bidirectional recurrent neural networks.* IEEE transactions on Signal Processing, 1997. **45**(11): p. 2673-2681.
12. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
13. Baudat, G. and F. Anouar, *Generalized discriminant analysis using a kernel approach.* Neural computation, 2000. **12**(10): p. 2385-2404.
14. Agarap, A.F., *Deep learning using rectified linear units (relu).* arXiv preprint arXiv:1803.08375, 2018.
15. Gauen, K., et al. *Low-power image recognition challenge*. in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2017. IEEE.
16. Krizhevsky, A., I. Sutskever, and G.E. Hinton. *ImageNet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*. 2012.
17. Szegedy, C., et al. *Inception-v4, inception-resnet and the impact of residual connections on learning*. in *Thirty-first AAAI conference on artificial intelligence*. 2017.
18. Howard, A.G., *Some improvements on deep convolutional neural network based image classification.* arXiv preprint arXiv:1312.5402, 2013.
19. Li, J., et al. *A comparison of deep learning methods for environmental sound detection*. in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. IEEE.
20. Marsalli, M. *Mcculloch-pitts neurons*. in *The 2008 Annual Meeting of the consortium on cognitive science instruction (ccsi)*. 2006.
21. Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning.[Sl]*. 2016, MIT press: 1 Rogers Street in Cambridge, MA 02142.
22. Nielsen, M.A., *Neural networks and deep learning*. Vol. 2018. 2015: Determination press San Francisco, CA, USA:.
23. Scardapane, S., et al., *Kafnets: Kernel-based non-parametric activation functions for neural networks.* Neural Networks, 2019. **110**: p. 19-32.
24. Chung, H., S.J. Lee, and J.G. Park. *Deep neural network using trainable activation functions*. in *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016. IEEE.
25. Baldassi, C., E.M. Malatesta, and R. Zecchina, *Properties of the Geometry of Solutions and Capacity of Multilayer Neural Networks with Rectified Linear Unit Activations.* Physical Review Letters, 2019. **123**(17): p. 6.
26. Pedamonti, D., *Comparison of non-linear activation functions for deep neural networks on MNIST classification task.* arXiv preprint arXiv:1804.02763, 2018.
27. Lemaréchal, C., *Cauchy and the gradient method.* Doc Math Extra, 2012. **251**: p. 254.
28. Ruder, S., *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747, 2016.
29. Bottou, L., *Large-scale machine learning with stochastic gradient descent*, in *Proceedings of COMPSTAT'2010*. 2010, Springer. p. 177-186.
30. Bottou, L., *Stochastic gradient descent tricks*, in *Neural networks: Tricks of the trade*. 2012, Springer. p. 421-436.
31. Khirirat, S., H.R. Feyzmahdavian, and M. Johansson. *Mini-batch gradient descent: Faster convergence under data sparsity*.

in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017. IEEE.
32. Hinton, G., N. Srivastava, and K. Swersky, *Overview of mini-batch gradient descent.* Neural Networks for Machine Learning, 2012. **575**.
33. Bengio, Y., P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult.* IEEE transactions on neural networks, 1994. **5**(2): p. 157-166.
34. Pascanu, R., T. Mikolov, and Y. Bengio, *Understanding the exploding gradient problem.* CoRR, abs/1211.5063, 2012. **2**: p. 417.
35. LeCun, Y., et al., *Backpropagation applied to handwritten zip code recognition.* Neural computation, 1989. **1**(4): p. 541-551.
36. Che, Z.-G., T.-A. Chiang, and Z.-H. Che, *Feed-forward neural networks training: a comparison between genetic algorithm and back-propagation learning algorithm.* International journal of innovative computing, information and control, 2011. **7**(10): p. 5839-5850.
37. Schmidt, W.F., et al. *Initializations, back-propagation and generalization of feed-forward classifiers*. in *IEEE International Conference on Neural Networks*. 1993. IEEE.
