Report Fashion Classification

Zemin XU

Exercise 1:In order to have a graphical description of the Fashion MNIST dataset display the first 9 images existent in the trainX variable using the OpenCV library.

Answer:

By using the “imshow” function in opencv we can visualize an image easily.

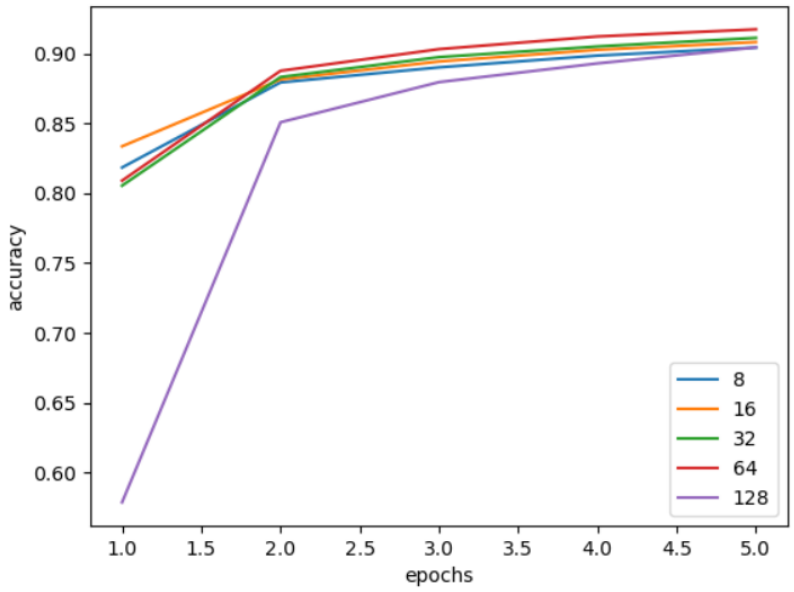
```
#TODO - Application 1 - Step 2 - Load the Fashion MNIST dataset in Keras
(trainX, trainY), (testX, testY) = fashion_mnist.load_data()

for i in range(9):
    cv2.imshow(str(i), trainX[i])
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exercise 2: Modify the number of filters in the convolutional layer as specified in Table 1. How is the system accuracy influenced by this parameter? How about the convergence time?

Answer:

Table 1. System performance evaluation for various numbers of filters

Number of filters	8	16	32	64	128
System accuracy	0.9043	0.9082	0.9112	0.9173	0.9044



