Deeper Networks for Image Classification

Razhan Hameed - 210667222[,⋆]

[1] *Department of Electronics Engineering and Computer Science, Queen Mary University Of London*

Abstract. Deep learning techniques based on deeper neural networks are achieving remarkable results in various domains. This paper showcases the performance and evaluation process for image classification tasks using deeper networks. It focuses on various implementations of AlexNet, VGG, GoogLeNet and ResNet and their performance in the image classification task of two different datasets. The chosen networks are all improvements over the previous one in terms of architecture, speed of training, and accuracy. Through experiments under various conditions, it can be found all the models achieved about 98% or above accuracy on the first dataset, which was MNIST, and 88.2% accuracy on the second dataset, which was CIFAR-10.

1 Introduction

Recent advances in image classification in the field of computer vision have been achieved using deep neural networks. Since CNNs created the opportunity to use very deep neural networks of over a hundred layers almost every week, a new architecture with a slight variation hits the headlines that had achieved higher accuracy than its previous competitors.

In this paper, I focus on investigating four different networks with regard to their effectiveness for image classification on the popular MNIST and CIFAR10 datasets. With the Pytorch framework's flexible implementation, we could experiment each network with many different configurations. The current paper addresses the architecture and performance of the following networks: AlexNet, VGG, GoogLeNet, and ResNet [1–4]. I also modified the ResNet block to just a 3x3 convolutional layer with no batch normalization, it did not give better results.

2 Related Work

Image classification is an essential computer vision task that attempts to understand an entire image. The objective is to classify an image by assigning it to a specific label (In the case of supervised learning). This section covers some other research using the networks mentioned above to perform image classification in various domains.

Traditional image classification required experts to manually extract features in different phases, like feature extraction and feature screening. Along came deep learning, which integrated the entire process into one end-to-end pipeline(from feature extraction to the final classified output). In 2012, AlexNet [**?**] opened the door for deeper neural networks in the task of image classification. The development in the field had helped in investigation of large datasets.

Yukun Huang et al.[5] deal with the high-resolution multispectral images, in which a new network is proposed. ResNet with the Residual pyramidal network, which improves the classification performance. In this paper, residual learning is suggested, and the core idea is to use all the following layers when the neural network reaches saturation in a particular layer.

This paper [6] detects the COVID-19 virus from X-ray images by using different variations of VGG, which we going to cover in the following section. The results are promising, as image classification can help doctors detect disease earlier because the networks can look at more images and memorize most of the details.

In 2017 MobileNet was proposed [7]. It attempted to reduce the number of parameters of the previous networks to make the network more lightweight so it could be deployed to mobile devices. There have been many iterations of this network since then, even making it lighter without losing the performance. In the following sections, we cover the original networks that inspired all the networks mentioned.

3 Model Description

3.1 AlexNet

When it comes to deep neural networks in the domain of computer vision, Alexnet is one of the first popular architectures that became popular. It won the Imagenet large-scale visual recognition challenge back in 2012. The paper [1] has been cited more than 100 thousand times. The overall network has eight layers; 5 convolutional layers with max-pooling, followed by three fully connected layers which use Relu as an activation function, except for the output layer that uses the Softmax function. The input layer is of dimensionality of size 227×227×3, where

[⋆]e-mail: ec21243@qmul.ac.uk

3 means the number of channels (R, G, B). The first convolutional layer has a filter size of 11×11, the number of filters 96, and the convolution stride s=4. (The filter size only lists the width and height, the number of channels of the filter matrix is the same as the number of channels of the input image).

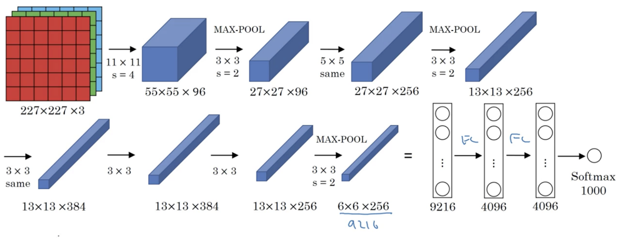


Figure 1: AlexNet Architecture

The reason they used Relu in the dense layers is that it increased the training process by six times. The model also uses dropout layers to prevent overfitting. The filter size is reduced to 5x5 and 3x3 in the second and third convolutional layers, respectively. As we go deeper into the architecture, the number of filters is increasing, therefore extracting more features. However, the filter size is reducing, which means in the beginning, the initial filter was larger, and as we go deeper, the filter size decreases, resulting in a decrease in the feature map shape.

