ANLY 535 Keras_CNN Report

– Xiaowei Wan

This report is to describe Convolutional Neoral Networks(CNN) in Python with Keras, and how to overcome overfitting with dropout.

Phase 1: Run the deep, feed-forward artificial neural networks in Keras.

1. Load data from Keras Fashion-MNIST dataset, including 10 category, each category has 70,000 images.
2. Split dataset into Training dataset(60,000) and Testing dataset (10,000)

```
('Training data shape : ', (60000, 28, 28), (60000,))
('Testing data shape : ', (10000, 28, 28), (10000,))
```

3. Analyze dataset – rescale the image pixels and resize the images.
4. Data processing

-- Convert 28 x 28 image of the train and test set into a matrix of size 28 x 28 x 1 to fed into the network.

```
((60000, 28, 28, 1), (10000, 28, 28, 1))
```

-- convert data type to float32 from int8.

```
('Original label:', 9)
('After conversion to one-hot:', array([0., 0., 0., 0., 0., 0., 0., 0.,
0., 1.], dtype=float32))
```

-- convert class labels into a one-hot encoding vector.

-- Separate training dataset into two parts – one part for training(80%), the other part for validation(20%).

5. Model data – import necessary modules to train model
6. Compile Model – optimize created model.
7. Train Model – Use Kera's fit() function to train model. Observing the training accuracy and loss.
8. Evaluate Model performance on Test Set
9. Plot to see training and validation accuracy, and training and validation loss. Observe that probably overfitting.

```
('Test loss:', 0.4190925658617169)
('Test accuracy:', 0.9181)
```

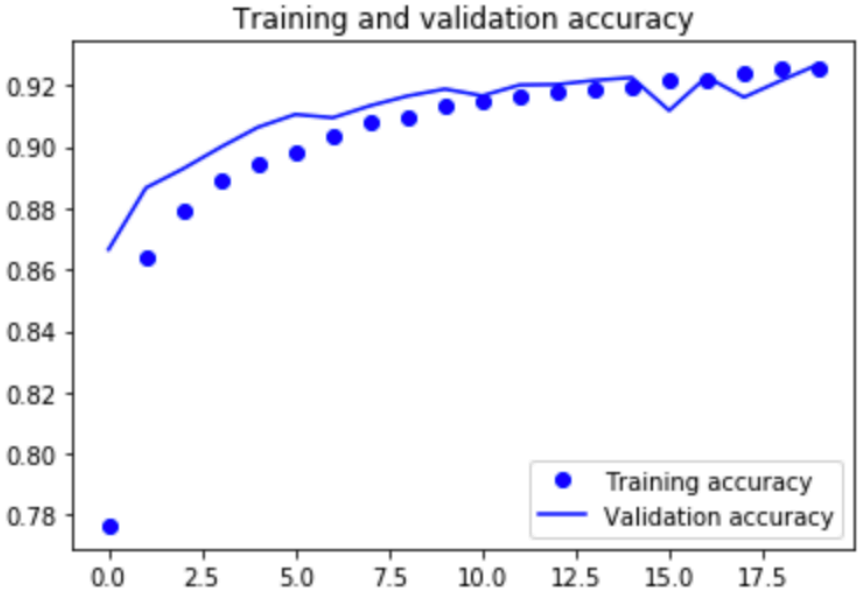
Fashion model summary:

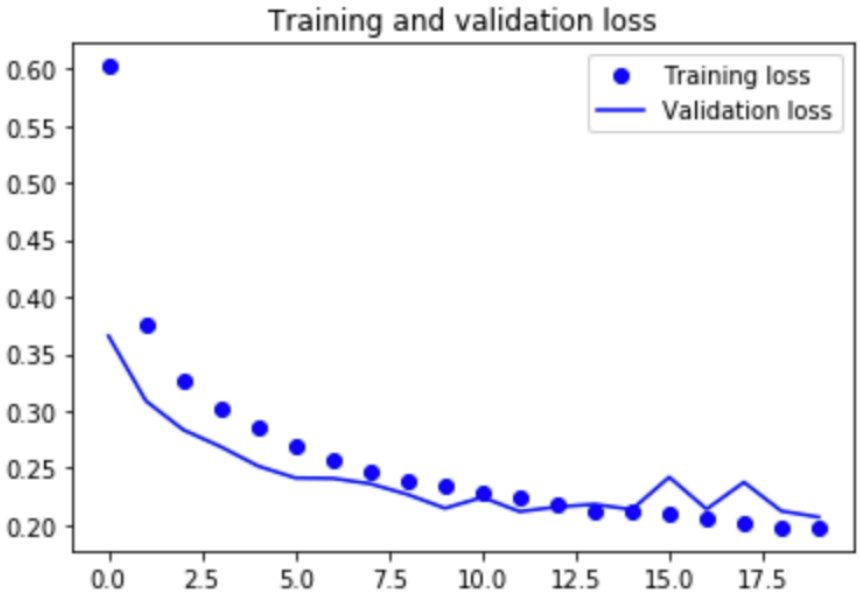
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 28, 28, 32)        320
_________________________________________________________________
leaky_re_lu_13 (LeakyReLU)   (None, 28, 28, 32)        0
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 64)        18496
_________________________________________________________________
leaky_re_lu_14 (LeakyReLU)   (None, 14, 14, 64)        0
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 7, 7, 128)         73856
_________________________________________________________________
leaky_re_lu_15 (LeakyReLU)   (None, 7, 7, 128)         0
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_7 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               262272
_________________________________________________________________
leaky_re_lu_16 (LeakyReLU)   (None, 128)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
=================================================================
Total params: 356,234
Trainable params: 356,234
Non-trainable params: 0
_________________________________________________________________
```

from plots we can tell that the validation accuracy almost became stagnant after 4-5 epochs and rarely increased at certain epochs.

10. Solve overfitting problem – introducing Dropout into the Network
11. Adding Dropout, re-compile and re-train the network.
12. RE-evaluate model on the test set, plots the training and validation accuracy, training and validation loss plots, observe that it is improved the performance than previous model.

```
('Test loss:', 0.217214672338962 55)
('Test accuracy:', 0.9225)
```

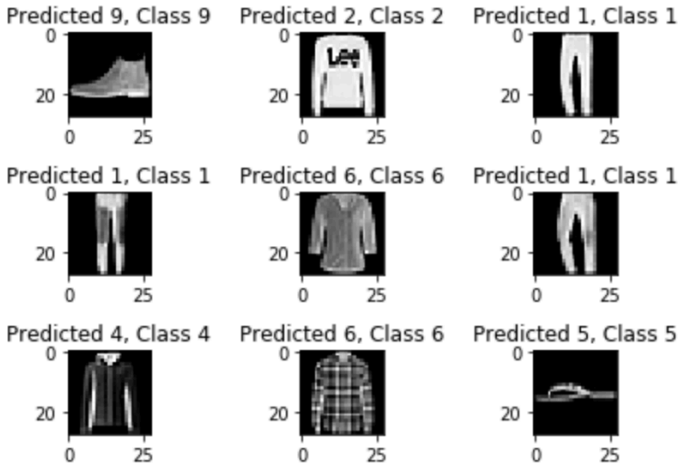




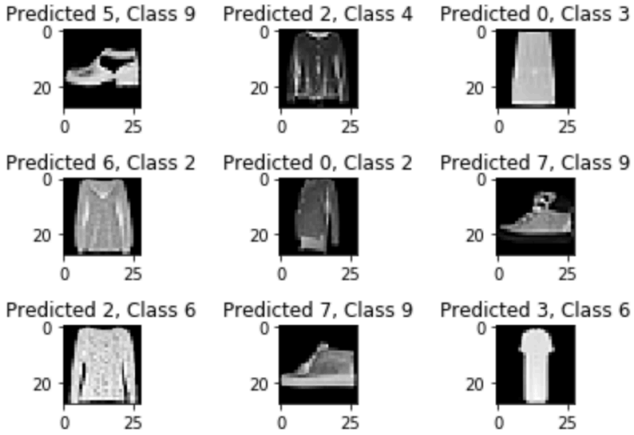
12. If performance is still bad and overfitting, need to re-compile and re-train the model couple more times to make sure the performance become better fit.

13. Predict labels