```
the average time for each epoch in case:  8  of neuron is:  8.316433715820313
the average time for each epoch in case:  16  of neuron is:  8.424666166305542
the average time for each epoch in case:  32  of neuron is:  11.74468502998352
the average time for each epoch in case:  64  of neuron is:  23.012382745742798
the average time for each epoch in case:  128  of neuron is:  45.520407915115356
```

The graph above shows the change of accuracy with increase of epoch for each case. We can see that, for case from 8 to 64, after the 3rd epoch, the lines become flat, which signs convergence. However, for the case of 128, it starts at a rather low accuracy and it converges after the 4th epoch. When this parameter ranges from 8 to 64, increasing it results in a better accuracy.

Generally, this parameter does not influence the accuracy, because the differences of accuracy are less than 0.01.

In terms of the convergence time, as the console output above, the convergence time is multiplied by 3 for the cases from 8 to 64. The case of 128 uses 4 epochs which are 182.08s in total, which is 628% more than the case of 8 and 16, 164% more than the case of 64.

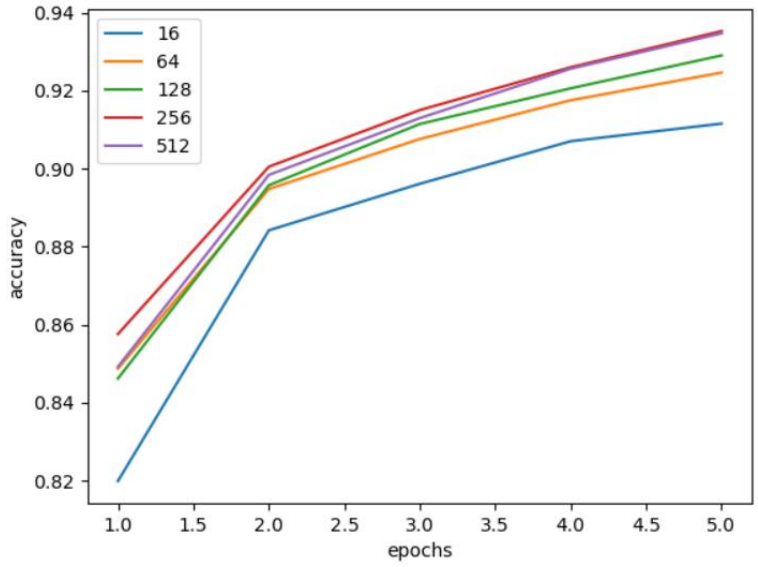
Exercise 3: Modify the number of neurons in the dense hidden layer as specified in Table 2. The number of filters in the convolutional layer remains set to 32. What can be observed regarding the system accuracy?

Answer:

Table 2. System performance evaluation for various numbers of neurons in the dense hidden layer

No of neurons	16	64	128	256	512
System accuracy	0.9116	0.9246	0.9290	0.9353	0.9347

