COMP219 Assignment 2

201375753 Zuosong Han

Check list:

☑Step 1: Loading Data
☑Step 2: Write two CNN models
☑Step 3: Train your CNN models
☑Step 4: Predict with Trained Model
☑Extra

Introduction:

This assignment focuses on implementing two image classifiers with convolutional neural networks. I have implemented LeNet architecture, AlexNet architecture and my own improved architecture. I choose MNIST handwritten dataset which is the most classic dataset to train my models. Firstly, the dataset was downloaded and preprocessed. Then I implemented two traditional CNN models. Finally, I trained and predicted randomly images from dataset. I tried to improve the model to decrease loss and increase accuracy.

Step 1: Loading Data

I selected dataset MNIST handwritten dataset and it would be download automatically when the program was executed. The dataset was split to train dataset, train labels, test dataset and test labels respectively. Then I displayed several details about dataset. There are 60,000 training data and 10,000 testing data. The training labels have 10 classes which are digit 0 to 9. I used "Counter" function to calculate the number of data entries in each classes. The details were shown on Figure 1.

```
Counter({1: 6742, 7: 6265, 3: 6131, 2: 5958, 9: 5949, 0: 5923, 6: 5918, 8: 5851, 4: 5842, 5: 5421})
Training data shape:  (60000, 28, 28)
Training labels shape:  (60000,)
Test data shape (10000, 28, 28)
Test labels shape (10000,)
```

Figure 1: the details of dataset

I used "plot" function to display some of data entries in training dataset. The pictures were shown on Figure 2.

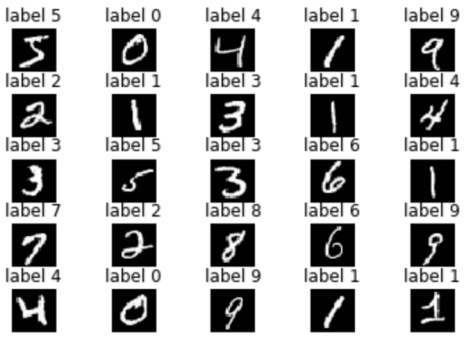


Figure 2: the pictures of training dataset

Step 2: Write two CNN models

I implemented two CNN models. One is LeNet which was one of very first convolutional neural networks created by Yann LeCun in 1988. The other is AlexNet which competed in the ImageNet Large Scale Visual Recognition Challenge in 2012 created by Alex Krizhevsky.

AlexNet

AlexNet architecture contains five convolutional layers and three fully connected layers. Relu is applied after very convolutional and fully connected layer. Dropout is applied before the first and the second fully connected year. On the first convolution layer, there are 96 convolution kernels and the filter's size is 11*11. Then I used max pooling layer to reduce the parameters. The output becomes 13*13*96. On the second convolution layer, the kernels increase to 256. The max pooling size becomes 3*3. The output becomes 6*6*256. The convolution kernels increase to 384 in third and fourth layer but decrease to 256 in fifth layer. The last max pooling function optimizes output to 2*2*256. When doing fully connected, it uses dropout instead of regularisation to deal with overfitting. The AlexNet architecture summary was shown on Figure 3.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 96)	11712
max_pooling2d_1 (MaxPooling2	(None, 13, 13, 96)	0
conv2d_2 (Conv2D)	(None, 13, 13, 256)	614656
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 256)	0
conv2d_3 (Conv2D)	(None, 6, 6, 384)	885120
conv2d_4 (Conv2D)	(None, 6, 6, 384)	1327488
conv2d_5 (Conv2D)	(None, 6, 6, 256)	884992
max_pooling2d_3 (MaxPooling2	(None, 2, 2, 256)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 4096)	4198400
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4096)	16781312
dropout_2 (Dropout)	(None, 4096)	0
dense_3 (Dense)	(None, 10)	40970

Total params: 24,744,650
Trainable params: 24,744,650
Non-trainable params: 0

Figure 3: AlexNet architecture summary

LeNet

LeNet architecture consists of two convolution layers, two max pooling layers, flatten layers, dense layers and activation layers. The input picture's size is 28*28. On the first convolution layer, there are 20 convolution kernels and the filter's size is 5*5. Stride is 1 and padding is same which means it will fill 0 to make sure the output size does not change. So the first output shape is 28*28*20. Then, I used a max pooling layer to reduce the parameters within the module also called sub-sampling. Therefore, the output becomes 14*14*20. The convolution kernels increase to 50 so that the model could extract more features so the output becomes 14*14*50. And it executes max pooling function again. The output become 7*7*50. It will do fully connected in dense layer. Finally, the final layer has digit 0 to 9 and each digit has a value which is the predicted probability. The LeNet architecture summary was shown on Figure 4.