```
Found 9189 correct labels
```




```
Found 811 incorrect labels
```



14. Classification report, help to identify the misclassified classes in more detail.

	precision	recall	f1-score	support
Class 0	0.78	0.90	0.83	1000
Class 1	0.99	0.99	0.99	1000
Class 2	0.88	0.89	0.89	1000
Class 3	0.95	0.90	0.92	1000
Class 4	0.88	0.88	0.88	1000
Class 5	0.99	0.99	0.99	1000
Class 6	0.82	0.72	0.77	1000
Class 7	0.95	0.98	0.97	1000
Class 8	0.98	0.99	0.98	1000
Class 9	0.98	0.96	0.97	1000
avg / total	0.92	0.92	0.92	10000

Phase 2: Solve the overfitting problem.

From above Phase 1 step 11 and step 12, introducing Dropout into network, and the performance get better from training and validation accuracy, training and validation loss plots.

Phase 3: This phase has an extra point. You need to improve the model to get better performance and accuracy.

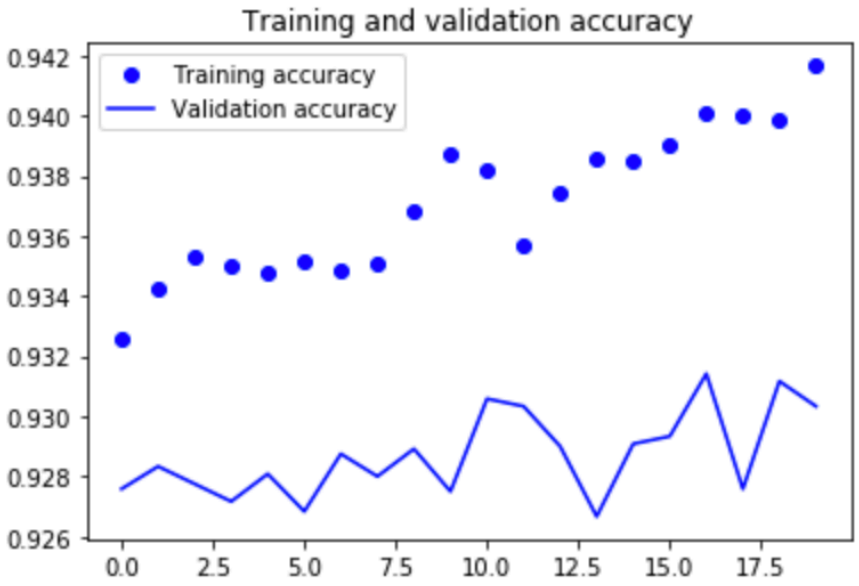
Backpropagation – By updating weights and biases on the cost from previous computation. Using Adam optimizer for backpropagation.

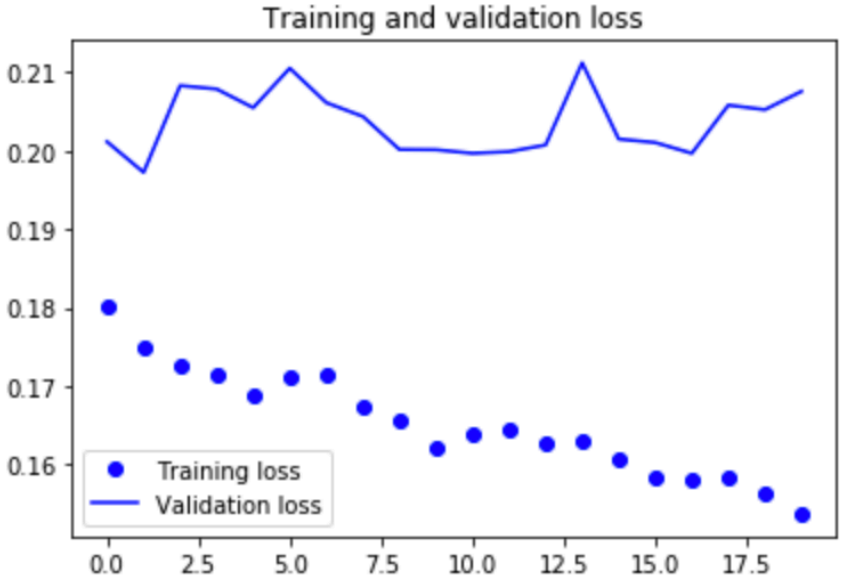
Re-compile and re-train the model by change the weights and biases to improve the model. After changed the batch size and other parameters, Performance is improved.

```
('Test loss:', 0.22278614352494477)
('Test accuracy:', 0.9254)
```

Test accuracy is 92.54%. It is improved a bit, not much, and we can run many times, the accuracy should be improved more.

Plots:


Training and validation accuracy
Training accuracy
Validation accuracy
0.942
0.940
0.938
0.936
0.934
0.932
0.930
0.928
0.926
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Training and validation loss
Training loss
Validation loss
0.21
0.20
0.19
0.18
0.17
0.16
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5