

MID-TERM EXAM

Deep Learning

PROFESSOR PARK SANGUK

Salaki Reynaldo Joshua
Student Number: 202228511



SALAKI REYNALDO JOSHUA

Description of assignment:

Announce the report (mid-term exam) related to the deep learning class. Please try implementing the "Image Classification Using Convolutional Neural Network" model, which was conducted in the 6th week, 3rd class with colab.

However, please try to implement a convolutional neural network using input image data other than "fashion_mnist" data. Try implementing an image classification model using "mnist" data, which is numeric handwritten data, or other image data.

Please attach the captured photo implemented in coLab to a HWP or Word file and submit it to IRURI system.

(International students may come to my office and discuss with me if have any questions.)

thank you. :)

Due Date : 2022-11-04 00:00

DEEP LEARNING CNN FOR FASHION-MNIST CLOTHING CLASSIFICATION

The Fashion-MNIST clothing classification problem is a new standard dataset used in computer vision and deep learning.

Although the dataset is relatively simple, it can be used as the basis for learning and practicing how to develop, evaluate, and use deep convolutional neural networks for image classification from scratch. This includes how to develop a robust test harness for estimating the performance of the model, how to explore improvements to the model, and how to save the model and later load it to make predictions on new data.

In this Mid-Term Exam (Project), the Author will discover how to develop a convolutional neural network for clothing classification from scratch.

The objectives of this project:

- How to develop a test harness to develop a robust evaluation of a model and establish a baseline of performance for a classification task.
- How to explore extensions to a baseline model to improve learning and model capacity.
- How to develop a finalized model, evaluate the performance of the final model, and use it to make predictions on new images.

FASHION MNIST CLOTHING CLASSIFICATION

1

FASHION MNIST CLOTHING CLASSIFICATION

The Fashion-MNIST dataset is proposed as a more challenging replacement dataset for the MNIST dataset.

It is a dataset comprised of 60,000 small square 28×28 pixel grayscale images of items of 10 types of clothing, such as shoes, t-shirts, dresses, and more. The mapping of all 0-9 integers to class labels is listed below.

- 0: T-shirt/top
- 1: Trouser
- 2: Pullover
- 3: Dress
- 4: Coat
- 5: Sandal
- 6: Shirt
- 7: Sneaker
- 8: Bag
- 9: Ankle boot

It is a more challenging classification problem than MNIST and top results are achieved by deep learning convolutional neural networks with a classification accuracy of about 90% to 95% on the hold-out test dataset.



+ Code + Text

RAM  Disk 

Editing

```
1 # assignment salaki reynaldo joshua loading the fashion mnist dataset
2 from matplotlib import pyplot
3 from keras.datasets import fashion_mnist
4 # load dataset
5 (trainX, trainy), (testX, testy) = fashion_mnist.load_data()
6 # summarize loaded dataset
7 print('Train: X=%s, y=%s' % (trainX.shape, trainy.shape))
8 print('Test: X=%s, y=%s' % (testX.shape, testy.shape))
9 # plot first few images
10 for i in range(9):
11     # define subplot
12     pyplot.subplot(330 + 1 + i)
13     # plot raw pixel data
14     pyplot.imshow(trainX[i], cmap=pyplot.get_cmap('gray'))
15 # show the figure
16 pyplot.show()
```

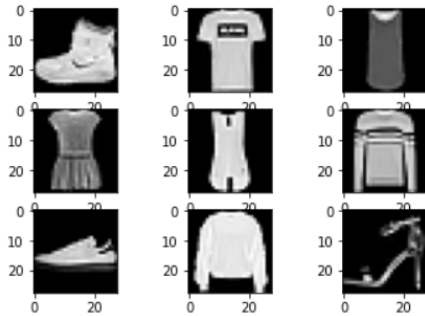


+ Code + Text

RAM  Disk 

Editing

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [=====] - 0s 0us/step
Train: X=(60000, 28, 28), y=(60000,)
Test: X=(10000, 28, 28), y=(10000,)
```

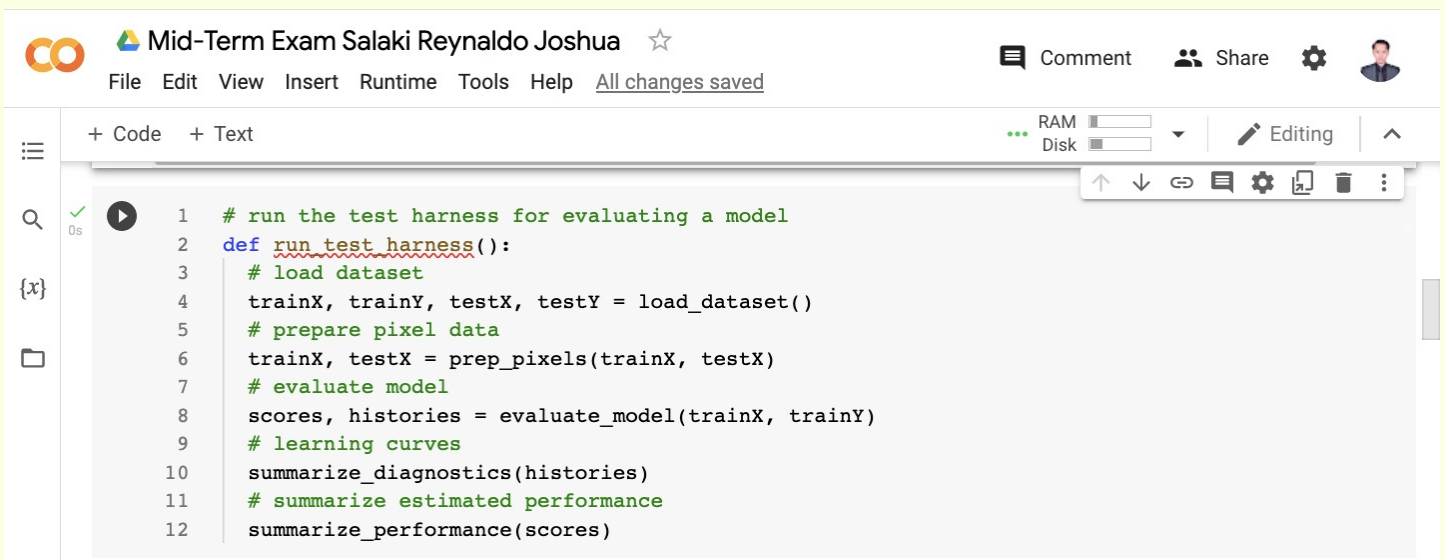


COMPLETE PROJECT

2

COMPLETE PROJECT

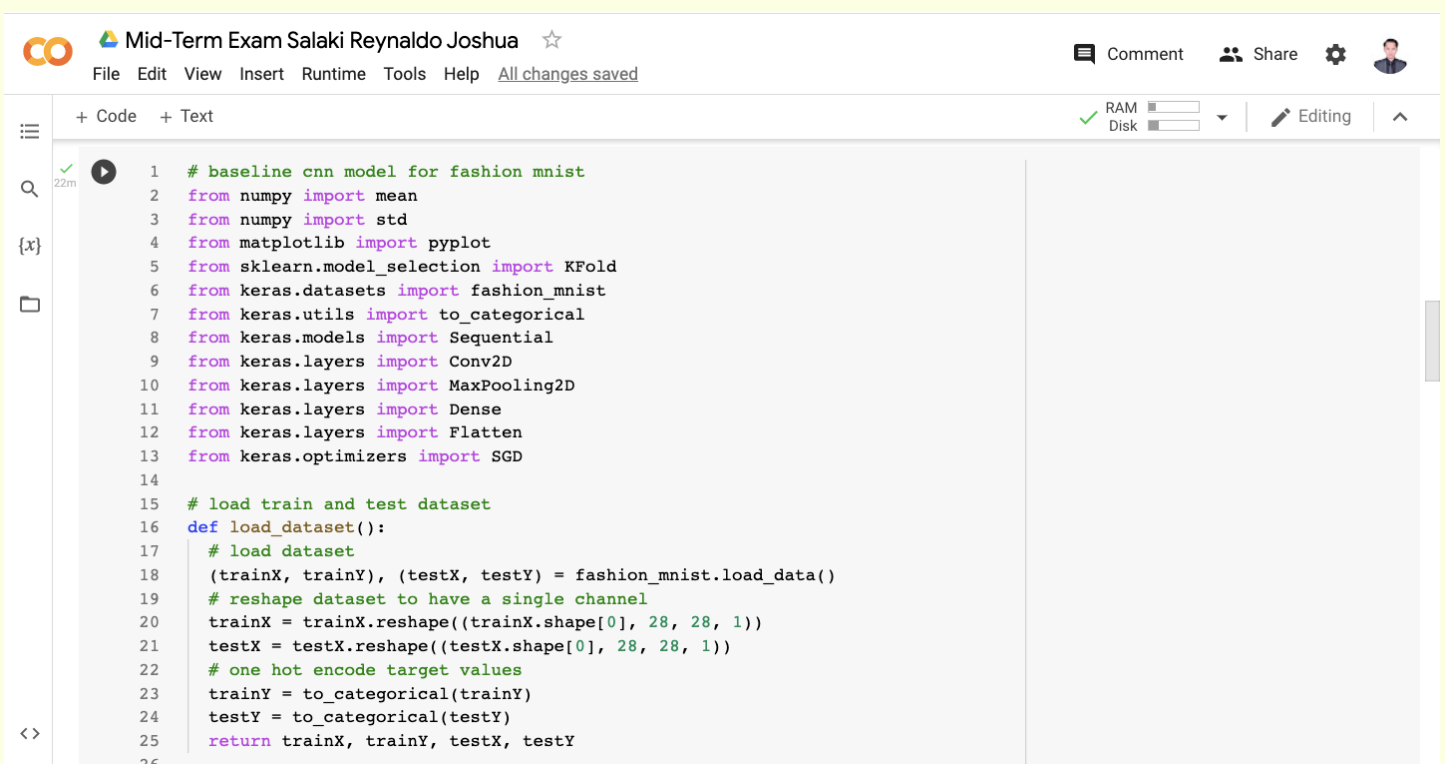
We need a function that will drive the test harness. This involves calling all of the defined functions.



The screenshot shows a code editor interface with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar (All changes saved). The code is written in Python and defines a function named `run_test_harness()`. The function performs the following steps: load dataset, prepare pixel data, evaluate model, learning curves, summarize diagnostics, summarize estimated performance, and summarize performance scores.