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 28, 28, 20)	520
activation_9 (Activation)	(None, 28, 28, 20)	0
max_pooling2d_5 (MaxPooling2	(None, 14, 14, 20)	0
conv2d_6 (Conv2D)	(None, 14, 14, 50)	25050
activation_10 (Activation)	(None, 14, 14, 50)	0
max_pooling2d_6 (MaxPooling2	(None, 7, 7, 50)	0
flatten_3 (Flatten)	(None, 2450)	0
dense_5 (Dense)	(None, 500)	1225500
activation_11 (Activation)	(None, 500)	0
dense_6 (Dense)	(None, 10)	5010
activation_12 (Activation)	(None, 10)	0

Total params: 1,256,080
Trainable params: 1,256,080
Non-trainable params: 0

Figure 4: LeNet architecture summary

Step 3: Train your CNN models:

At the beginning, it will check whether there is a pre-trained model or not. If the folder exists a pre-trained model, the program will skip the training part and run the evaluating part directly. If it does not exist save model, the first step is using “reshape” function to adjust training dataset and testing dataset. It also reduces images from the range [0, 255] to [0, 1.0] which it a simply scaling technique. Then it will initialise the optimizer which is Stochastic Gradient Descent (SGD) with a learning rate of 0.01. Categorical cross-entropy will be used as loss function, a fairly standard choice when working with datasets that have more than two class labels. The training network will use “fit” method to instantiated a model ad train for 20 epochs. The training results of LeNet architecture and AlexNet architecture were shown on Figure 5.

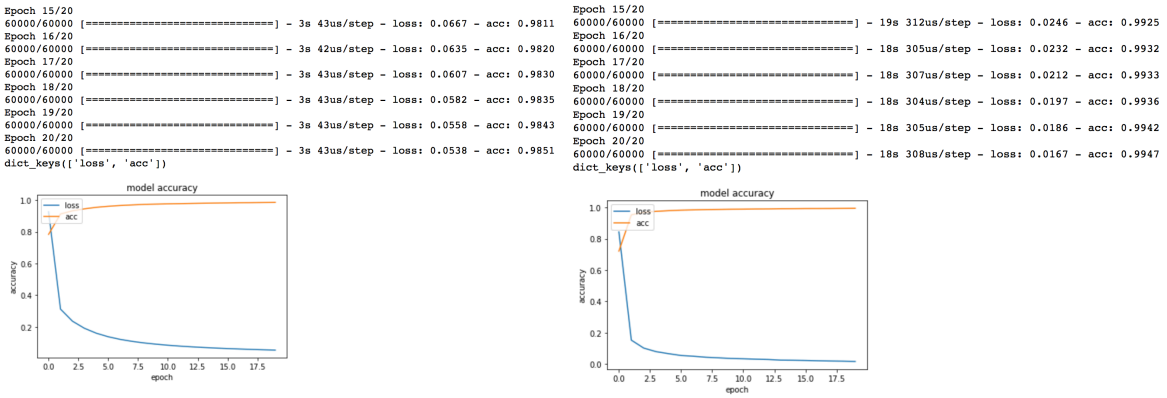


Figure 5: The training results of LeNet(left) and AlexNet(right)

The result shows that the accuracy of AlexNet (99.47%) higher than LeNet (98.51%). The loss of AlexNet is lower than LeNet.

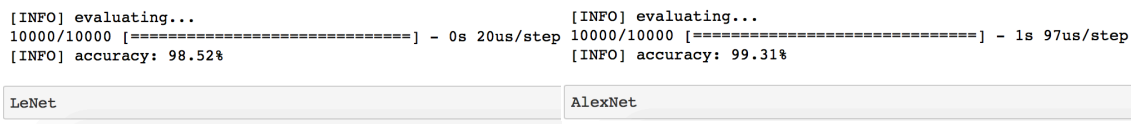


Figure 6: The testing results of LeNet(left) and AlexNet(right)

The testing results of LeNet and AlexNet were shown on Figure 6. In testing part, the score of AlexNet (99.31%) is still higher than LeNet (98.52%). Therefore, Alex performance is better than LeNet in MNIST handwritten dataset.

