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

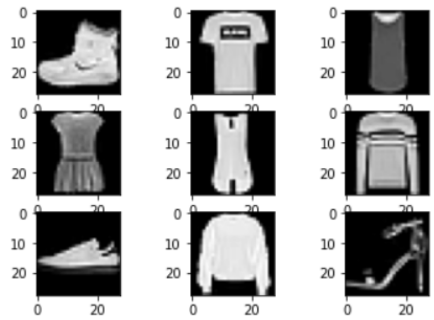
It is a more challenging classification problem than MNIST and top results are achieved by deep learning convolutional neural networks with a classification accuracy of about 90% to 95% on the hold-out test dataset.

Mid-Term Exam Salaki Reynaldo Joshua

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

Mid-Term Exam Salaki Reynaldo Joshua

```
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.

```python
# run the test harness for evaluating a model
def run_test_harness():
  # load dataset
  trainX, trainY, testX, testY = load_dataset()
  # prepare pixel data
  trainX, testX = prep_pixels(trainX, testX)
  # evaluate model
  scores, histories = evaluate_model(trainX, trainY)
  # learning curves
  summarize_diagnostics(histories)
  # summarize estimated performance
  summarize_performance(scores)
```

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

```python
# baseline cnn model for fashion mnist
from numpy import mean
from numpy import std
from matplotlib import pyplot
from sklearn.model_selection import KFold
from keras.datasets import fashion_mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD

# load train and test dataset
def load_dataset():
  # load dataset
  (trainX, trainY), (testX, testY) = fashion_mnist.load_data()
  # reshape dataset to have a single channel
  trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
  testX = testX.reshape((testX.shape[0], 28, 28, 1))
  # one hot encode target values
  trainY = to_categorical(trainY)
  testY = to_categorical(testY)
  return trainX, trainY, testX, testY

```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text RAM Disk Editing

```python
# 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(32, (3, 3), 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
```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text RAM Disk Editing

```python
# evaluate a model using k-fold cross-validation
def evaluate_model(dataX, dataY, n_folds=5):
  scores, histories = list(), list()
  # prepare cross validation
  kfold = KFold(n_folds, shuffle=True, random_state=1)
  # enumerate splits
  for train_ix, test_ix in kfold.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
```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
# 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()
```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