3.2 VGG

VGG is a very classic convolutional neural network structure, and it was named after the Visual Geometry Group from oxford [2]. Its an improvement over Alexnet, mainly known for homogeneous topology and the increased depth. Compared with AlexNet, there are two main changes:

- Use 3 x 3 kernels instead of large kernels in AlexNet
- Use 2 x 2 pooling kernels instead of AlexNet's 3 x 3 pooling kernels

There are many types of VGGNet, and the paper proposes four different levels of network structures (from 11 layers to 19 layers) [2] VGG has many advantages. An essential feature is to replace the large convolution kernel with a small convolution kernel (3x3). Two 3x3 convolution stacks are equal to one 5x5 convolution, and three 3x3 stacks are equal to one 7x7 convolution. The size of the receptive field stays unchanged. It can be imagined that when the stride is 1 and the padding is 0, the size of the image after 2 3x3 convolutions is $(((N-3)/1+1)-3)/1+1 = ((N-3+1)-3+1) = N-4 = (N-5)/1+1$. And after convolution, the obtained features are all extracted from the same pixels on the original image (every 5x5 airspace pixels of the original image corresponds to a new feature), so the size of the receptive field remains unchanged. Therefore, two 3x3 convolution kernels are equivalent to 5x5 convolution kernels.

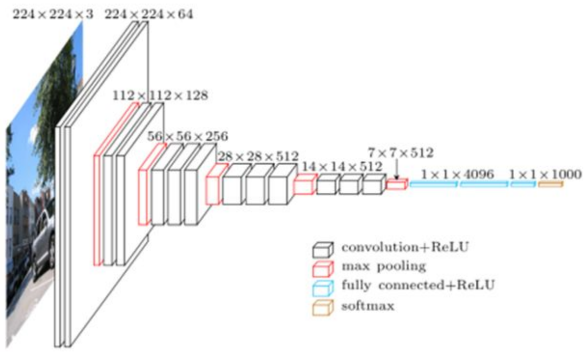


Figure 2: VGG Architecture

3.3 GoogLeNet

GoogLeNet [8] was proposed by the Google team around the same time as the VGG network came out (note that the capital L in GoogLeNet is to pay tribute to LeNet [3]) and won the first place in the Classification Task in the ImageNet competition that year. The Architecture has 22 deep Network layers with 27 pooling layers and proposes an Inception module with an excellent local feature structure; that is, multiple convolution operations and pooling of different sizes are performed on the features in parallel and finally stitched together.

GoogleNet contains 9 Inception modules which are split into 2,5,2 into 3 layers. These inception modules are connected to the global average pooling layer. Since the convolution operations of 1×1, 3×3, and 5×5 correspond to different feature map regions, this advantage is that better image representation information can be obtained. In order for the outputs of the four branches to be spliced in the depth order, the feature matrix height and width of the outputs of the four branches must be the same. The Inception module is shown in Figure 3. It uses three convolution kernels of different sizes for convolution operations and a maximum pooling. Then, it cascades these four parts (channel splicing) and sends them to the next layer.

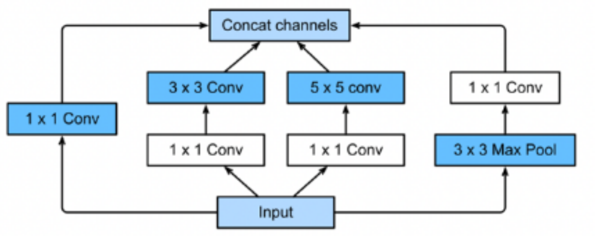


Figure 3: Inception Module

In order to reduce the number of network parameters, Inception has added multiple 1×1 convolution modules. As shown in Figure 3, this 1×1 module can reduce the dimension of the feature map first and then send it to the convolution kernels of 3×3 and 5×5 sizes. Due to reducing the number of channels, the amount of parameters has also been greatly reduced.