```
1 # run the test harness for evaluating a model
2 def run_test_harness():
3     # load dataset
4     trainX, trainY, testX, testY = load_dataset()
5     # prepare pixel data
6     trainX, testX = prep_pixels(trainX, testX)
7     # evaluate model
8     scores, histories = evaluate_model(trainX, trainY)
9     # learning curves
10    summarize_diagnostics(histories)
11    # summarize estimated performance
12    summarize_performance(scores)
```

The complete code example for a baseline convolutional neural network model on the MNIST dataset is listed below.



The screenshot shows a code editor interface with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar (All changes saved). The code is written in Python and defines a function named `load_dataset()`. The function performs the following steps: load train and test dataset, reshape dataset to have a single channel, one hot encode target values, and return trainX, trainY, testX, testY.

```
1 # baseline cnn model for fashion mnist
2 from numpy import mean
3 from numpy import std
4 from matplotlib import pyplot
5 from sklearn.model_selection import KFold
6 from keras.datasets import fashion_mnist
7 from keras.utils import to_categorical
8 from keras.models import Sequential
9 from keras.layers import Conv2D
10 from keras.layers import MaxPooling2D
11 from keras.layers import Dense
12 from keras.layers import Flatten
13 from keras.optimizers import SGD
14
15 # load train and test dataset
16 def load_dataset():
17     # load dataset
18     (trainX, trainY), (testX, testY) = fashion_mnist.load_data()
19     # reshape dataset to have a single channel
20     trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
21     testX = testX.reshape((testX.shape[0], 28, 28, 1))
22     # one hot encode target values
23     trainY = to_categorical(trainY)
24     testY = to_categorical(testY)
25     return trainX, trainY, testX, testY
26
```


CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment

Share

Settings

Profile

+ Code + Text

RAM Disk Editing

22m

26

27 # scale pixels

28 def prep_pixels(train, test):

29 # convert from integers to floats

30 train_norm = train.astype('float32')

31 test_norm = test.astype('float32')

32 # normalize to range 0-1

33 train_norm = train_norm / 255.0

34 test_norm = test_norm / 255.0

35 # return normalized images

36 return train_norm, test_norm

37

38 # define cnn model

39 def define_model():

40 model = Sequential()

41 model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))

42 model.add(MaxPooling2D((2, 2)))

43 model.add(Flatten())

44 model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))

45 model.add(Dense(10, activation='softmax'))

46 # compile model

47 opt = SGD(lr=0.01, momentum=0.9)

48 model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

49 return model

50

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment

Share

Settings

Profile

+ Code + Text

RAM Disk Editing

22m

50

51 # evaluate a model using k-fold cross-validation

52 def evaluate_model(dataX, dataY, n_folds=5):

53 scores, histories = list(), list()

54 # prepare cross validation

55 kfold = KFold(n_folds, shuffle=True, random_state=1)

56 # enumerate splits

57 for train_ix, test_ix in kfold.split(dataX):

58 # define model

59 model = define_model()

60 # select rows for train and test

61 trainX, trainY, testX, testY = dataX[train_ix], dataY[train_ix], dataX[test_ix], dataY[test_ix]

62 # fit model

63 history = model.fit(trainX, trainY, epochs=10, batch_size=32, validation_data=(testX, testY), verbose=0)

64 # evaluate model

65 _, acc = model.evaluate(testX, testY, verbose=0)

66 print('> %.3f' % (acc * 100.0))

