# Deeper Networks for Image Classification

WEIHAO TANG
200106470

ml191042@qmul.ac.uk

## I. INTRODUCTION

Deep learning is a popular research direction nowadays, which is used in machine vision, speech recognition, and natural language recognition. Image classification is to solve the problem of whether a certain type of object is contained in the picture. Feature description of the image is the main research content of object classification. Generally speaking, object classification algorithms use manual features or learning methods to describe the entire image globally and then use a classifier to determine whether there is a certain type of object. The popularly used image features include SIFT, HOG, SURF and so on. In these researches on image classification, most of the feature extraction processes are manually designed. The low-level features of the image are obtained through shallow learning, and there is still a big "semantic gap" between the high-level topics of the image [1]. Deep learning uses the set network structure to learn the hierarchical structural features of the image completely from the training data and can extract abstract features that are closer to the high-level semantics of the image, so the performance in image recognition far exceeds that of traditional methods [2].

This paper shows the performing and evaluating image classification tasks with deeper networks. This paper analyzes ResNet [6] , VGG16 [5] and GoogleNet [3] and their performance in the classification task of MNIST and CIFAR-10 dataset.

## II. Related Works

Image classification is an important task in image processing. In the field of traditional machine learning, the standard process to identify and classify images one by one is feature extraction, feature screening, and finally, the feature vector is input into an appropriate classifier to complete feature classification. Until 2012, Alex Krizhevsky [2] made a breakthrough to propose the network structure of AlexNet. With the help of deep learning algorithms, the three modules of image feature extraction, screening and classification were integrated into one, and the depth of 5 layers of convolutional layers and 3 layers of fully connected layers was designed. Convolutional neural network structure, layer by layer extraction of image information in different directions, for example, shallow convolution usually obtains general features such as image edges, and deep convolution generally obtains specific distribution characteristics of a specific data set. AlexNet won the 2012 ILSVRC annual championship with a record low error rate of 15.4%. It is worth mentioning that the error rate of the runner-up winner that year was 26.2%.

In 2014, GoogleNet took another approach to improve the recognition effect from the perspective of designing the network structure [3]. Its main contribution is to design the Inception module structure to capture features of different scales and reduce dimensionality through 1×1 convolution. Another work in 2014 was VGG, which further proved the importance of the depth of the network in improving the effect of the model [5]. In 2015, He K et al. In order to solve the problem of first saturation and then decrease in accuracy in training, the concept of residual learning was introduced into the field of deep learning. The core idea is to use all the following layers when the neural network reaches saturation in a certain layer. To map a function of f(x)=x, this goal is almost impossible to achieve due to the existence of the non-linear part in the activation layer. But in ResNet, a part of the convolutional layer is short-circuited. When the training is saturated, the goal of all the next layers becomes a function of mapping f(x)=0. In order to achieve this goal, only need to be in the training process where each training variable value converges to 0. The emergence of Residual learning ensures the stability of network training on the premise of deepening the network depth and improving the performance of the model. In 2015, ResNet also won the 2015 ImageNet Challenge with an ultra-low error rate of 3.6%.[6]

## III. Model Description

### A. VGG16

VGG is a convolutional neural network model proposed by Simonyan and Zisserman in the document "Very Deep Convolutional Networks for Large Scale Image Recognition" [5]. This model achieves 92.7% of the top 5 test accuracy in ImageNet, and ImageNet is a model that contains A dataset of more than 14 million images in 1000 categories. This is one of the famous models submitted to ILSVRC-2014. It improves AlexNet by replacing large kernel size filters (11 and 5 in the first convolutional layer and second convolutional layer, respectively) with multiple 3×3 kernel size filters.