3.4 ResNet

When everyone was still amazed at the inception structure of GoogLeNet, researchers at Microsoft Research Asia were already designing a deeper but simpler network, ResNet [4]. With this network, they won the first place in the classification task and the first place in the target detection in the ImageNet competition. It won first place in object detection and image segmentation in COCO dataset. Figure 5 is a schematic diagram of the structure of the ResNet18 layer model In the ResNet paper[4], it is said

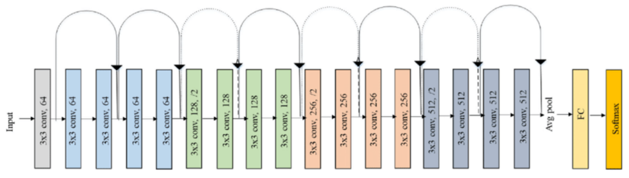


Figure 4: ResNet Architecture

that the problem of gradient disappearance or gradient explosion can be solved by data preprocessing and the use of the Batch Normalization layer in the network, and the residual structure (residual structure) can alleviate the degradation problem. At this point, the fitting target becomes F(x), and F(x) is the residual. The residual structure figure shows a very important point here. For the second layer, it does not have a Relu activation function. It needs to be added to x and then the activation function Relu is performed. The idea of residual network is based on identity shortcut connection, in which the network can skip some layers.

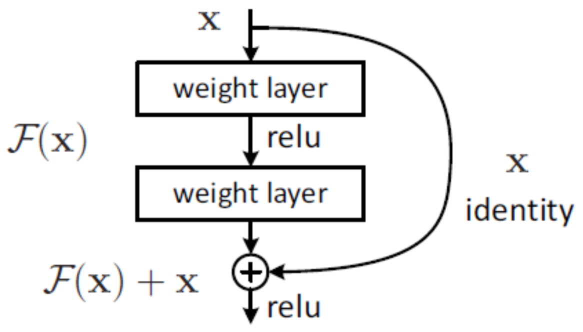

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 5: Residual Structure

3.4.1 *Residual structure of ResNet18/34*

We first conduct an analysis of the residual structure of ResNet18/34. The main branch of the residual structure is composed of two layers of 3x3 convolutional layers, and the connecting line on the right side of the residual structure is the shortcut branch, also known as the shortcut branch. In order to enable the output matrix on the main branch to be able to add to the output matrix on our shortcut branch, we must ensure that the two output feature matrices have the same shape. We will find that there are some dotted line structures, which are described in the paper as dimensionality reduction with 1x1 convolutions.

3.4.2 *Bottleneck structure of ResNet50/101/152*

Then we will analyze the residual structure for ResNet50/101/152. In this residual structure, the main branch uses three convolutional layers, the first is a 1x1 convolutional layer to compress the channel dimension, the second is a 3x3 convolutional layer, and the third is a 1x1 convolutional layer. The layer is used to restore the channel dimension. The number of convolution kernels used in the first convolution layer and the second convolution layer on the main branch is the same, and the third time is 4 times that of the first layer, hence the name bottleneck model.

4 Experiments

4.1 Datasets

This section showcases the performance and the evaluation process for the following deeper networks: AlexNet, VGG16, GoogLeNet, and ResNet, applied to the two most popular datasets, MNIST and CIFAR-10.

4.1.1 *MNIST*

Yann LeCun et al. provide a widely-used open dataset of handwritten digits called MNIST [9]. It is a solid dataset ,it has a total of 70,000 images divided into 60,000 training images and the remaining 10,000 testing images. All the images are grayscale with a resolution of (28x28). MNIST is already available in torchvision library which can be easily imported in the code, ready to use. Several data augmentation tricks are applied to this dataset based on the compatibility of the network. For example, resizing the images to 32x32 for the VGG network or randomly horizontally flipping the images.

4.1.2 *CIFAR-10*

CIFAR-10 is a smaller subset of a larger dataset which houses about 80 million tiny images. It contains 60,000 small resolution images divided into 50,000 training images and 10,000 testing images that belong to one of 10 classes. The images are 3 channel RGB with a resolution of 32x32. There are 6000 images per class. Examples of classes are cat, dog, car, planes, etc. CIFAR-10 is also readily

4.2 Configurations

After experimenting with different values for each of the below parameters, I found that the below values give the best result:

I used early stopping for MNIST dataset, all the four networks converged before the 20th epochs. Therefore, I decided to train all the networks for 20 epochs.