67 # append scores

68 scores.append(acc)

69 histories.append(history)

70 return scores, histories

71

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share ⚙️ 👤

+ Code + Text

✓ 22m

71

72 # plot diagnostic learning curves

73 def summarize_diagnostics(histories):

74 for i in range(len(histories)):

75 # plot loss

76 pyplot.subplot(211)

77 pyplot.title('Cross Entropy Loss')

78 pyplot.plot(histories[i].history['loss'], color='blue', label='train')

79 pyplot.plot(histories[i].history['val_loss'], color='orange', label='test')

80 # plot accuracy

81 pyplot.subplot(212)

82 pyplot.title('Classification Accuracy')

83 pyplot.plot(histories[i].history['accuracy'], color='blue', label='train')

84 pyplot.plot(histories[i].history['val_accuracy'], color='orange', label='test')

85 pyplot.show()

86

87 # summarize model performance

88 def summarize_performance(scores):

89 # print summary

90 print('Accuracy: mean=%.3f std=%.3f, n=%d' % (mean(scores)*100, std(scores)*100, len(scores)))

91 # box and whisker plots of results

92 pyplot.boxplot(scores)

93 pyplot.show()

94

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share ⚙️ 👤

+ Code + Text

✓ 22m

94

95 # run the test harness for evaluating a model

96 def run_test_harness():

97 # load dataset

98 trainX, trainY, testX, testY = load_dataset()

99 # prepare pixel data

100 trainX, testX = prep_pixels(trainX, testX)

101 # evaluate model

102 scores, histories = evaluate_model(trainX, trainY)

103 # learning curves

104 summarize_diagnostics(histories)

105 # summarize estimated performance

106 summarize_performance(scores)

107

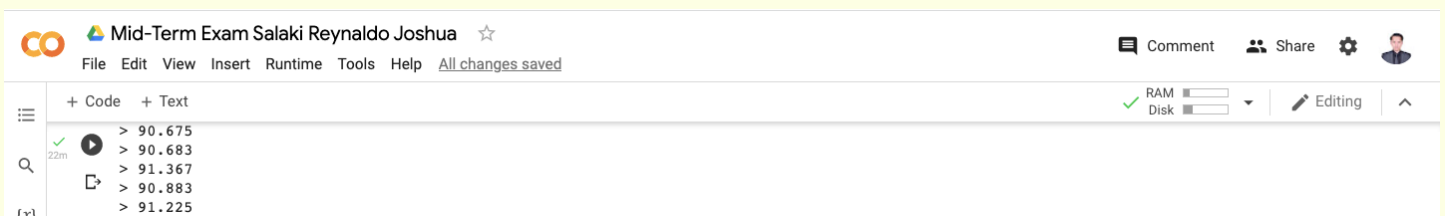
108 # entry point, run the test harness

109 run_test_harness()

RUNNING RESULT

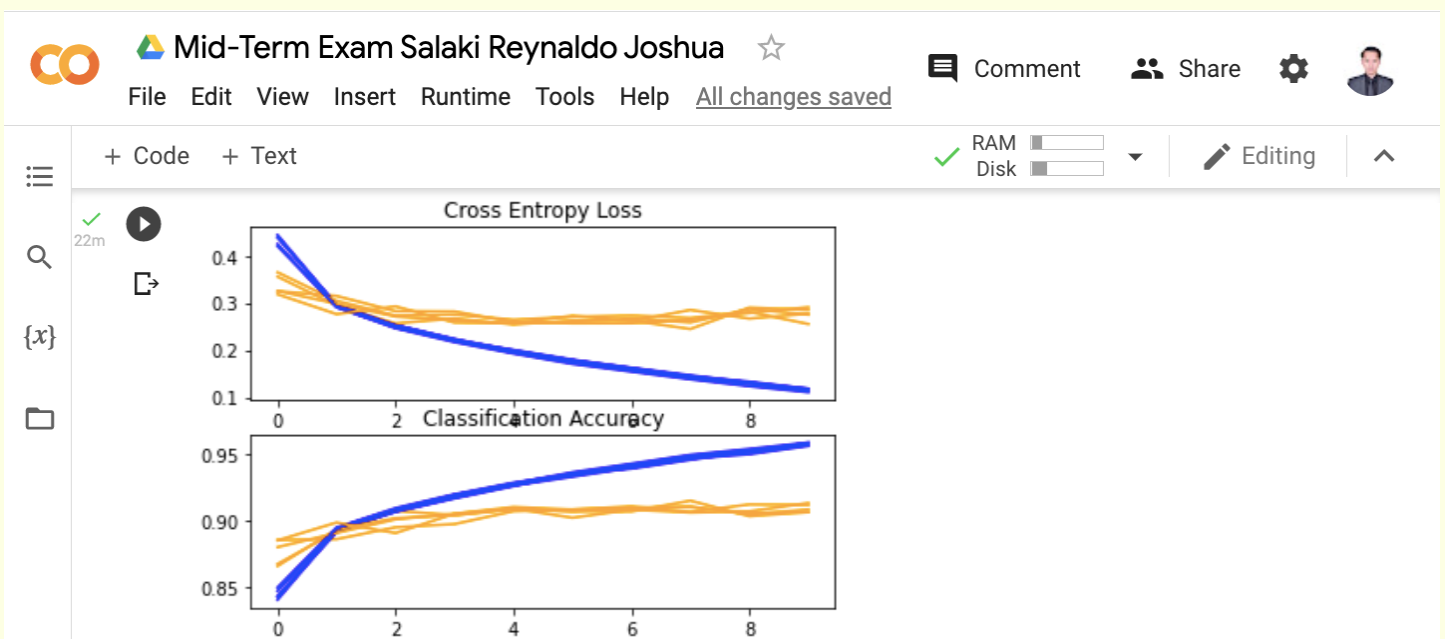
Running the classification accuracy for each fold of the cross-validation process. This is helpful to get an idea that the model evaluation is progressing.

We can see that for each fold, the baseline model achieved an error rate below 10%, and in two cases 98% and 99% accuracy. These are good results.



Next, a diagnostic plot is shown, giving insight into the learning behavior of the model across each fold.

In this case, we can see that the model generally achieves a good fit, with train and test learning curves converging. There may be some signs of slight overfitting.



Next, the summary of the model performance is calculated. We can see in this case, the model has an estimated skill of about 96%, which is impressive.

CO Mid-Term Exam Salaki Reynaldo Joshua ☆

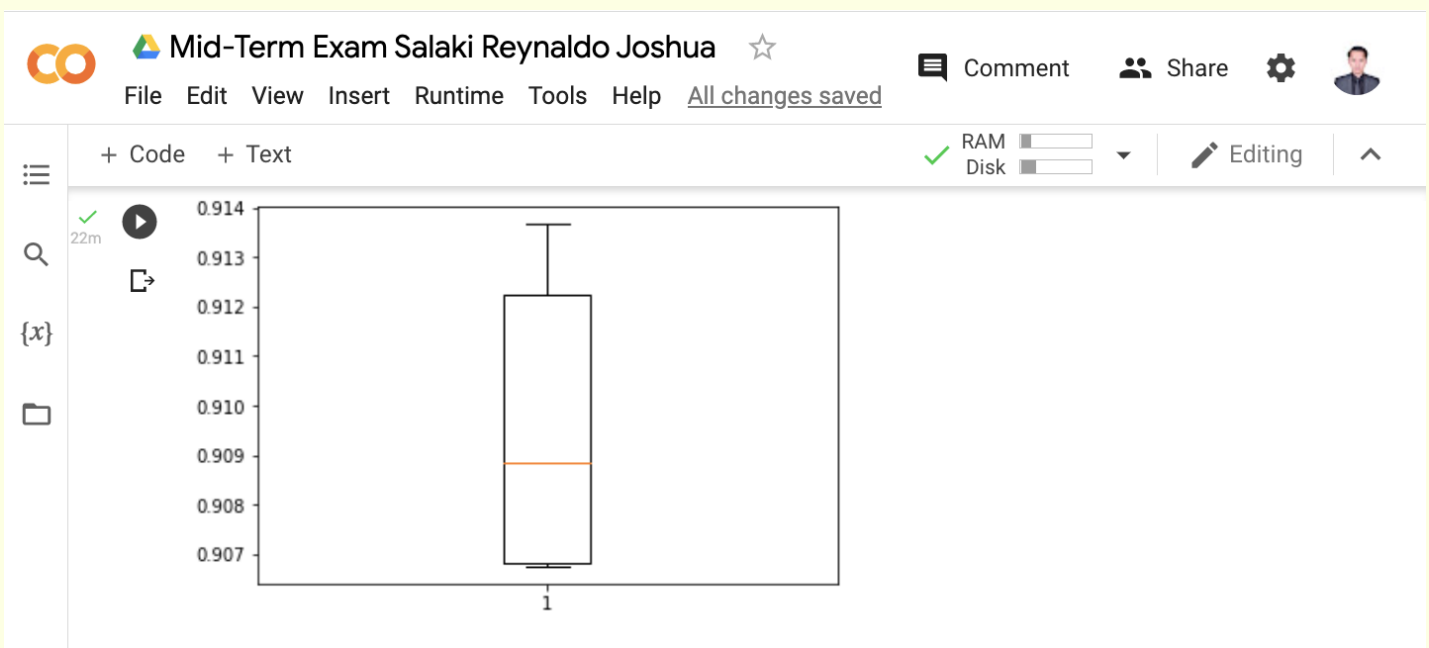
File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

✓ RAM Disk Editing

✓ Accuracy: mean=90.967 std=0.282, n=5

Finally, a box and whisker plot is created to summarize the distribution of accuracy scores.



As we would expect, the distribution spread across the low-nineties. We now have a robust test harness and a well-performing baseline model.

PADDING CONVOLUTIONS

3

DEVELOP AN IMPROVED MODEL

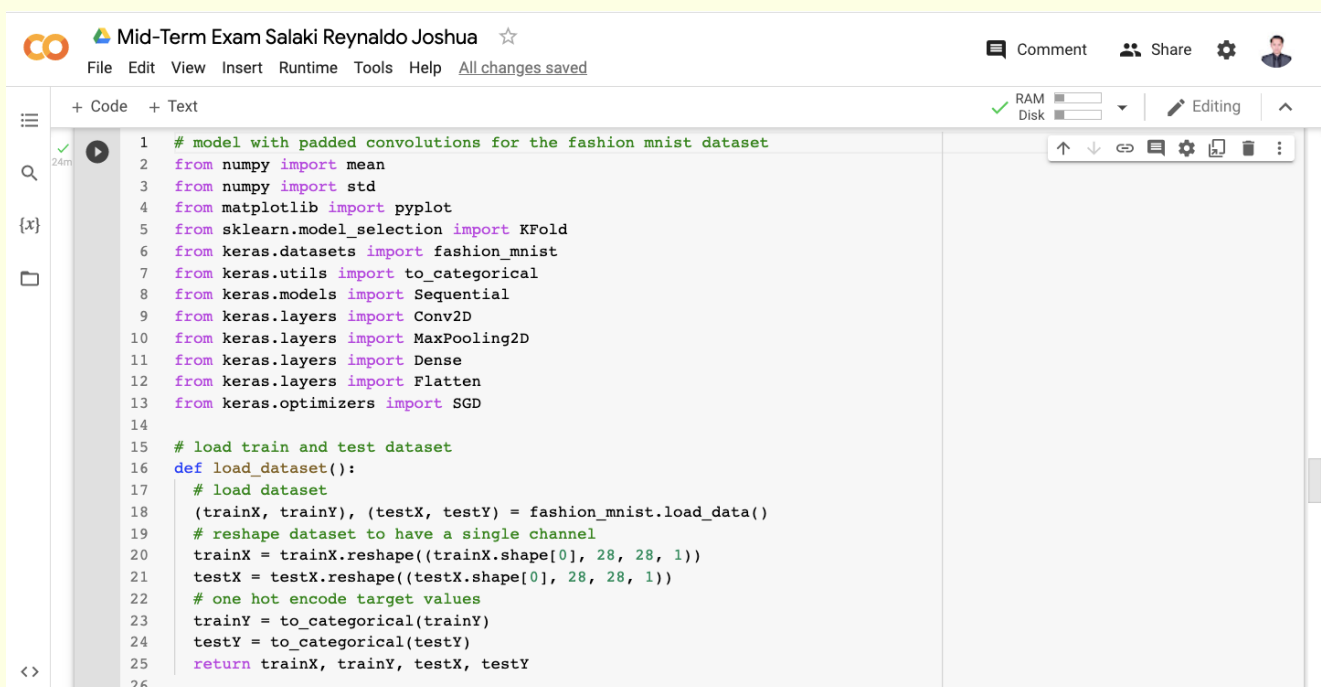
There are many ways that we might explore improvements to the baseline model.

We will look at areas that often result in an improvement, so-called low-hanging fruit. The first will be a change to the convolutional operation to add padding and the second will build on this to increase the number of filters.

Padding Convolutions

Adding padding to the convolutional operation can often result in better model performance, as more of the input image or 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 'same' padding so that zero values are added around the input such that the output has the same size as the input.