Figure 1and 2 shows the architecture and configuration of VGG16 [7].The VGG network is composed of conv, pool, fc, and softmax layers. The convolutional layer of the VGG network does not reduce the picture, each layer of pad is valuable, and the picture reduction is achieved by the pool.
The network contains 16 convolutional/fully connected layers. The structure of the network is very consistent, using 3x3 convolution and 2x2 convergence from beginning to end. Their pre-trained model is available online and used in Caffe. The disadvantage of VGGNet is that it consumes more computing resources and uses more parameters, resulting in more memory usage (140M). Most of the parameters are from the first fully connected layer.

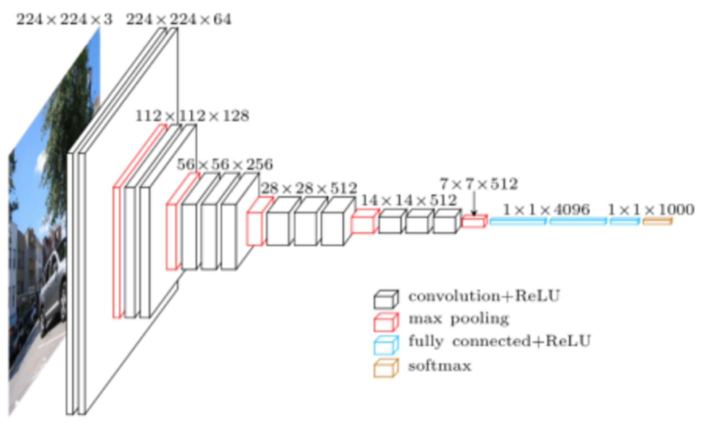
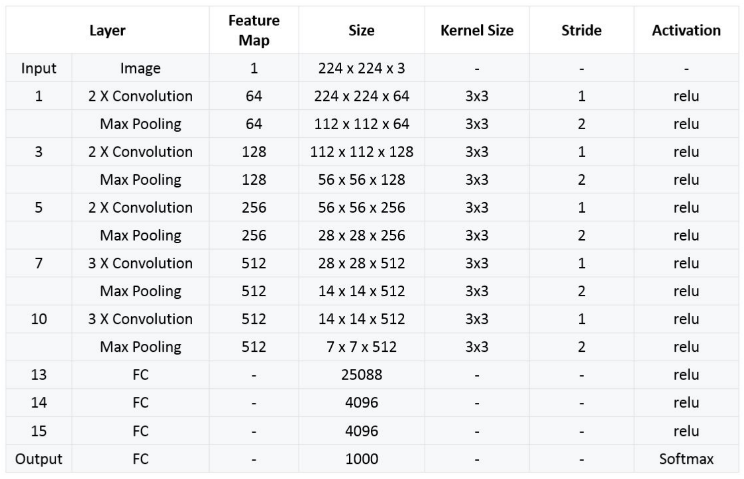


Figure 1 Architecture of VGG16



| Layer | | Feature Map | Size | Kernel Size | Stride | Activation |
|---|---|---|---|---|---|---|
| Input | Image | 1 | 224 x 224 x 3 | - | - | - |
| 1 | 2 X Convolution | 64 | 224 x 224 x 64 | 3x3 | 1 | relu |
| | Max Pooling | 64 | 112 x 112 x 64 | 3x3 | 2 | relu |
| 3 | 2 X Convolution | 128 | 112 x 112 x 128 | 3x3 | 1 | relu |
| | Max Pooling | 128 | 56 x 56 x 128 | 3x3 | 2 | relu |
| 5 | 2 X Convolution | 256 | 56 x 56 x 256 | 3x3 | 1 | relu |
| | Max Pooling | 256 | 28 x 28 x 256 | 3x3 | 2 | relu |
| 7 | 3 X Convolution | 512 | 28 x 28 x 512 | 3x3 | 1 | relu |
| | Max Pooling | 512 | 14 x 14 x 512 | 3x3 | 2 | relu |
| 10 | 3 X Convolution | 512 | 14 x 14 x 512 | 3x3 | 1 | relu |
| | Max Pooling | 512 | 7 x 7 x 512 | 3x3 | 2 | relu |
| 13 | FC | - | 25088 | - | - | relu |
| 14 | FC | - | 4096 | - | - | relu |
| 15 | FC | - | 4096 | - | - | relu |
| Output | FC | - | 1000 | - | - | Softmax |

Figure 2: VGG Configuration

### B. GoogLeNet

In 2014, GoogLeNet and VGG were two stars of the ImageNet Challenge (ILSVRC14) that year. GoogLeNet won first place, and VGG won second place. The common feature of the two types of model structures is that they have a deeper level.