Table 1: MNIST Parameters

Parameters	Value
Epoch	20
Batch Size	128
Optimizer	Stochastic Gradient Descent (SGD)
Learning Rate	0.01

Table 2: CIFAR-10 Parameters

Parameters	Value
Epoch	50
Batch Size	64
Optimizer	Stochastic Gradient Descent (SGD)
Learning Rate	0.01

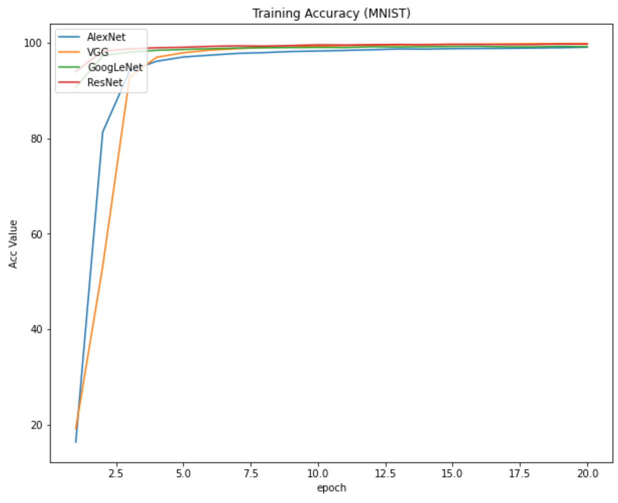


Figure 6: Training Accuracy (MNIST)

Table 3: MNIST results

Model Test Loss	Test Acc	
AlexNet	0.00021	98.20%
VGG	0.00018	98.82%
GoogLeNet	0.00001	99.05%
ResNet	0.00021	99.55%

4.3 Evaluation

According to the figures 6, 7, 8 , and 9 all the models achieve a high accuracy of 98% or higher, with with loss of almost zero. This shows deep networks do actually improve the performance. However, this dataset is well balanced and simple.

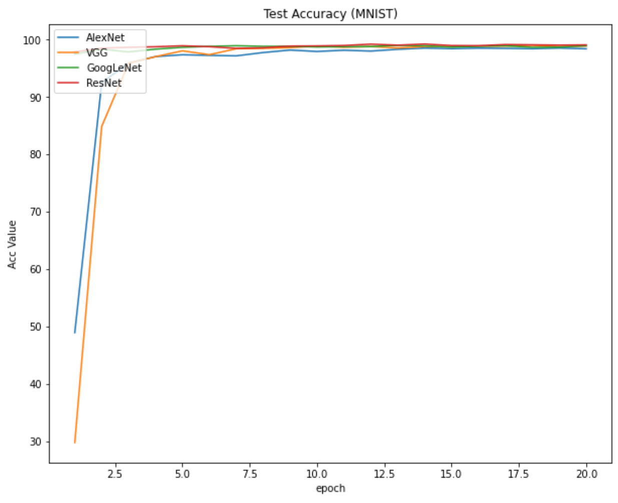


Figure 7: Test Accuracy (MNIST)

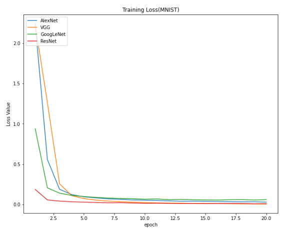


Figure 8: training loss

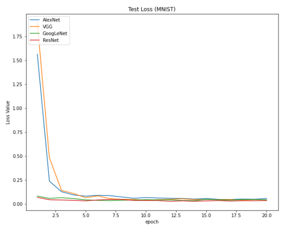


Figure 9: MNIST test loss

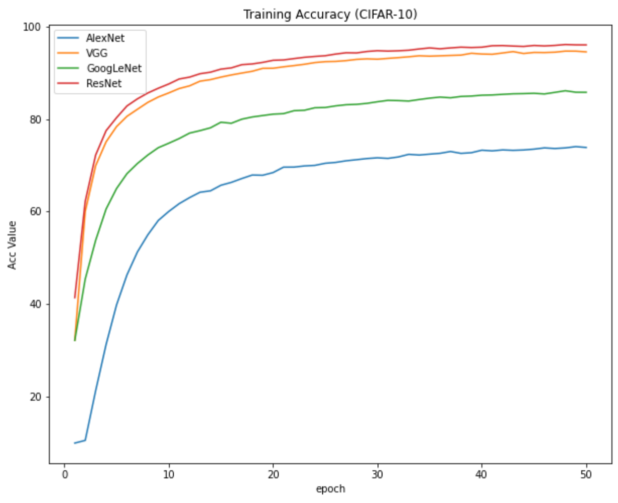


Figure 10: Training Accuracy (CIFAR-10)