```
1 # model with padded convolutions for the fashion mnist dataset
2 from numpy import mean
3 from numpy import std
4 from matplotlib import pyplot
5 from sklearn.model_selection import KFold
6 from keras.datasets import fashion_mnist
7 from keras.utils import to_categorical
8 from keras.models import Sequential
9 from keras.layers import Conv2D
10 from keras.layers import MaxPooling2D
11 from keras.layers import Dense
12 from keras.layers import Flatten
13 from keras.optimizers import SGD
14
15 # load train and test dataset
16 def load_dataset():
17     # load dataset
18     (trainX, trainY), (testX, testY) = fashion_mnist.load_data()
19     # reshape dataset to have a single channel
20     trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
21     testX = testX.reshape((testX.shape[0], 28, 28, 1))
22     # one hot encode target values
23     trainY = to_categorical(trainY)
24     testY = to_categorical(testY)
25     return trainX, trainY, testX, testY
26
```

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings User

+ Code + Text

RAM Disk Editing

26

27 # scale pixels

28 def prep_pixels(train, test):

29 # convert from integers to floats

30 train_norm = train.astype('float32')

31 test_norm = test.astype('float32')

32 # normalize to range 0-1

33 train_norm = train_norm / 255.0

34 test_norm = test_norm / 255.0

35 # return normalized images

36 return train_norm, test_norm

37

38 # define cnn model

39 def define_model():

40 model = Sequential()

41 model.add(Conv2D(32, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))

42 model.add(MaxPooling2D((2, 2)))

43 model.add(Flatten())

44 model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))

45 model.add(Dense(10, activation='softmax'))

46 # compile model

47 opt = SGD(lr=0.01, momentum=0.9)

48 model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

49 return model

50

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings User

+ Code + Text

RAM Disk Editing

51 # evaluate a model using k-fold cross-validation

52 def evaluate_model(dataX, dataY, n_folds=5):

53 scores, histories = list(), list()

54 # prepare cross validation

55 kf = KFold(n_folds, shuffle=True, random_state=1)

56 # enumerate splits

57 for train_ix, test_ix in kf.split(dataX):

58 # define model

59 model = define_model()

60 # select rows for train and test

61 trainX, trainY, testX, testY = dataX[train_ix], dataY[train_ix], dataX[test_ix], dataY[test_ix]

62 # fit model

63 history = model.fit(trainX, trainY, epochs=10, batch_size=32, validation_data=(testX, testY), verbose=0)

64 # evaluate model

65 _, acc = model.evaluate(testX, testY, verbose=0)

66 print('> %.3f' % (acc * 100.0))

67 # append scores

68 scores.append(acc)

69 histories.append(history)

70 return scores, histories

71

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings User

+ Code + Text

RAM Disk Editing

72 # plot diagnostic learning curves

73 def summarize_diagnostics(histories):

74 for i in range(len(histories)):

75 # plot loss

76 pyplot.subplot(211)

77 pyplot.title('Cross Entropy Loss')

78 pyplot.plot(histories[i].history['loss'], color='blue', label='train')

79 pyplot.plot(histories[i].history['val_loss'], color='orange', label='test')

80 # plot accuracy

81 pyplot.subplot(212)