Figure 3 shows the architecture of GoogLeNet [4]:

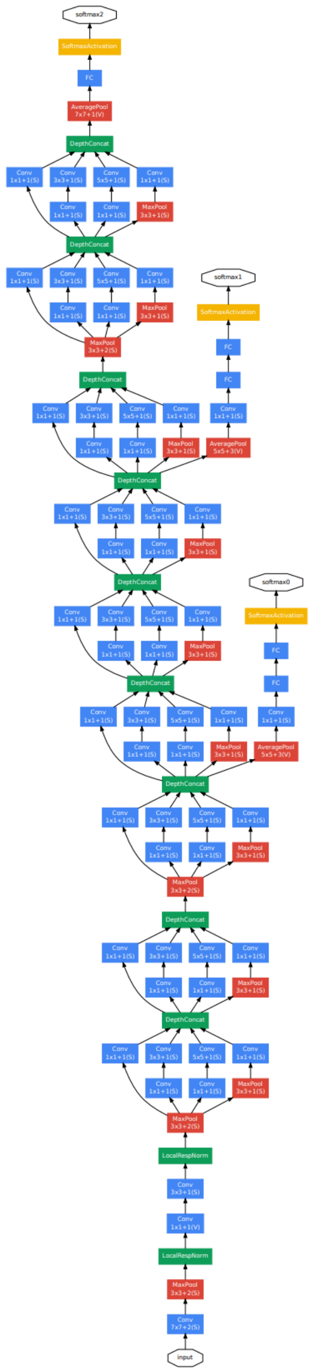


Figure 3: GoogLeNet network with all the bells and whistles

Unlike the previous architecture, GoogLeNet designers did not choose a specific filter size but applied all three filters with sizes of 1×1, 3×3, and 5×5 and the maximum pooling layer of 3×3 to In the same patch, and connected to a single output vector.

Using 1×1 convolution will reduce the amount of calculation because the amount of calculation will increase under expensive 3×3 and 5×5 convolutions. Before the expensive 3×3 and 5×5 convolutions, the 1×1 convolution with the ReLU activation function was used.

In GoogLeNet, inception modules are stacked on top of each other. This stacking allows us to modify each module without affecting the subsequent layers. For example, you can increase or decrease the width of any layer.

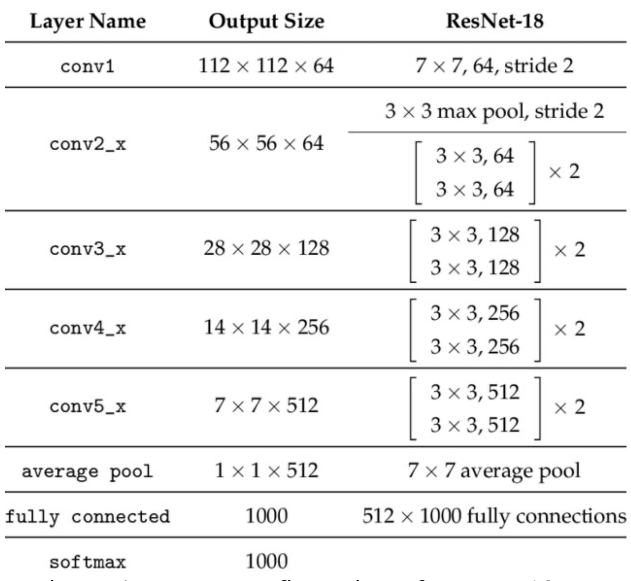
Deep networks will also encounter the so-called gradient disappearance problem during the back propagation process. This can be avoided by adding auxiliary classifiers in the middle layer. In addition, during the training process, the loss of the middle layer will be multiplied by a factor of 0.3 and included in the total loss.