The graph below shows the change of accuracy with increase of epoch for each case. We can see that, for case from 16 to 256, the accuracy becomes better when the parameter increases. The case of 512 is the same as that of 256 but takes more time.



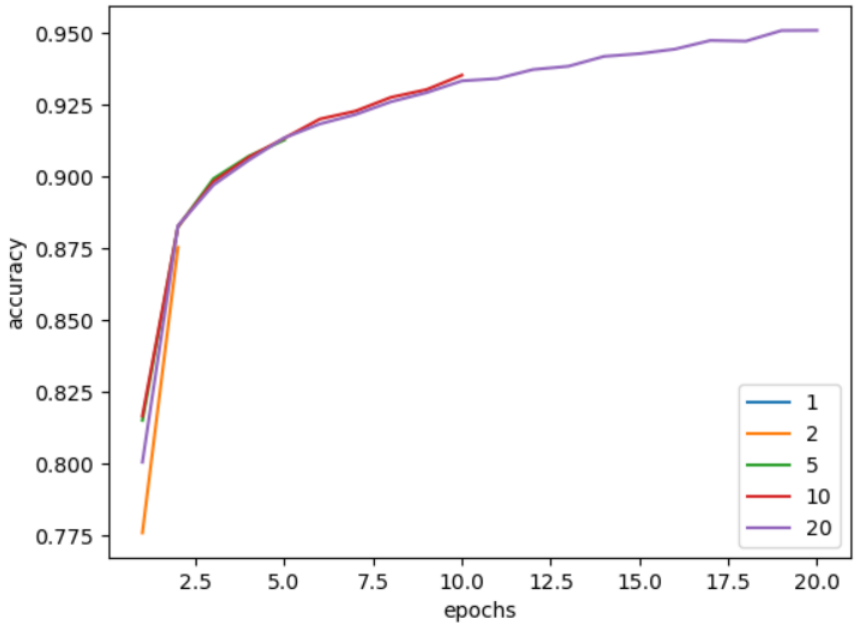
Exercise 4: Modify the number of epochs used to train the model as specified in Table 3. The number of neurons in the dense hidden layer is set to 16. What can be observed regarding the system accuracy? How about the convergence time?

Answer:

Table 3. System performance evaluation for different values of the number of epochs

No of epochs	1	2	5	10	20
System accuracy	0.8245	0.8753	0.9127	0.9353	0.9509

From the graph below, we can see that the accuracy keeps improving while number of epochs increases, and the increase rate is coming down. In terms of the convergence time, according to the graph, we cannot decide. But if we look at the answer of exercise 6, we can find that after 4th epoch the overfitting starts. With the time it takes we can know the convergence time easily.



Exercise 5: Modify the learning rate of the stochastic gradient descend as presented in Table 4. Train the model for 5 epochs. What can be observed regarding the system accuracy and the convergence time?

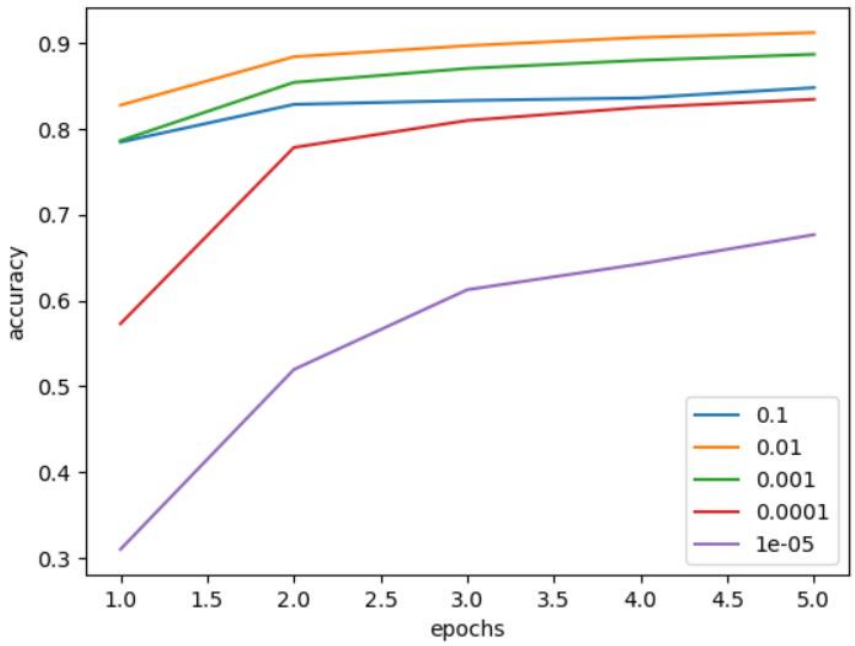
Answer:

Table 4. System performance evaluation for various learning rates of the SGD

Learning rate	0.1	0.01	0.001	0.0001	0.00001
System accuracy	0.84	0.91	0.88	0.83	0.67

As we can see from the graph and console output below, the case of 0.01 brings the best accuracy among them. The average time for an epoch for each case is very close to 12.5s. However, the case from 0.1 to 0.0001 converges after the 2nd epoch, and the convergence time is about 25s. The case of 0.00001 cannot converge within 5 epochs.