82 pyplot.title('Classification Accuracy')

83 pyplot.plot(histories[i].history['accuracy'], color='blue', label='train')

84 pyplot.plot(histories[i].history['val_accuracy'], color='orange', label='test')

85 pyplot.show()

86

87 # summarize model performance

88 def summarize_performance(scores):

89 # print summary

90 print('Accuracy: mean=%.3f std=%.3f, n=%d' % (mean(scores)*100, std(scores)*100, len(scores)))

91 # box and whisker plots of results

92 pyplot.boxplot(scores)

93 pyplot.show()

94



+ Code + Text

✓ RAM
Disk

Editing



24m

{x}



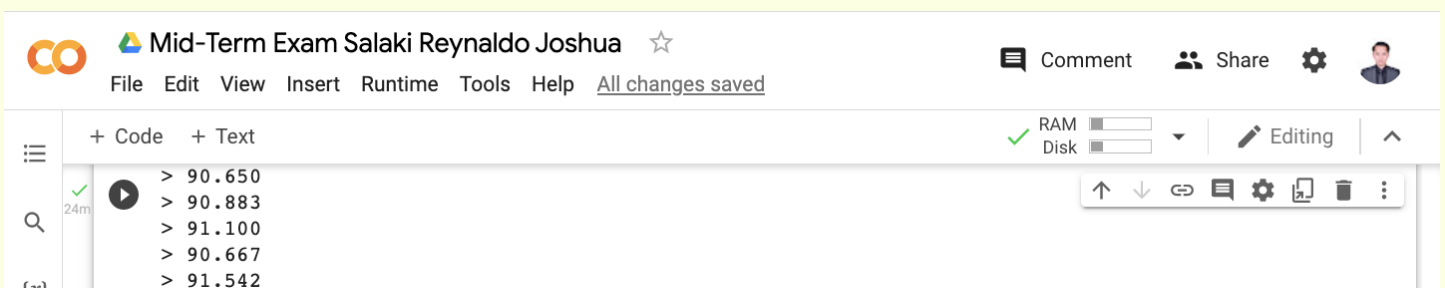
```
95 # run the test harness for evaluating a model
96 def run_test_harness():
97     # load dataset
98     trainX, trainY, testX, testY = load_dataset()
99     # prepare pixel data
100     trainX, testX = prep_pixels(trainX, testX)
101     # evaluate model
102     scores, histories = evaluate_model(trainX, trainY)
103     # learning curves
104     summarize_diagnostics(histories)
105     # summarize estimated performance
106     summarize_performance(scores)
107
108 # entry point, run the test harness
109 run_test_harness()
```



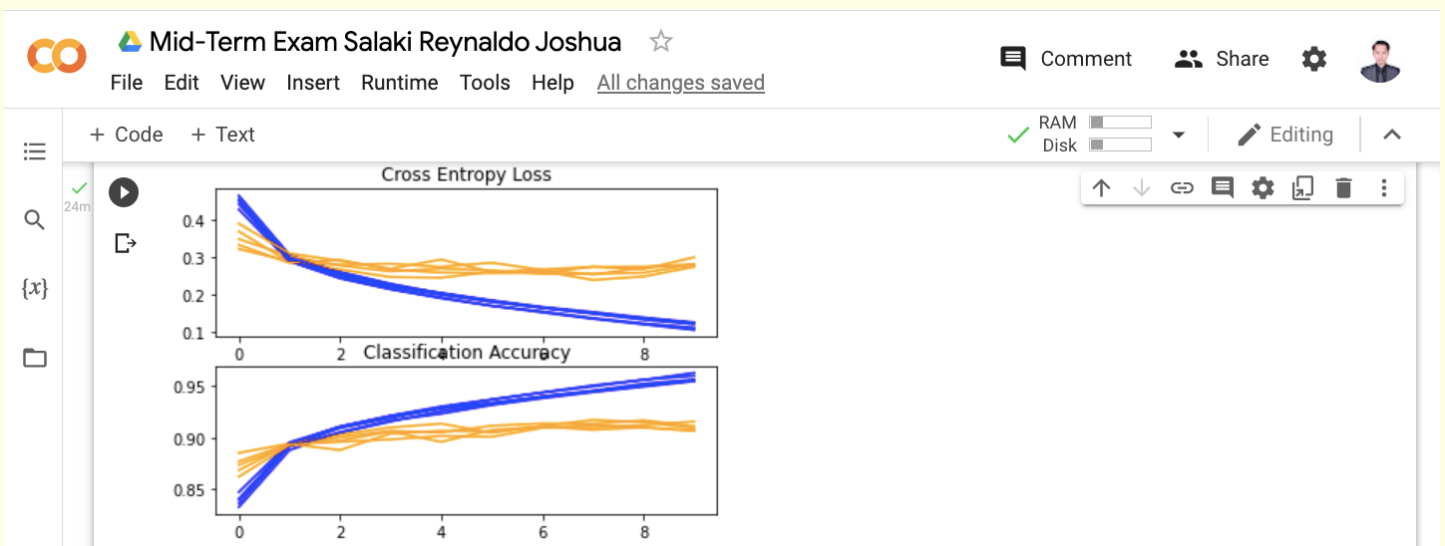
RUNNING RESULT

Running again reports model performance for each fold of the cross-validation process.


We can see perhaps a small improvement in model performance as compared to the baseline across the cross-validation folds.



A plot of the learning curves is created. As with the baseline model, we may see some slight overfitting. This could be addressed perhaps with the use of regularization or the training for fewer epochs.



Next, the estimated performance of the model is presented, showing performance with a very slight decrease in the mean accuracy of the model, 90.968% as compared to 90.967% with the baseline model. This may or may not be a real effect as it is within the bounds of the standard deviation. Perhaps more repeats of the experiment could tease out this fact.

 Mid-Term Exam Salaki Reynaldo Joshua ☆
File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings User

+ Code + Text

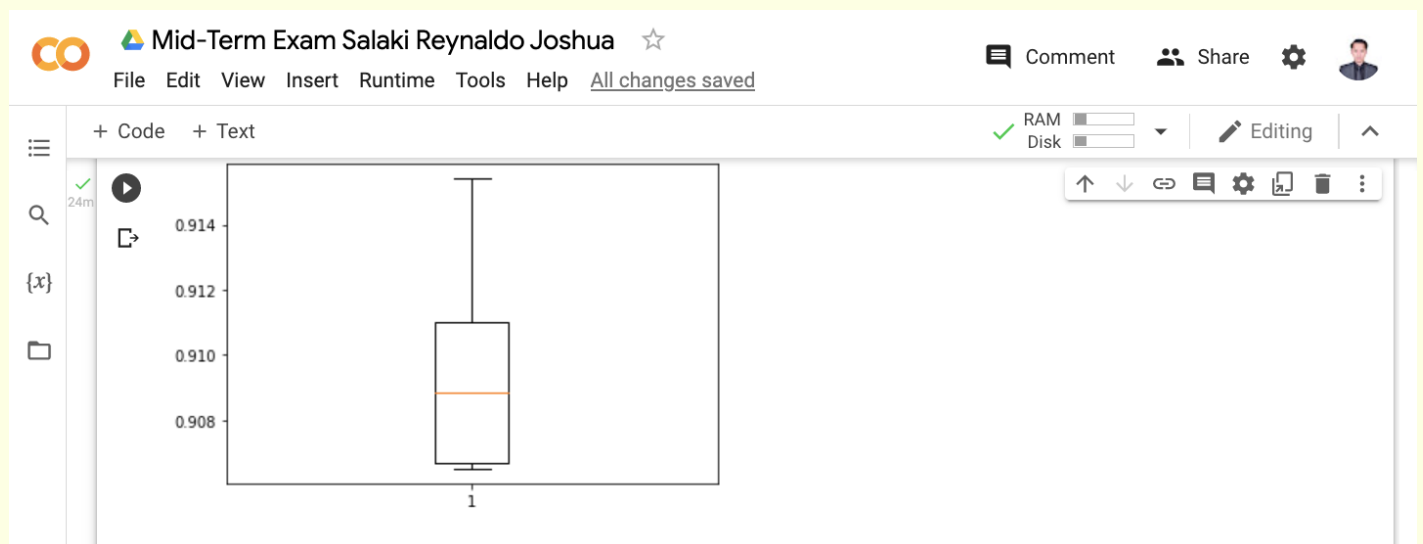
✓ RAM Disk

Editing

24m

Accuracy: mean=90.968 std=0.330, n=5

Up Down Link Comment Settings Copy Paste Delete More



INCREASING FILTERS

4

INCREASING FILTERS

An increase in the number of filters used in the convolutional layer can often improve performance, as it can provide more opportunities for extracting simple features from the input images.

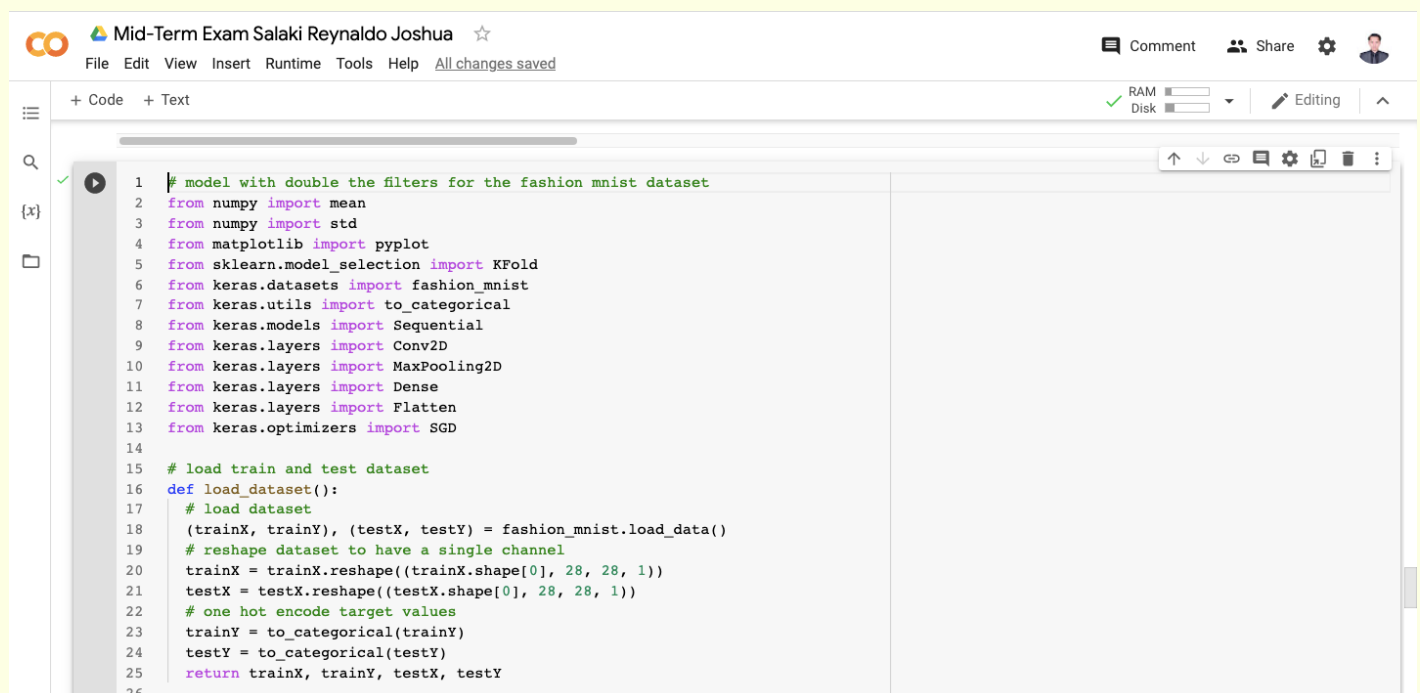
This is especially relevant when very small filters are used, such as 3×3 pixels.

In this change, we can increase the number of filters in the convolutional layer from 32 to double that at 64. We will also build upon the possible improvement offered by using 'same' padding.

An increase in the number of filters used in the convolutional layer can often improve performance, as it can provide more opportunities for extracting simple features from the input images.

This is especially relevant when very small filters are used, such as 3×3 pixels.

In this change, we can increase the number of filters in the convolutional layer from 32 to double that at 64. We will also build upon the possible improvement offered by using 'same' padding.