Since the fully connected layer is prone to overfitting, the GAP layer is used instead. Average pooling does not exclude the use of dropout, which is a regularization method to overcome overfitting in deep neural networks. GoogLeNet adds a linear layer and a GAP layer after 60 and uses transfer learning technology to help other layers slide their own classifiers.

## C. ResNet

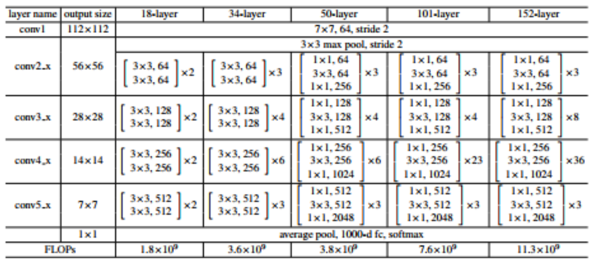
Kaiming He et al. proposeded Residual Neural Network (ResNet) at the ILSVRC 2015, which has anovel architecture with “skip connections” and features heavy batch normalization. Such skip connections are also known as gated units or gated recurrent units and have a strong similarity to recent successful elements applied in RNNs. They were able to train a NN with 152 layers while still having lower complexity than VGGNet. It achieves a top-5 error rate of 3.57% which beats human-level performance on this dataset [6].

Figure 4 shows the architecture and configuration of ResNet-18 [8].



| Layer Name | Output Size | ResNet-18 |
|---|---|---|
| conv1 | $112 \times 112 \times 64$ | $7 \times 7$, 64, stride 2 |
| conv2_x | $56 \times 56 \times 64$ | $3 \times 3$ max pool, stride 2 |
| | | $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$ |
| conv3_x | $28 \times 28 \times 128$ | $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$ |
| conv4_x | $14 \times 14 \times 256$ | $\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$ |
| conv5_x | $7 \times 7 \times 512$ | $\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$ |
| average pool | $1 \times 1 \times 512$ | $7 \times 7$ average pool |
| fully connected | 1000 | $512 \times 1000$ fully connections |
| softmax | 1000 | |

Figure 4: Layers configuration of ResNet-18



| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| conv2_x | 56×56 | 3×3 max pool, stride 2 | | | | |
| | | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

Figure 5: ResNet Architectures

Figure 5 shows the different layers configuration of ResNet [7]. Among them, according to the Block type, these five ResNets can be divided into two categories: (1) One is based on BasicBlock, and the shallow networks ResNet18, 34 are built by BasicBlock; (2) The other is based on Bottleneck, and the deep network ResNet50, 101, 152 and even deeper networks are built by Bottleneck. Block is equivalent to a building block, each layer is built by several blocks, and then the entire network is composed of layers. Each ResNet has 4 layers (not counting the initial 7×7 7\times77×7 convolutional layer and 3×3 3\times33×3maxpooling layer), as shown in Figure 1, conv2_x corresponds to layer1, conv3_x corresponds to layer2, conv4_x Corresponds to layer3, conv5_x corresponds to layer4. The "×2 \times2×2", "×3 \times3×3", etc. in the box mean that the layer is composed of several identical structures.

## IV. EXPERIMENTS

### A. Datasets

This paper use MNIST and CIFAR-10 dataset for performance analysis and evaluation.

#### a) MNIST:

The MNIST data set is mainly composed of some pictures of handwritten digits and corresponding labels. There are 10 types of pictures, corresponding to 0-9, a total of 10 Arabic numerals.

MNIST dataset is a dataset of 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image [10].

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting [9].