38. Rastegari, M., et al. *Xnor-net: Imagenet classification using binary convolutional neural networks*. in *European conference on computer vision*. 2016. Springer.
39. Rouhi, R., et al., *Benign and malignant breast tumors classification based on region growing and CNN segmentation.* Expert Systems with Applications, 2015. **42**(3): p. 990-1002.
40. Dasgupta, A. and S. Singh. *A fully convolutional neural network based structured prediction approach towards the retinal vessel segmentation*. in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. 2017. IEEE.
41. Juefei-Xu, F., V. Naresh Boddeti, and M. Savvides. *Local binary convolutional neural networks*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
42. Li, Y., X. Zhang, and D. Chen. *Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
43. Yu, F. and V. Koltun, *Multi-scale context aggregation by dilated convolutions.* arXiv preprint arXiv:1511.07122, 2015.
44. Zeiler, M.D., G.W. Taylor, and R. Fergus. *Adaptive deconvolutional networks for mid and high level feature learning*. in *2011 International Conference on Computer Vision*. 2011. IEEE.
45. Zeiler, M.D. and R. Fergus. *Visualizing and understanding convolutional networks*. in *European conference on computer vision*. 2014. Springer.
46. Dumoulin, V. and F. Visin, *A guide to convolution arithmetic for deep learning.* arXiv preprint arXiv:1603.07285, 2016.
47. Chollet, F. *Xception: Deep learning with depthwise separable convolutions*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
48. Chen, L.-C., et al. *Encoder-decoder with atrous separable convolution for semantic image segmentation*. in *Proceedings of the European conference on computer vision (ECCV)*. 2018.
49. Sheng, T., et al. *A quantization-friendly separable convolution for mobilenets*. in *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*. 2018. IEEE.
50. Sun, M., et al., *Learning pooling for convolutional neural network.* Neurocomputing, 2017. **224**: p. 96-104.
51. Zubair, S., F. Yan, and W. Wang, *Dictionary learning based sparse coefficients for audio classification with max and average pooling.* Digital Signal Processing, 2013. **23**(3): p. 960-970.
52. Zeiler, M.D. and R. Fergus, *Stochastic pooling for regularization of deep convolutional neural networks.* arXiv preprint arXiv:1301.3557, 2013.
53. Yu, D., et al. *Mixed pooling for convolutional neural networks*. in *International conference on rough sets and knowledge technology*. 2014. Springer.
54. Huang, J.-T., J. Li, and Y. Gong. *An analysis of convolutional neural networks for speech recognition*. in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015. IEEE.

55. Zarándy, Á., et al. *Overview of CNN research: 25 years history and the current trends*. in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2015. IEEE.
56. Iandola, F.N., et al., *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size.* arXiv preprint arXiv:1602.07360, 2016.
57. Yuan, Z.-W. and J. Zhang. *Feature extraction and image retrieval based on AlexNet*. in *Eighth International Conference on Digital Image Processing (ICDIP 2016)*. 2016. International Society for Optics and Photonics.
58. Simonyan, K. and A. Zisserman, *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556, 2014.
59. Mateen, M., et al., *Fundus image classification using VGG-19 architecture with PCA and SVD.* Symmetry, 2019. **11**(1): p. 1.
60. Szegedy, C., et al. *Rethinking the inception architecture for computer vision*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
61. Szegedy, C., et al. *Going deeper with convolutions*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
62. Lin, M., Q. Chen, and S. Yan, *Network in network.* arXiv preprint arXiv:1312.4400, 2013.
63. Salekin, M.S., A.B. Jelodar, and R. Kushol. *Cooking state recognition from images using inception architecture*. in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*. 2019. IEEE.
64. He, K., et al. *Identity mappings in deep residual networks*. in *European conference on computer vision*. 2016. Springer.
65. Ronneberger, O., P. Fischer, and T. Brox. *U-net: Convolutional networks for biomedical image segmentation*. in *International Conference on Medical image computing and computer-assisted intervention*. 2015. Springer.