Step 4: Predict with Trained Model

In the last part of my code, firstly, I used “random” method to choose 10 random number form testing dataset. These digits are passed into network for classification. Then, I used “cv2” package to merge the channels into one image because the origin data is displayed by metrics. I used “resize” function to resize the image from a 28*28 image to a 96*96 image so that we could see it better. And then I set the prediction results shown on the top left of the images. Finally, I printed out the images on the screen. The testing results of LeNet and AlexNet were shown on Figure 7.

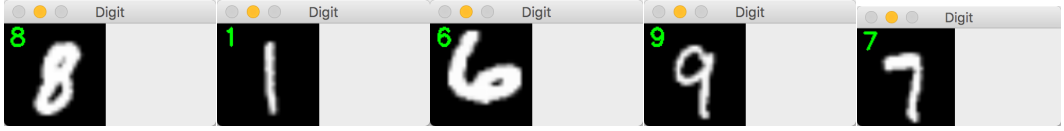


Figure 7: the predicted result of random digits

Extra

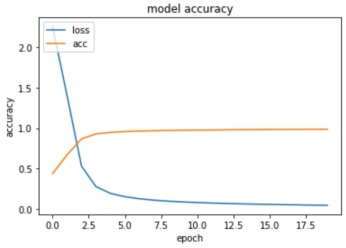
From the above of these two network, I found that the more convolutional layers could have better performance in predicting. And more convolution kernels could have higher score as well. However, the more convolution layers and kernels, the more calculations for computers. It is impossible to train a complex network for student like me who use a not high performance notebook. How could I find the balance between simple model and high accuracy? I noticed that the convolution kernels control the features of images. And max pooling decides the image resolution. So I extract more features on the first convolution layers and make a bigger size on first max pooling. The improved model called HanNet summary was shown on Figure 8.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 28, 28, 360)       9360
_________________________________________________________________
activation_10 (Activation)   (None, 28, 28, 360)       0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 360)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 64)          576064
_________________________________________________________________
activation_11 (Activation)   (None, 7, 7, 64)          0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 3, 3, 36)          57636
_________________________________________________________________
activation_12 (Activation)   (None, 3, 3, 36)          0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 1, 1, 36)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 36)                0
_________________________________________________________________
dense_5 (Dense)              (None, 500)               18500
_________________________________________________________________
activation_13 (Activation)   (None, 500)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                5010
_________________________________________________________________
activation_14 (Activation)   (None, 10)                0
=================================================================
Total params: 666,570
Trainable params: 666,570
Non-trainable params: 0
```

Figure 8: HanNet architecture summary

HanNet architecture contains three convolutional layers and two fully connected layers. However, the parameters are less than LeNet and AlexNet. And it saved some time to train dataset.

```
Epoch 15/20
60000/60000 [==============================] - 7s 113us/step - loss: 0.0620 - acc: 0.9819
Epoch 16/20
60000/60000 [==============================] - 7s 112us/step - loss: 0.0588 - acc: 0.9830
Epoch 17/20
60000/60000 [==============================] - 7s 112us/step - loss: 0.0557 - acc: 0.9838
Epoch 18/20
60000/60000 [==============================] - 7s 112us/step - loss: 0.0531 - acc: 0.9845
Epoch 19/20
60000/60000 [==============================] - 7s 112us/step - loss: 0.0502 - acc: 0.9858
Epoch 20/20
60000/60000 [==============================] - 7s 112us/step - loss: 0.0483 - acc: 0.9860
dict_keys(['loss', 'acc'])
```




```
[INFO] evaluating...
10000/10000 [==============================] - 1s 54us/step
[INFO] accuracy: 98.61%
```

```
HanNet
```

Figure 9: The training results of HanNet

Figure 9 shows that HanNet has a pretty good performance. the accuracy up to 98.6% which is higher than LeNet but lower than AlexNet. It is an available model as an image classifier.