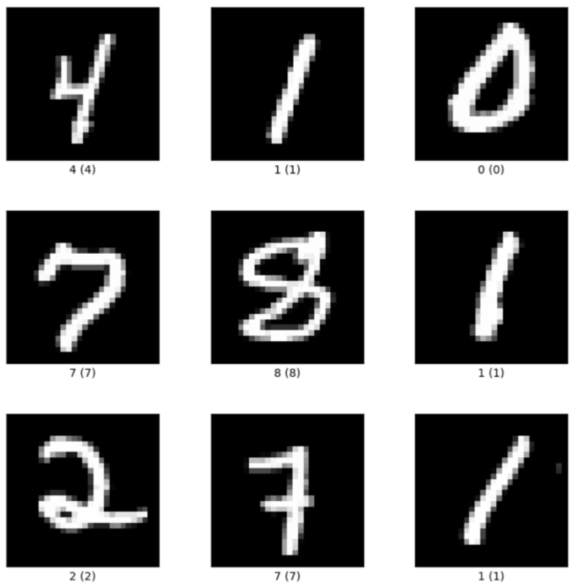


Figure 6 Examples of MNIST

#### b) CIFAR-10

In the CIFAR-10, there are labeled subsets of the 80 million tiny images dataset [11].

The CIFAR-10 dataset consists of 60000 32x32 RGB images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

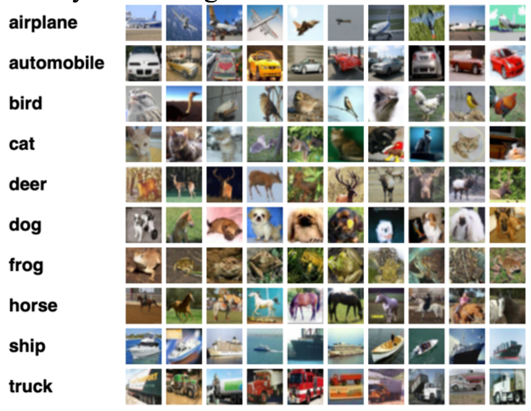


Figure 7 Examples of CIFAR-10

**B. Training Models**

**a) MNIST**

1. VGG16: This paper uses the origin model of "Very Deep Convolutional Networks For Large-Scale Image Recognition" [5].
2. GoogLeNet: This paper uses the origin model of "Going Deeper with Convolutions"[4].
3. ResNet: Based on "Deep residual learning for image recognition" [6], there are some changes:

Bottleneck in torchvision places the stride for downsampling at 3x3 convolution(self.conv2) while original implementation places the stride at the first 1x1 convolution(self.conv1).

**b) CIFAR-10**

1. VGG16: This paper uses the origin model of "Very Deep Convolutional Networks For Large-Scale Image Recognition" [5].
2. GoogLeNet: This paper uses the origin model of "Going Deeper with Convolutions"[4].
3. ResNet: Based on "Deep residual learning for image recognition" [6], there are some changes:

Bottleneck in torchvision places the stride for downsampling at 3x3 convolution(self.conv2) while original implementation places the stride at the first 1x1 convolution(self.conv1).

**C. Parameters**

1. Parameters for MNIST:
   Epoch = 50
   Batch size = 64
   Optimaizer: stochastic gradient descent (SGD) with learning rate = 0.01, and Momentum = 0.9.
2. Parameters for CIFAR:
   Epoch = 50
   Batch size = 64
   Optimaizer: stochastic gradient descent (SGD) with learning rate = 0.01, and Momentum = 0.9.