66. Gkioxari, G., R. Girshick, and J. Malik. *Contextual action recognition with r* cnn*. in *Proceedings of the IEEE international conference on computer vision*. 2015.
67. Lenc, K. and A. Vedaldi, *R-cnn minus r.* arXiv preprint arXiv:1506.06981, 2015.
68. Purkait, P., C. Zhao, and C. Zach, *SPP-Net: Deep absolute pose regression with synthetic views.* arXiv preprint arXiv:1712.03452, 2017.
69. Girshick, R. *Fast r-cnn*. in *Proceedings of the IEEE international conference on computer vision*. 2015.
70. Ren, S., et al. *Faster r-cnn: Towards real-time object detection with region proposal networks*. in *Advances in neural information processing systems*. 2015.
71. Du, J. *Understanding of object detection based on CNN family and YOLO*. in *Journal of Physics: Conference Series*. 2018. IOP Publishing.
72. Liu, W., et al. *Ssd: Single shot multibox detector*. in *European conference on computer vision*. 2016. Springer.
73. Schairer, C., et al., *Probabilities of death from breast cancer and other causes among female breast cancer patients.* Journal of the National Cancer Institute, 2004. **96**(17): p. 1311-1321.
74. Azamjah, N., Y. Soltan-Zadeh, and F. Zayeri, *Global Trend of Breast Cancer Mortality Rate: A 25-Year Study.* Asian Pacific journal of cancer prevention: APJCP, 2019. **20**(7): p. 2015.
75. Waks, A.G. and E.P. Winer, *Breast cancer treatment: a review.* Jama, 2019. **321**(3): p. 288-300.
76. Lévy, D. and A. Jain, *Breast mass classification from mammograms using deep convolutional neural networks.* arXiv preprint arXiv:1612.00542, 2016.
77. Mordang, J.-J., et al. *Automatic microcalcification detection in multi-vendor mammography using convolutional neural networks*. in *International Workshop on Breast Imaging*. 2016. Springer.
78. Gao, F., et al., *SD-CNN: A shallow-deep CNN for improved breast cancer diagnosis.* Computerized Medical Imaging and Graphics, 2018. **70**: p. 53-62.
79. Sun, W., et al., *Enhancing deep convolutional neural network scheme for breast cancer diagnosis with unlabeled data.* Computerized Medical Imaging and Graphics, 2017. **57**: p. 4-9.
80. Su, H., et al. *Region segmentation in histopathological breast cancer images using deep convolutional neural network*. in

2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI). 2015. IEEE.

81. Ting, F.F., Y.J. Tan, and K.S. Sim, *Convolutional neural network improvement for breast cancer classification.* Expert Systems with Applications, 2019. **120**: p. 103-115.
82. Bardou, D., K. Zhang, and S.M. Ahmad, *Classification of breast cancer based on histology images using convolutional neural networks.* IEEE Access, 2018. **6**: p. 24680-24693.
83. Bray, F., et al., *Erratum: Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries.*
84. de Koning, H.J., et al., *Reduced lung-cancer mortality with volume CT screening in a randomized trial.* New England Journal of Medicine, 2020. **382**(6): p. 503-513.
85. Alakwaa, W., M. Nassef, and A. Badr, *Lung cancer detection and classification with 3D convolutional neural network (3D-CNN).* Lung Cancer, 2017. **8**(8): p. 409.
86. Kang, G., et al., *3D multi-view convolutional neural networks for lung nodule classification.* PloS one, 2017. **12**(11): p. p. 1-21.
87. Shen, W., et al. *Multi-scale convolutional neural networks for lung nodule classification*. in *International Conference on Information Processing in Medical Imaging*. 2015. Springer.
88. Anthimopoulos, M., et al., *Lung pattern classification for interstitial lung diseases using a deep convolutional neural network.* IEEE transactions on medical imaging, 2016. **35**(5): p. 1207-1216.
89. Hamidian, S., et al. *3D convolutional neural network for automatic detection of lung nodules in chest CT*. in *Medical Imaging 2017: Computer-Aided Diagnosis*. 2017. International Society for Optics and Photonics.