According to the figures and the table 4, the accuracy of the VGG and the ResNet is higher than that of AlexNet and GoogLeNet. This makes sense, since each of the network was originally an improvement over the previous network except for GoogLeNet, which was an improvement over VGG. However, on the CIFAR-10 dataset VGG performed better; this is due to the dataset being more complex with RGB channels. For MNIST we do not see much difference in terms of their performance because

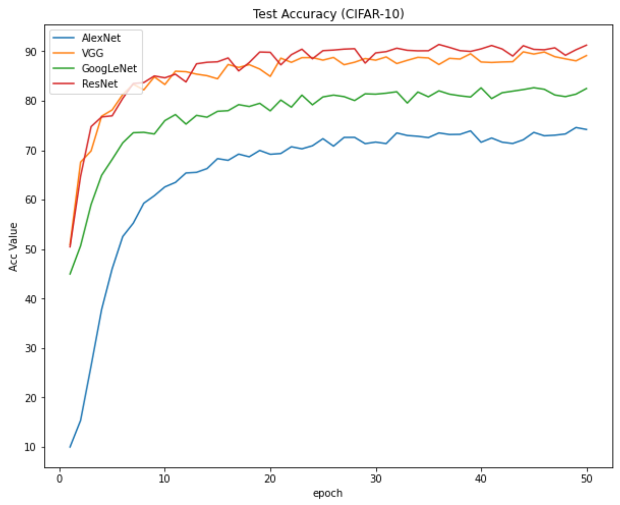


Figure 11: Test Accuracy (CIFAR-10)

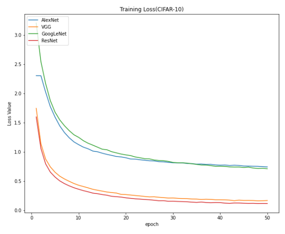


Figure 12: CIFAR training Loss

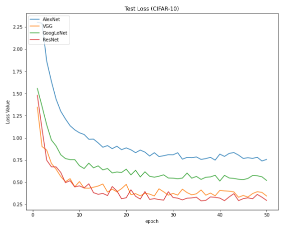


Figure 13: CIFAR test Loss

Table 4: CIFAR-10 results

Model	Test Loss	Test Acc
AlexNet	0.756619	74.20%
VGG	0.344327	89.15%
GoogLeNet	0.519368	82.49%
ResNet	0.294142	91.26%

the simplicty and robustness of the dataset that only have one grayscale channel, and its due to perfectly preproccing that's already made before making it available for the public.

4.4 Incorrectly Predicted Images

The best models among the four was ResNet on both MNIST and CIFAR-10 dataset. Therefore I am mainly focusing on the miss-classified images by ResNet, even though the evalutaion is available for all the datasets in the notebooks. 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. *P.S. Green labels indicate images predicted correctly. Red labels indicated incorrectly predicted images*

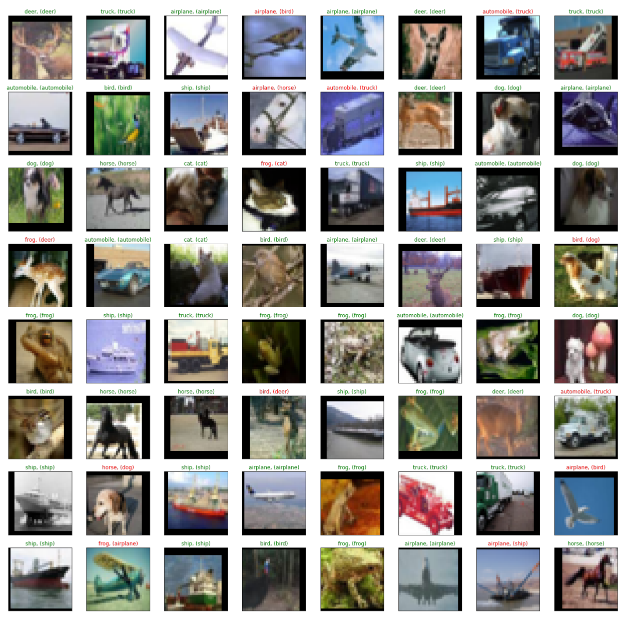


Figure 14: Incorrectly predicted images (ResNet)

According to the confusion matrix in figure 15 most of the mistakes happen in classes cat and dog.