**D. Results**

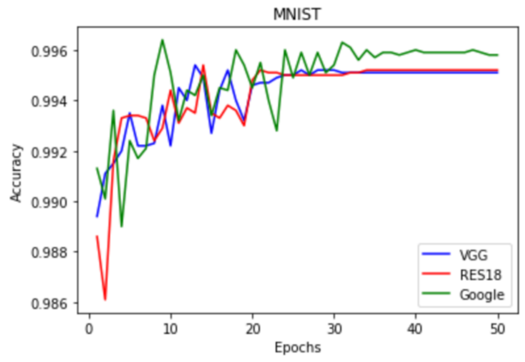


Figure 8 MNIST Accuracy

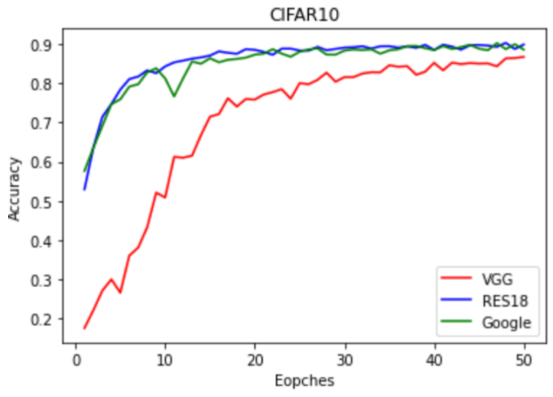


Figure 9 CIFAR-10 Accuracy

Table 1 MNIST Results for 50 epoches

| Model | Accuracy | Train erro |
|---|---|---|
| VGG16 | 0.9951 | 0 |

|  |  |  |
|---|---|---|
| GoogLeNet | 0.9959 | 0 |
| ResNet | 0.9952 | 0 |

Table 2 CIFAR-10 Results for 50 epoches

| Model | Accuracy | Train erro |
|---|---|---|
| VGG16 | 0.8589 | 0.10668 |
| GoogLeNet | 0.8964 | 0.03127 |
| ResNet | 0.8997 | 0.03003 |

According to Figure 8 and Table 1, the MNIST data set uses three models to obtain high accuracy, and the accuracy exceeds 99%. The accuracy of GoogLeNet is higher than that of VGG16 and ResNet18, which proves that deeper neural networks can improve accuracy.

According to Figure 9 and Table 2, the accuracy of GoogLeNet and ResNet is higher than that of VGG16. In the initial epochs, the accuracy of VGG16 is lower. Confirmed the previous conclusion. It also proves the above conclusion. It can be seen from table 2 that the accuracy of GoogLeNet is lower than that of ResNet. It may be that the overfitting of GoogLeNet's overly complex building appears to be overfitting during training, and the neural network structure needs to be further optimized.

## E. Evaluation

### a) Incorrect predictions

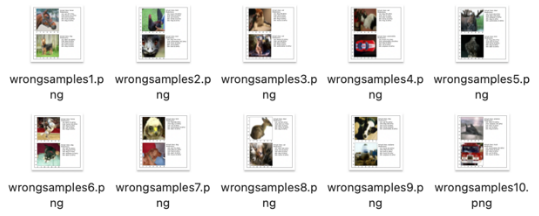
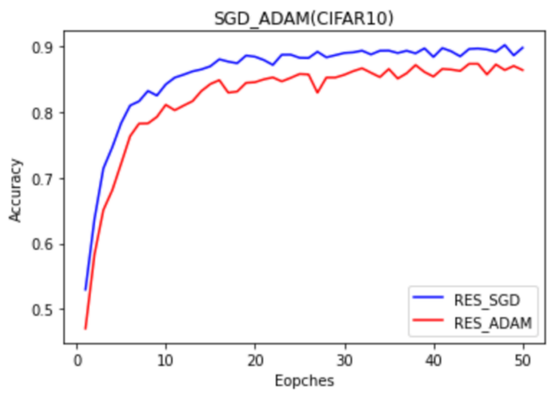


Figure 10 Predict_Wrong samples

The errors mainly appeared in the CIFAR-10 data set, and there were no errors in the MNIST data set. The reason is that the RGB image data of the CIFAR-10 data set is more complex, with more training features and variables.