```
1 # model with double the filters for the fashion mnist dataset
2 from numpy import mean
3 from numpy import std
4 from matplotlib import pyplot
5 from sklearn.model_selection import KFold
6 from keras.datasets import fashion_mnist
7 from keras.utils import to_categorical
8 from keras.models import Sequential
9 from keras.layers import Conv2D
10 from keras.layers import MaxPooling2D
11 from keras.layers import Dense
12 from keras.layers import Flatten
13 from keras.optimizers import SGD
14
15 # load train and test dataset
16 def load_dataset():
17     # load dataset
18     (trainX, trainY), (testX, testY) = fashion_mnist.load_data()
19     # reshape dataset to have a single channel
20     trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
21     testX = testX.reshape((testX.shape[0], 28, 28, 1))
22     # one hot encode target values
23     trainY = to_categorical(trainY)
24     testY = to_categorical(testY)
25     return trainX, trainY, testX, testY
26
```

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Profile

+ Code + Text

RAM Disk Editing

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

```
# scale pixels
def prep_pixels(train, test):
    # convert from integers to floats
    train_norm = train.astype('float32')
    test_norm = test.astype('float32')
    # normalize to range 0-1
    train_norm = train_norm / 255.0
    test_norm = test_norm / 255.0
    # return normalized images
    return train_norm, test_norm

# define cnn model
def define_model():
    model = Sequential()
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    # compile model
    opt = SGD(lr=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Profile

+ Code + Text

RAM Disk Editing

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

```
# evaluate a model using k-fold cross-validation
def evaluate_model(dataX, dataY, n_folds=5):
    scores, histories = list(), list()
    # prepare cross validation
    kf = KFold(n_folds, shuffle=True, random_state=1)
    # enumerate splits
    for train_ix, test_ix in kf.split(dataX):
        # define model
        model = define_model()
        # select rows for train and test
        trainX, trainY, testX, testY = dataX[train_ix], dataY[train_ix], dataX[test_ix], dataY[test_ix]
        # fit model
        history = model.fit(trainX, trainY, epochs=10, batch_size=32, validation_data=(testX, testY), verbose=0)
        # evaluate model
        _, acc = model.evaluate(testX, testY, verbose=0)
        print('> %.3f' % (acc * 100.0))
        # append scores
        scores.append(acc)
        histories.append(history)
    return scores, histories
```

CO

Mid-Term Exam Salaki Reynaldo Joshua ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Profile

+ Code + Text

RAM Disk Editing

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

```
# plot diagnostic learning curves
def summarize_diagnostics(histories):
    for i in range(len(histories)):
        # plot loss
        pyplot.subplot(211)
        pyplot.title('Cross Entropy Loss')
        pyplot.plot(histories[i].history['loss'], color='blue', label='train')
        pyplot.plot(histories[i].history['val_loss'], color='orange', label='test')
        # plot accuracy
        pyplot.subplot(212)
        pyplot.title('Classification Accuracy')
        pyplot.plot(histories[i].history['accuracy'], color='blue', label='train')
        pyplot.plot(histories[i].history['val_accuracy'], color='orange', label='test')
    pyplot.show()

# summarize model performance
def summarize_performance(scores):
    # print summary
    print('Accuracy: mean=%.3f std=%.3f, n=%d' % (mean(scores)*100, std(scores)*100, len(scores)))
    # box and whisker plots of results
    pyplot.boxplot(scores)
    pyplot.show()
```



+ Code + Text



RAM

Disk

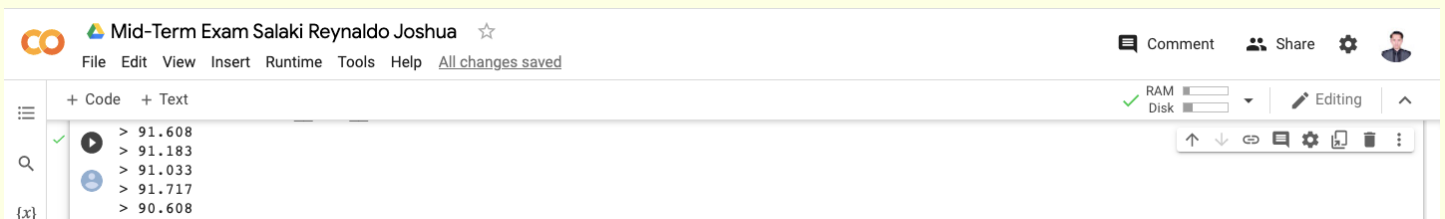
Editing



```
72 # plot diagnostic learning curves
73 def summarize_diagnostics(histories):
74     for i in range(len(histories)):
75         # plot loss
76         pyplot.subplot(211)
77         pyplot.title('Cross Entropy Loss')
78         pyplot.plot(histories[i].history['loss'], color='blue', label='train')
79         pyplot.plot(histories[i].history['val_loss'], color='orange', label='test')
80         # plot accuracy
81         pyplot.subplot(212)
82         pyplot.title('Classification Accuracy')
83         pyplot.plot(histories[i].history['accuracy'], color='blue', label='train')
84         pyplot.plot(histories[i].history['val_accuracy'], color='orange', label='test')
85     pyplot.show()
86
87 # summarize model performance
88 def summarize_performance(scores):
89     # print summary
90     print('Accuracy: mean=%.3f std=%.3f, n=%d' % (mean(scores)*100, std(scores)*100, len(scores)))
91     # box and whisker plots of results
92     pyplot.boxplot(scores)
93     pyplot.show()
94
```