```
the average time for each epoch in case:  0.1  of neuron is:  12.355079841613769
the average time for each epoch in case:  0.01  of neuron is:  12.428230237960815
the average time for each epoch in case:  0.001  of neuron is:  12.586997652053833
the average time for each epoch in case:  0.0001  of neuron is:  12.354510879516601
the average time for each epoch in case:  1e-05  of neuron is:  13.026374101638794
```



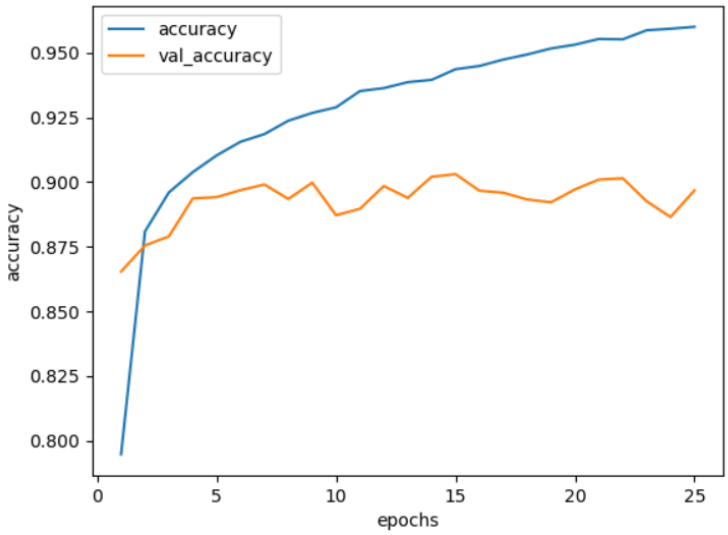
Exercise 6: Explore how adding regularization impacts model performance as compared to the baseline model. Add a Dropout layer on the CNN (after the MaxPoolinglayer) that randomly excludes 20% of neurons (model.add(Dropout(0.2))). Select for the learning rate a value of 0.01. Analyze the convergence speed with and without this regularization operation?

Modify the percentage of excluded neurons in the dropout layer as presented in Table 5. Select a value of 3x3 neurons for convolutional kernel. How is the system accuracy influenced by this parameter?

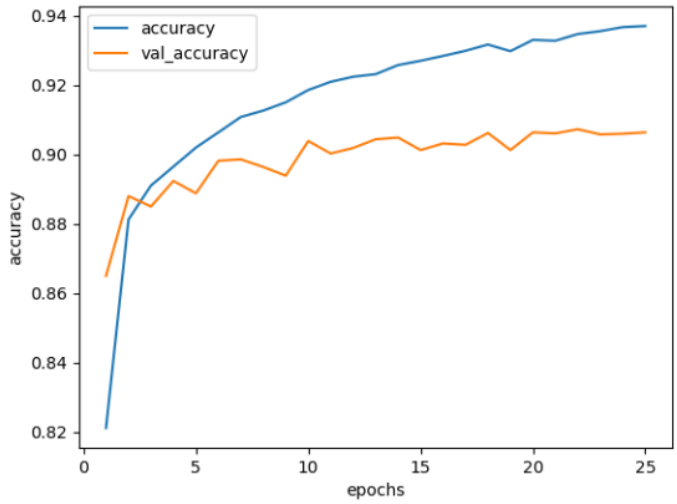
Answer:

Table 5. System performance evaluation for number of neurons

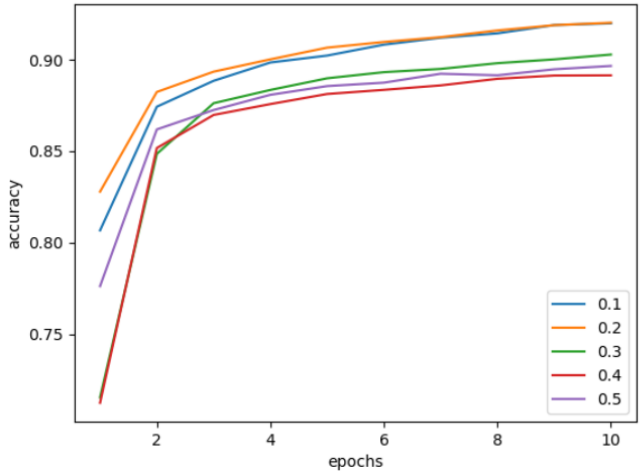
Dropped percentage	0.1	0.2	0.3	0.4	0.5
System accuracy	0.9202	0.9210	0.9029	0.8914	0.8966



First, I check the epoch after which overfitting[1] starts. From the graph above (without dropout layer) we can know that after 4th epoch the overfitting starts. We can also see that, without dropout layer, the accuracy at convergence is about 0.90.



Second, I check the overfitting for a dropout layer with rate of 0.2. From the graph above we can know that after 10th epoch the overfitting starts. Therefore, I will set epoch number as 10 for testing different values as dropout rate, in order to compare the accuracy. The best one is 0.2 whose value is close to that of 0.1.



Exercise 7: Specify the optimal values for the following parameters (the number of filters in the convolutional layer, the size of the convolutional kernel, the number of neurons in the dense hidden layer, the number of epochs used for training, the learning rate) that maximize the system accuracy.

Answer:

From the experiments above, we can get the optimal value for these parameters:

Number of filters in convolutional layer: 64

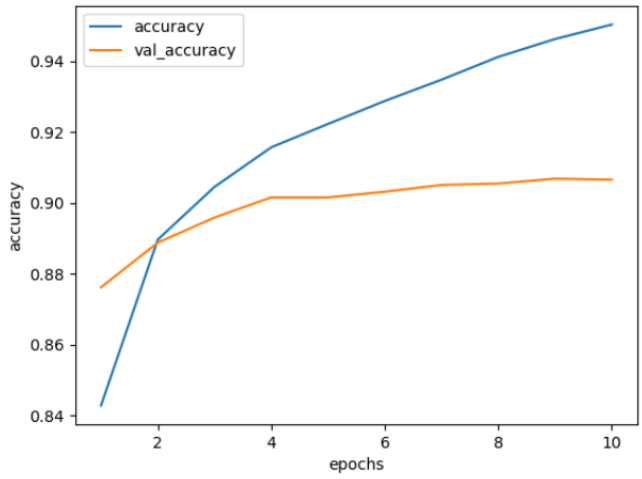
Number of neurons in the dense hidden layer: 256

Number of epochs used for training: 4

Learning rate: 0.01

Size of convolutional kernel: 7*7

The last parameter is what I test and obtain from 3*3 to 9*9.



Exercise 8: Once fit, we can save the final model (architecture and weights) to an *.h5 file by calling the save()function on the model and pass in the selectedfilename(model.save('Fashion_MNIST_model.h5')). We can use our saved model to make a prediction on new images.

Using the pre-trained model (saved above) write a Python script able to make an automatic prediction regarding the category of the image presented in Fig.1. For this example, we expect class "2" that corresponds to a "Pullover".

Answer:

I implement a function to parse the image and do prediction like below:

```python
def load_and_test(sample_path):
    # change img to grayscale
    sample = cv2.imread(sample_path, cv2.IMREAD_GRAYSCALE)

    # resize to 28 * 28
    sample = cv2.resize(sample, (28,28))

    # reshape to 4 dimension
    sample = sample.reshape((-1, 28, 28, 1))

    model = keras.models.load_model("Fashion_MNIST_model.h5")
    pred = model.predict(sample)
    print(pred)
```

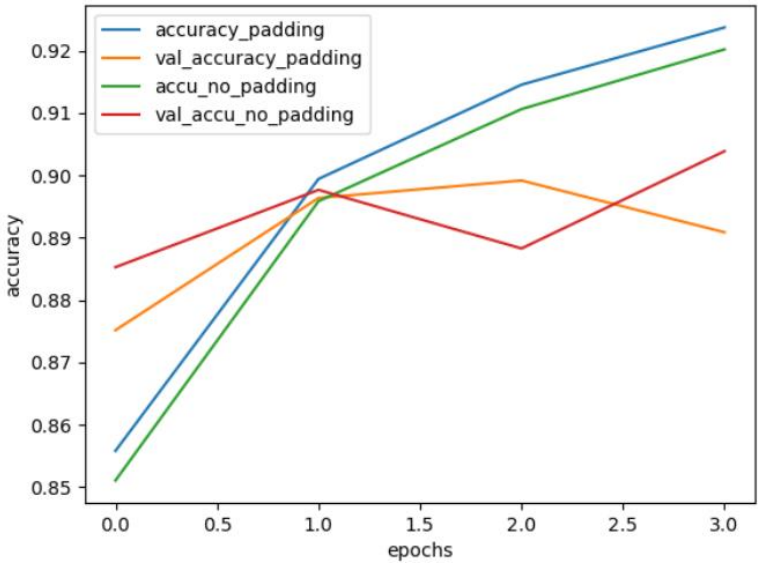
And I get the prediction result as class "2"(start from 0), which is the correct answer.

```
[[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]]
```

Exercise 9: Adding padding to the convolutional operation can often result in better model performance, as more of the input image of feature maps are given an opportunity to participate or contribute to the output. By default, the convolutional operation uses 'valid' padding, which means that convolutions are only applied where possible. This can be changed to padding='same' so that zero values are added around the input such that the output has the same size as the input. For Application 1, how is the system accuracy influenced by the padding operation?

Answer:

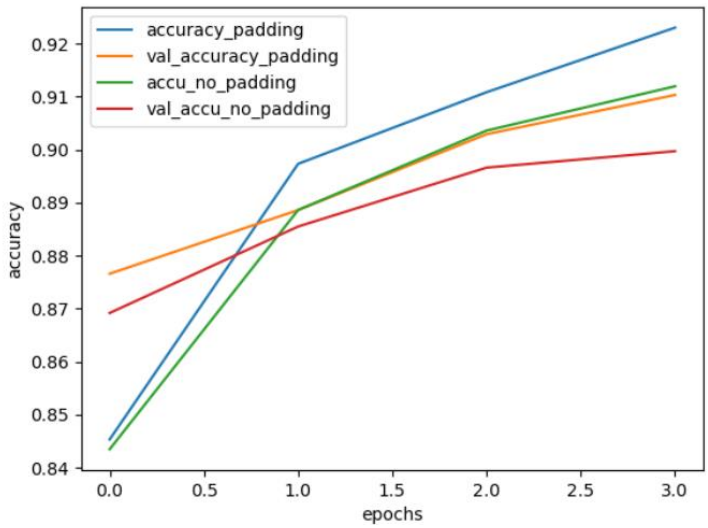
As we can see from the graph below, by comparing val_accuracy, after the 1st epoch, the option without padding is in overfitting, but the option with padding is still in underfitting. The system accuracy can be influenced by avoiding overfitting. The code can be found in "exercise3_9.py".



Exercise 10: An increase in the number of filters used in the convolutional layer can often improve performance, as it can provide more opportunity for extracting simple features from the input images. This is especially relevant when very small filters are used, such as 3×3 pixels. By applying the padding operation (padding='same') within the convolutional process, increase the number of filters (in the convolutional layer) from 32 to double that at 64. For Application 1, how is the system accuracy influenced by this parameter?

Answer:

From the graph below, we can see that the system accuracy is better with padding and small filters. Compared to that without padding, it gains higher accuracy because the small filters can detect simple details, with the help of more neurons, it can fit better if there is no other convolutional layer. What is more, the padding ensures that each little detail to be captured by filters.



References:

[1]. Convolutional neural network 3: convnets and overfitting » AI Geek Programmer