90. Hussein, S., et al. *Tumornet: Lung nodule characterization using multi-view convolutional neural network with gaussian process*. in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. 2017. IEEE.
91. Toğaçar, M., B. Ergen, and Z. Cömert, *BrainMRNet: Brain tumor detection using magnetic resonance images with a novel convolutional neural network model.* Medical Hypotheses, 2020. **134**: p. 109531.
92. Xue, Y., et al., *A multi-path 2.5 dimensional convolutional neural network system for segmenting stroke lesions in brain MRI images.* NeuroImage: Clinical, 2020. **25**: p. 102118.
93. Jiang, Y., et al., *A novel distributed multitask fuzzy clustering algorithm for automatic MR brain image segmentation.* Journal of medical systems, 2019. **43**(5): p. 118.
94. Jiang, Y., et al. *A Novel Negative-Transfer-Resistant Fuzzy Clustering Model with a Shared Cross-Domain Transfer Latent Space and its Application to Brain CT Image Segmentation*. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2020. DOI: 10.1109/TCBB.2019.2963873.
95. Jiang, Y., et al. *EEG-Based Driver Drowsiness Estimation Using an Online Multi-View and Transfer TSK Fuzzy System*. IEEE Transactions on Intelligent Transportation Systems, 2020. 1-13 DOI: 10.1109/TITS.2020.2973673.
96. Jiang, Y.Z., et al., *Seizure Classification From EEG Signals Using Transfer Learning, Semi-Supervised Learning and TSK Fuzzy System.* IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2017. **25**(12): p. 2270-2284.
97. Jiang, Y.Z., et al., *Recognition of Epileptic EEG Signals Using a Novel Multiview TSK Fuzzy System.* IEEE Transactions on Fuzzy Systems, 2017. **25**(1): p. 3-20.
98. Jiang, Y.Z., et al., *Collaborative Fuzzy Clustering From Multiple Weighted Views.* IEEE Transactions on Cybernetics, 2015. **45**(4): p. 688-701.
99. Jiang, Y.Z., et al., *Seizure Recognition Using a Novel Multitask Radial Basis Function Neural Network.* Journal of Medical Imaging and Health Informatics, 2019. **9**(9): p. 1865-1870.
100. Jiang, Y.Z., et al., *A Novel Synthetic CT Generation Method Using Multitask Maximum Entropy Clustering.* Ieee Access, 2019. **7**: p. 119644-119653.
101. Khan, H., et al., *Cascading handcrafted features and Convolutional Neural Network for IoT-enabled brain tumor segmentation.* Computer Communications, 2020.

102. Sajjad, M., et al., *Multi-grade brain tumor classification using deep CNN with extensive data augmentation.* Journal of computational science, 2019. **30**: p. 174-182.
103. Anaraki, A.K., M. Ayati, and F. Kazemi, *Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms.* Biocybernetics and Biomedical Engineering, 2019. **39**(1): p. 63-74.
104. Moeskops, P., et al., *Automatic segmentation of MR brain images with a convolutional neural network.* IEEE transactions on medical imaging, 2016. **35**(5): p. 1252-1261.
105. Salehi, S.S.M., D. Erdogmus, and A. Gholipour, *Auto-context convolutional neural network (auto-net) for brain extraction in magnetic resonance imaging.* IEEE transactions on medical imaging, 2017. **36**(11): p. 2319-2330.
106. Kamnitsas, K., et al., *Multi-scale 3D convolutional neural networks for lesion segmentation in brain MRI.* Ischemic stroke lesion segmentation, 2015. **13**: p. 46.
107. Hussain, S., S.M. Anwar, and M. Majid, *Segmentation of glioma tumors in brain using deep convolutional neural network.* Neurocomputing, 2018. **282**: p. 248-261.
108. Mehta, R. and J. Sivaswamy. *M-net: A convolutional neural network for deep brain structure segmentation*. in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. 2017. IEEE.