Figure 16 shows accuracy per class for the ResNet network in the CIFAR-10 dataset. Again the lowest accuracy are dog and cat classes.

5 Conclusion

This paper studies the deep networks for image classification. It focuses on the AlexNet, VGG, GoogLeNet and ResNet models, and compares the performance of the four models on two different datasets, namely MNIST and CIFAR10. We understood 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 models in image classification and is a future research direction.

These networks, although large, use the full capacity of CNNs. Deep networks such as these affect the performance and time taken to run the model. Nonetheless, with

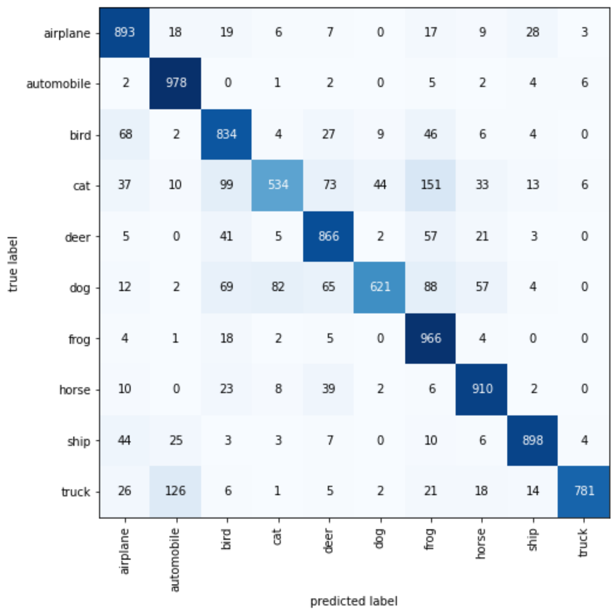
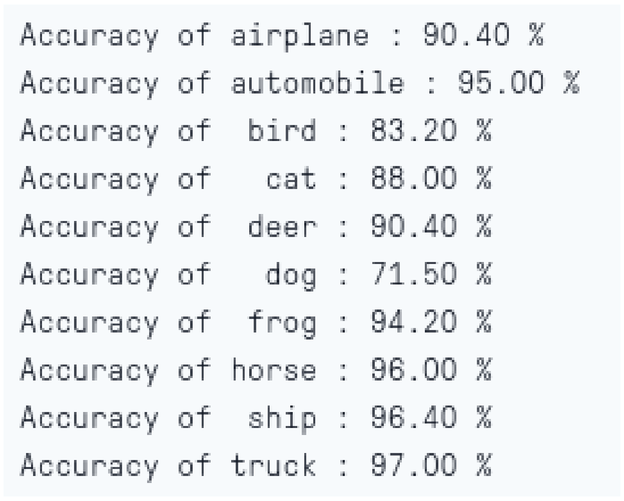


Figure 15: ResNet Confussion Matrix (CIFAR-10)



```
Accuracy of airplane : 90.40 %
Accuracy of automobile : 95.00 %
Accuracy of  bird : 83.20 %
Accuracy of   cat : 88.00 %
Accuracy of  deer : 90.40 %
Accuracy of   dog : 71.50 %
Accuracy of  frog : 94.20 %
Accuracy of horse : 96.00 %
Accuracy of  ship : 96.40 %
Accuracy of truck : 97.00 %
```

Figure 16: Accuracy per class in CIFAR-10

the right architecture and appropriate techniques, the depth of the networks can undoubtedly improve the model's performance. We concluded that ResNet was the best model among the models mentioned above. However, more experiments are needed in the future to improve the research of this paper.