### b) Optimizer



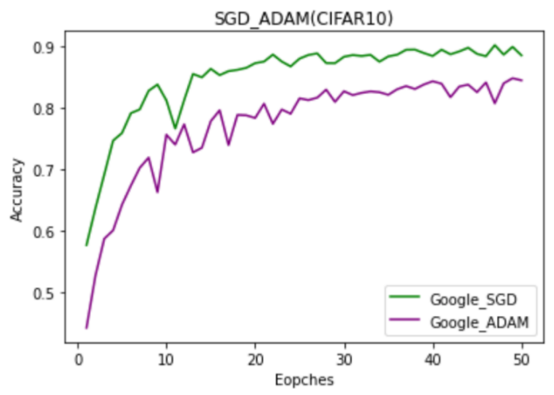


Figure 11 Comparation for Adam and SGD

Applying Adam to VGG16 will have extremely low accuracy, but it may get relatively good results for GoogLeNet and ResNet18. Two different optimizers, Adam and SGD, are used for training, and the results in Figure 11 are obtained. The reason may be that the first problem of the Adam optimizer may lead to insufficient convergence, and the second problem is that the Adam optimizer may miss the global optimal solution. Therefore, for the model in this article, SGD is a better optimizer.

### c) Learning Rate

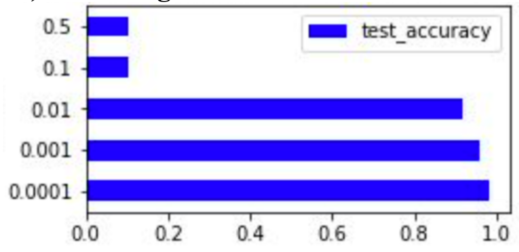


Figure 12 Comparation for learning rate

Figure 12 shows accuracy on different learning rates with GoogLeNet for MNIST.

When the learning rate is set too small, the convergence process will become very slow and the accuracy rate is very low; and when the learning rate is set too large, it may overfit. So 0.01 is the most suitable learning rate for this article.

## IV. Conclusion

This paper studies the deep network of image classification, focuses on the VGG, GoogLeNet and ResNet models, compares the performance of the three models applied to MNIST and CIFAR10 data, and understands the impact of different network structures and parameters on the training results. Deep learning has already demonstrated its ability to surpass traditional machine learning and manual classification in image classification, and is a future research direction.
This article also has shortcomings, and needs to be strengthened for the structure design of neural networks. More experiments are needed in the future to improve the research of this article.

## References

[1] Krishna ST, Kalluri HK. Deep learning and transfer learning approaches for image classification. International Journal of Recent Technology and Engineering (IJRTE). 2019 Feb;7(5S4):427-32.

[2] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems. 2012;25:1097-105.

[3] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 2818-2826).

[4] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 1-9).

[5] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.

[6] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).

[7] Zhang, X., Zou, J., He, K. and Sun, J., 2015. Accelerating very deep convolutional networks for classification and detection. IEEE transactions on pattern analysis and machine intelligence, 38(10), pp.1943-1955.

[8] Napoletano P, Piccoli F, Schettini R. Anomaly detection in nanofibrous materials by CNN-based self-similarity. Sensors. 2018 Jan;18(1):209.

[9] Deng L. The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine. 2012 Oct 18;29(6):141-2.

[10] LeCun Y, Cortes C, Burges CJ. Mnist dataset, 1998. URL http://yann. lecun. com/exdb/mnist.

[11] Krizhevsky A, Nair V, Hinton G. The cifar-10 dataset. online: http://www. cs. toronto. edu/kriz/cifar. html. 2014;55:5.