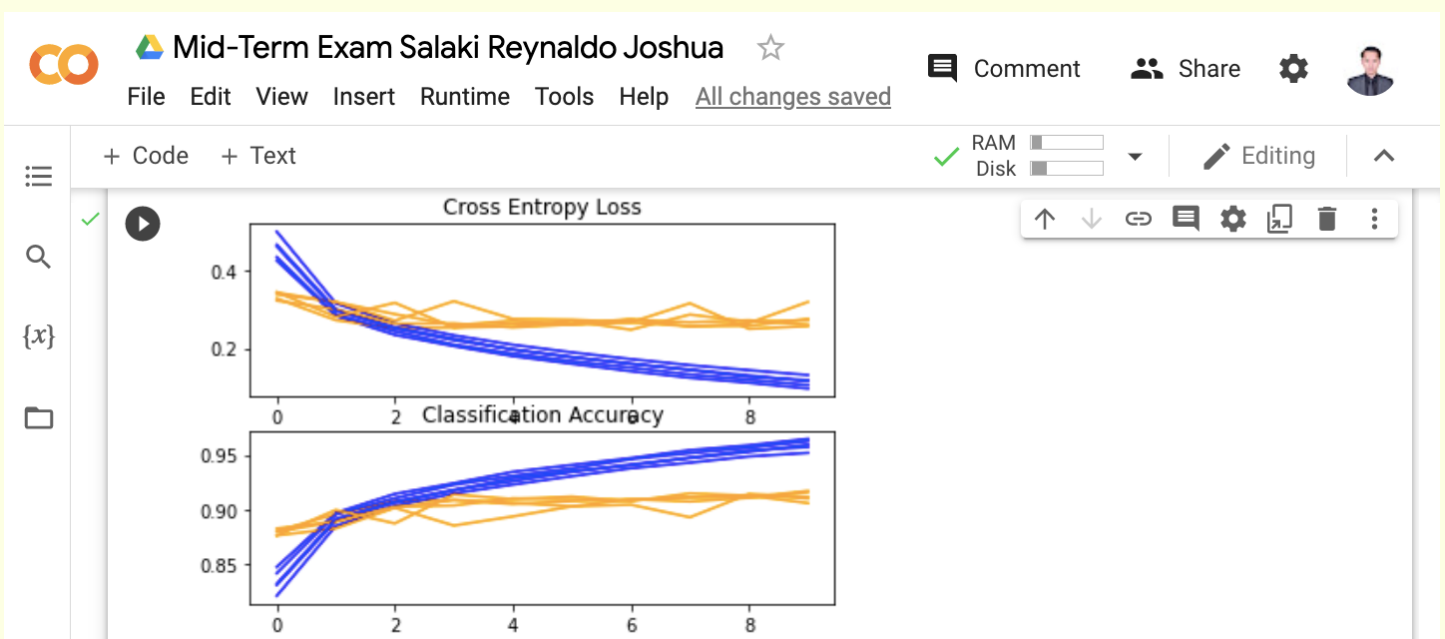
RUNNING RESULT

Running the reports model performance for each fold of the cross-validation process. The per-fold scores may suggest some further improvement over the baseline and using same padding alone.



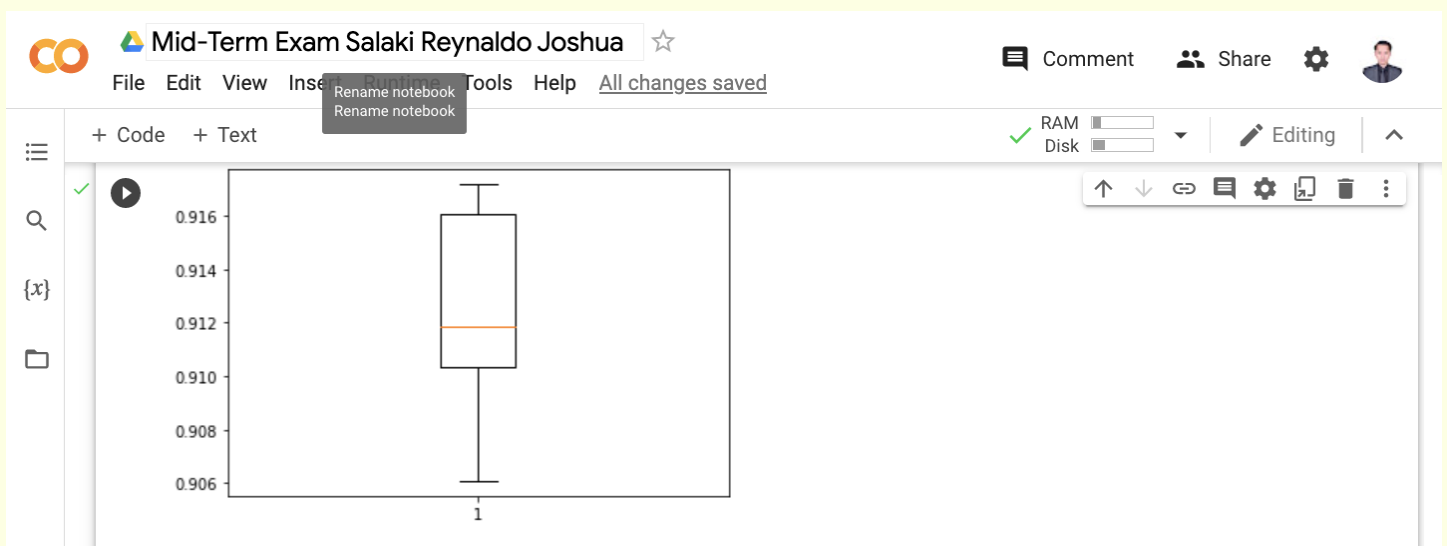
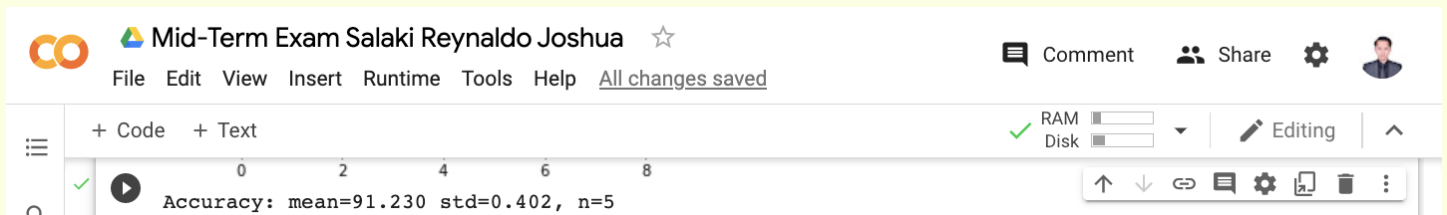
```
Mid-Term Exam Salaki Reynaldo Joshua
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
> 91.608
> 91.183
> 91.033
> 91.717
> 90.608
```

A plot of the learning curves is created, in this case showing that the models still have a reasonable fit on the problem, with a small sign of some of the runs overfitting.



Next, the estimated performance of the model is presented, showing a small improvement in performance as compared to the baseline from 91.230% to 90.968%.

Again, the change is still within the bounds of the standard deviation, and it is not clear whether the effect is real.



FINALIZE THE MODEL AND MAKE PREDICTIONS

5

FINALIZE THE MODEL AND MAKE PREDICTIONS

The process of model improvement may continue for as long as we have ideas and the time and resources to test them out.