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Advances in neural information processing systems **25** (2012)

[2] K. Simonyan, A. Zisserman, arXiv preprint arXiv:1409.1556 (2014)

[3] Y. LeCun et al., URL: http://yann. lecun. com/exdb/lenet **20**, 14 (2015)

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

[5] Y. Huang, J. Wei, W. Tang, C. He, *Pyramid Convolutional Neural Networks and Bottleneck Residual Modules for Classification of Multispectral Images*, in *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium* (IEEE, 2020), pp. 1949–1952

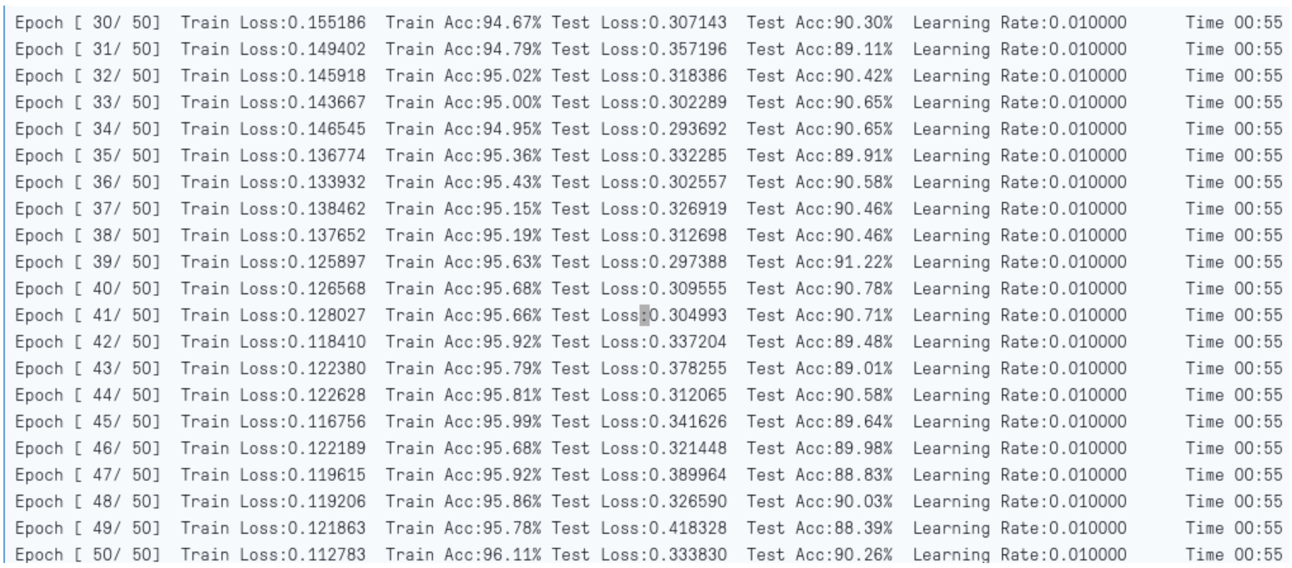
[6] T. Ozturk, M. Talo, E.A. Yildirim, U.B. Baloglu, O. Yildirim, U.R. Acharya, Computers in biology and medicine **121**, 103792 (2020)

[7] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, arXiv preprint arXiv:1704.04861 (2017)

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

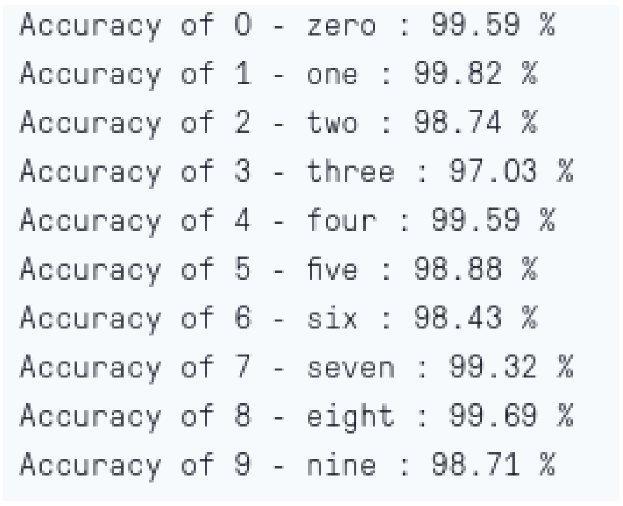
[9] Y. LeCun, http://yann. lecun. com/exdb/mnist/ (1998)