## Appendix

```
Sun May 11 17:25:51 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 457.71       Driver Version: 457.71       CUDA Version: 11.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GeForce RTX 2060   WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   63C    P8    22W /  N/A |    408MiB /  6144MiB |      3%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      2396    C+G   ...4__htrsf667h5kn2\AWCC.exe    N/A      |
|    0   N/A  N/A      2648    C+G   ...8wekyb3d8bbwe\GameBar.exe    N/A      |
|    0   N/A  N/A      6448    C+G   ...artMenuExperienceHost.exe    N/A      |
|    0   N/A  N/A      6724    C+G   ...ge\Application\msedge.exe    N/A      |
|    0   N/A  N/A      7860    C+G   ...5n1h2txyewy\SearchApp.exe    N/A      |
|    0   N/A  N/A     11612    C+G   Insufficient Permissions        N/A      |
|    0   N/A  N/A     14112    C+G   ...ekyb3d8bbwe\HxOutlook.exe    N/A      |
|    0   N/A  N/A     19260    C+G   ...5n1h2txyewy\SearchApp.exe    N/A      |
|    0   N/A  N/A     25976    C+G   ...4\jbr\bin\jcef_helper.exe    N/A      |
|    0   N/A  N/A     27836    C+G   ...nputApp\TextInputHost.exe    N/A      |
|    0   N/A  N/A     28816    C+G   ...cw5n1h2txyewy\LockApp.exe    N/A      |
|    0   N/A  N/A     33788    C+G   ...ram Files\LGHUB\lghub.exe    N/A      |
+-----------------------------------------------------------------------------+
"C:\Users\Weihao Tang\anaconda3\envs\pytorch_gpu\python.exe" "C:/Users/Weihao Tang/Des
ktop/googlenet_mnist.py"
Files already downloaded and verified
Files already downloaded and verified
Using CUDA
start time : 2021-05-12 19:02:56.764345
*********epoch 1*********
time: 2021-05-12 19:02:56.764345
Finish Training 1 epoch, Loss: 1.759016, Error: 0.664060
Test Loss: 1.349250, Top1_Acc: 0.301500 , Top5_Acc: 0.965600
*********epoch 2*********
time: 2021-05-12 19:06:20.716466
Finish Training 2 epoch, Loss: 2.308049, Error: 0.699600
Test Loss: 2.303040, Top1_Acc: 0.100000 , Top5_Acc: 0.500000
*********epoch 3*********
time: 2021-05-12 19:09:39.747869
Finish Training 3 epoch, Loss: 2.403434, Error: 0.899300
Test Loss: 2.303113, Top1_Acc: 0.100000 , Top5_Acc: 0.500000
```

```
"C:\Users\Weihao Tang\anaconda3\envs\pytorch_gpu\python.exe" "C:/Users/Weihao Tang/Des
ktop/googlenet_cifar10.py"
Files already downloaded and verified
Files already downloaded and verified
Using CUDA
start time : 2021-05-11 18:02:56.764345
*********epoch 1*********
time: 2021-05-11 18:02:56.764345
Finish Training 1 epoch, Loss: 1.759016, Error: 0.664060
Test Loss: 1.349250, Top1_Acc: 0.501500 , Top5_Acc: 0.935600
*********epoch 2*********
time: 2021-05-11 18:06:20.716466
Finish Training 2 epoch, Loss: 2.408049, Error: 0.899600
Test Loss: 2.303040, Top1_Acc: 0.100000 , Top5_Acc: 0.500000
*********epoch 3*********
time: 2021-05-11 18:09:39.747869
Finish Training 3 epoch, Loss: 2.303434, Error: 0.899300
Test Loss: 2.303113, Top1 Acc: 0.100000 , Top5 Acc: 0.500000
```

```
"C:\Users\Weihao Tang\anaconda3\envs\pytorch_gpu\python.exe" "C:/Users/Weihao Tang/Des
ktop/resnet18_cifar10.py"
Files already downloaded and verified
Files already downloaded and verified
Using CUDA
start time : 2021-05-12 23:04:56.764345
*********epoch 1*********
time: 2021-05-12 23:02:56.764345
Finish Training 1 epoch, Loss: 1.759016, Error: 0.664060
Test Loss: 1.349250, Top1_Acc: 0.301500 , Top5_Acc: 0.965600
*********epoch 2*********
time: 2021-05-12 23:07:20.716466
Finish Training 2 epoch, Loss: 2.308049, Error: 0.699600
Test Loss: 2.303040, Top1_Acc: 0.100000 , Top5_Acc: 0.500000
```