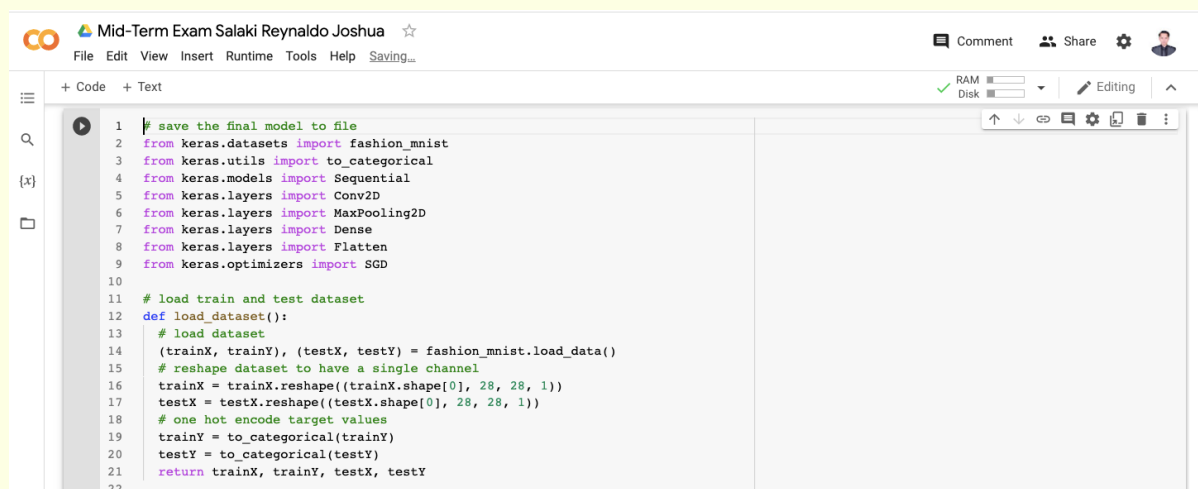
At some point, a final model configuration must be chosen and adopted. In this case, we will keep things simple and use the baseline model as the final model.

First, we will finalize our model, but fitting a model on the entire training dataset and saving the model to file for later use. We will then load the model and evaluate its performance on the hold out test dataset, to get an idea of how well the chosen model actually performs in practice. Finally, we will use the saved model to make a prediction on a single image.

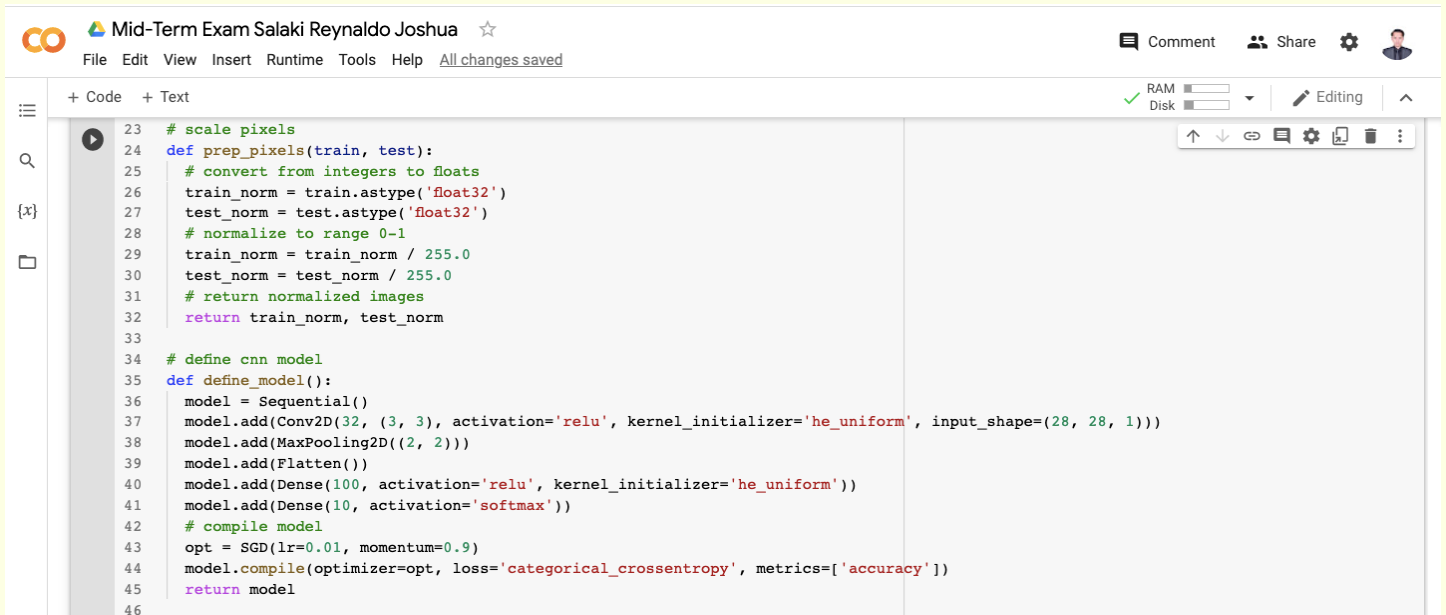
Save Final Model

A final model is typically fit on all available data, such as the combination of all train and test dataset.

In this Mid-Term Exam (Project), I am intentionally holding back a test dataset so that we can estimate the performance of the final model, which can be a good idea in practice. As such, we will fit our model on the training dataset only.



```
1 # save the final model to file
2 from keras.datasets import fashion_mnist
3 from keras.utils import to_categorical
4 from keras.models import Sequential
5 from keras.layers import Conv2D
6 from keras.layers import MaxPooling2D
7 from keras.layers import Dense
8 from keras.layers import Flatten
9 from keras.optimizers import SGD
10
11 # load train and test dataset
12 def load_dataset():
13     # load dataset
14     (trainX, trainY), (testX, testY) = fashion_mnist.load_data()
15     # reshape dataset to have a single channel
16     trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
17     testX = testX.reshape((testX.shape[0], 28, 28, 1))
18     # one hot encode target values
19     trainY = to_categorical(trainY)
20     testY = to_categorical(testY)
21     return trainX, trainY, testX, testY
22
```



The screenshot shows a code editor window titled "Mid-Term Exam Salaki Reynaldo Joshua". The code defines a function `prep_pixels` to normalize training and testing images, and a function `define_model` to create a sequential CNN model with layers: Conv2D, MaxPooling2D, Flatten, Dense (100 units), and Dense (10 units for softmax). The model is compiled with SGD optimizer, categorical crossentropy loss, and accuracy metric.

```
23 # scale pixels
24 def prep_pixels(train, test):
25     # convert from integers to floats
26     train_norm = train.astype('float32')
27     test_norm = test.astype('float32')
28     # normalize to range 0-1
29     train_norm = train_norm / 255.0
30     test_norm = test_norm / 255.0
31     # return normalized images
32     return train_norm, test_norm
33
34 # define cnn model
35 def define_model():
36     model = Sequential()
37     model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))
38     model.add(MaxPooling2D((2, 2)))
39     model.add(Flatten())
40     model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
41     model.add(Dense(10, activation='softmax'))
42     # compile model
43     opt = SGD(lr=0.01, momentum=0.9)
44     model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
45     return model
46
```



The screenshot shows the continuation of the code, defining a function `run_test_harness` to evaluate the model. It loads the dataset, prepares the pixel data, defines the model, fits it with 10 epochs and a batch size of 32, saves the model as `final_model.h5`, and then runs the test harness.

```
47 # run the test harness for evaluating a model
48 def run_test_harness():
49     # load dataset
50     trainX, trainY, testX, testY = load_dataset()
51     # prepare pixel data
52     trainX, testX = prep_pixels(trainX, testX)
53     # define model
54     model = define_model()
55     # fit model
56     model.fit(trainX, trainY, epochs=10, batch_size=32, verbose=0)
57     # save model
58     model.save('final_model.h5')
59
60 # entry point, run the test harness
61 run_test_harness()

```

Running the example loads the saved model and evaluates the model on the hold out test dataset.

CONCLUSION

6

CONCLUSION

In this mid-term exam (project), I discovered how to develop a convolutional neural network for clothing classification from scratch.

Specifically, I have learned:

- How to develop a test harness to develop a robust evaluation of a model and establish a baseline of performance for a classification task.
- How to explore extensions to a baseline model to improve learning and model capacity.
- How to develop a finalized model, evaluate the performance of the final model, and use it to make predictions on new images.

View the Mid-Term Exam Project

SCAN ME



ACKNOWLEDGEMENTS

My gratitude to you Professor Park Sanguk for the class. I truly appreciate you and the time you spent. Thank you very much for the class content on Image Classification Using the Convolutional Neural Network model. I enjoyed every minute of your lecture. Thank you, truly and sincerely.

CONTACT

Salaki Reynaldo Joshua

+62 821 8952 7635

joshua@kangwon.ac.kr