Appendix



```
Epoch [ 30/ 50]  Train Loss:0.155186  Train Acc:94.67% Test Loss:0.307143  Test Acc:90.30%  Learning Rate:0.010000      Time 00:55
Epoch [ 31/ 50]  Train Loss:0.149402  Train Acc:94.79% Test Loss:0.357196  Test Acc:89.11%  Learning Rate:0.010000      Time 00:55
Epoch [ 32/ 50]  Train Loss:0.145918  Train Acc:95.02% Test Loss:0.318386  Test Acc:90.42%  Learning Rate:0.010000      Time 00:55
Epoch [ 33/ 50]  Train Loss:0.143667  Train Acc:95.00% Test Loss:0.302289  Test Acc:90.65%  Learning Rate:0.010000      Time 00:55
Epoch [ 34/ 50]  Train Loss:0.146545  Train Acc:94.95% Test Loss:0.293692  Test Acc:90.65%  Learning Rate:0.010000      Time 00:55
Epoch [ 35/ 50]  Train Loss:0.136774  Train Acc:95.36% Test Loss:0.332285  Test Acc:89.91%  Learning Rate:0.010000      Time 00:55
Epoch [ 36/ 50]  Train Loss:0.133932  Train Acc:95.43% Test Loss:0.302557  Test Acc:90.58%  Learning Rate:0.010000      Time 00:55
Epoch [ 37/ 50]  Train Loss:0.138462  Train Acc:95.15% Test Loss:0.326919  Test Acc:90.46%  Learning Rate:0.010000      Time 00:55
Epoch [ 38/ 50]  Train Loss:0.137652  Train Acc:95.19% Test Loss:0.312698  Test Acc:90.46%  Learning Rate:0.010000      Time 00:55
Epoch [ 39/ 50]  Train Loss:0.125897  Train Acc:95.63% Test Loss:0.297388  Test Acc:91.22%  Learning Rate:0.010000      Time 00:55
Epoch [ 40/ 50]  Train Loss:0.126568  Train Acc:95.68% Test Loss:0.309555  Test Acc:90.78%  Learning Rate:0.010000      Time 00:55
Epoch [ 41/ 50]  Train Loss:0.128027  Train Acc:95.66% Test Loss:0.304993  Test Acc:90.71%  Learning Rate:0.010000      Time 00:55
Epoch [ 42/ 50]  Train Loss:0.118410  Train Acc:95.92% Test Loss:0.337204  Test Acc:89.48%  Learning Rate:0.010000      Time 00:55
Epoch [ 43/ 50]  Train Loss:0.122380  Train Acc:95.79% Test Loss:0.378255  Test Acc:89.01%  Learning Rate:0.010000      Time 00:55
Epoch [ 44/ 50]  Train Loss:0.122628  Train Acc:95.81% Test Loss:0.312065  Test Acc:90.58%  Learning Rate:0.010000      Time 00:55
Epoch [ 45/ 50]  Train Loss:0.116756  Train Acc:95.99% Test Loss:0.341626  Test Acc:89.64%  Learning Rate:0.010000      Time 00:55
Epoch [ 46/ 50]  Train Loss:0.122189  Train Acc:95.68% Test Loss:0.321448  Test Acc:89.98%  Learning Rate:0.010000      Time 00:55
Epoch [ 47/ 50]  Train Loss:0.119615  Train Acc:95.92% Test Loss:0.389964  Test Acc:88.83%  Learning Rate:0.010000      Time 00:55
Epoch [ 48/ 50]  Train Loss:0.119206  Train Acc:95.86% Test Loss:0.326590  Test Acc:90.03%  Learning Rate:0.010000      Time 00:55
Epoch [ 49/ 50]  Train Loss:0.121863  Train Acc:95.78% Test Loss:0.418328  Test Acc:88.39%  Learning Rate:0.010000      Time 00:55
Epoch [ 50/ 50]  Train Loss:0.112783  Train Acc:96.11% Test Loss:0.333830  Test Acc:90.26%  Learning Rate:0.010000      Time 00:55
```

Figure 17: Training ResNet on CIFAR-10 for 50 epochs



```
Accuracy of 0 - zero : 99.59 %
Accuracy of 1 - one : 99.82 %
Accuracy of 2 - two : 98.74 %
Accuracy of 3 - three : 97.03 %
Accuracy of 4 - four : 99.59 %
Accuracy of 5 - five : 98.88 %
Accuracy of 6 - six : 98.43 %
Accuracy of 7 - seven : 99.32 %
Accuracy of 8 - eight : 99.69 %
Accuracy of 9 - nine : 98.71 %
```

Figure 18: Accuracy per Class in MNIST dataset. (ResNet