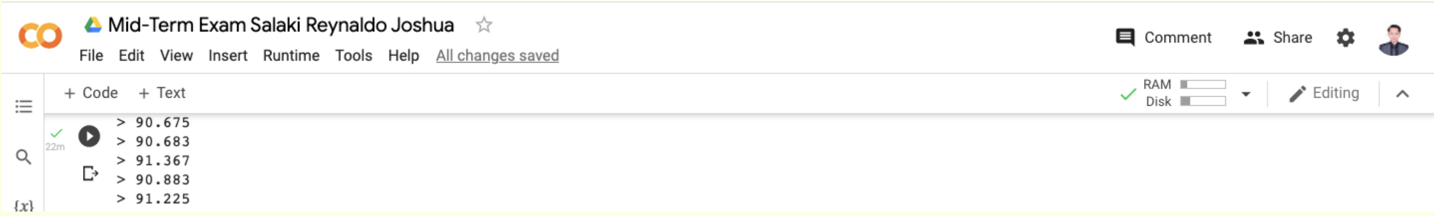
+ Code + Text

RAM Disk Editing

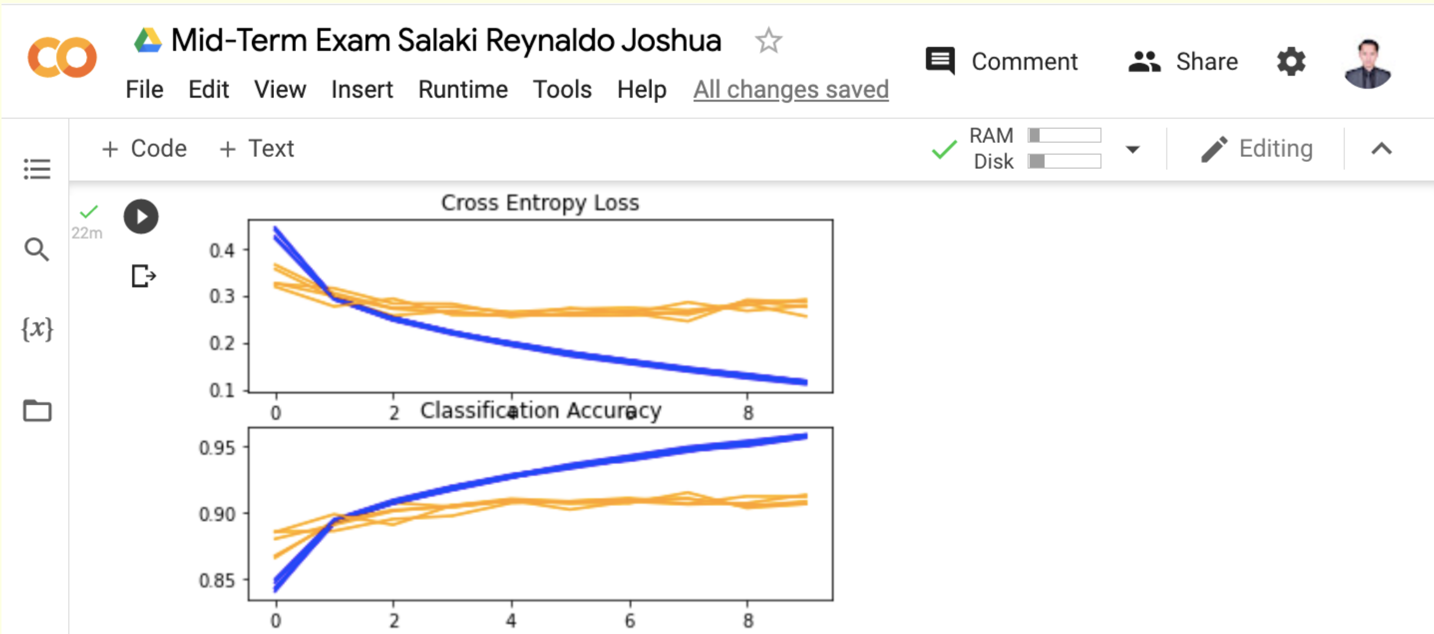
```

# run the test harness for evaluating a model
def run_test_harness():
  # load dataset
  trainX, trainY, testX, testY = load_dataset()
  # prepare pixel data
  trainX, testX = prep_pixels(trainX, testX)
  # evaluate model
  scores, histories = evaluate_model(trainX, trainY)
  # learning curves
  summarize_diagnostics(histories)
  # summarize estimated performance
  summarize_performance(scores)

# entry point, run the test harness
run_test_harness()
```

RUNNING RESULT

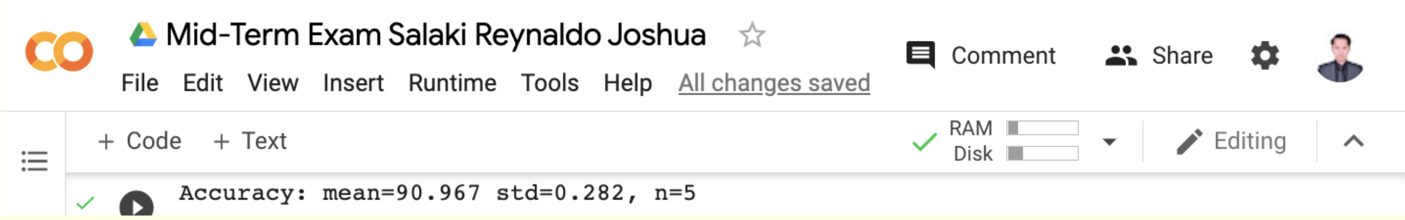
Running the classification accuracy for each fold of the cross-validation process. This is helpful to get an idea that the model evaluation is progressing.
We can see that for each fold, the baseline model achieved an error rate below 10%, and in two cases 98% and 99% accuracy. These are good results.



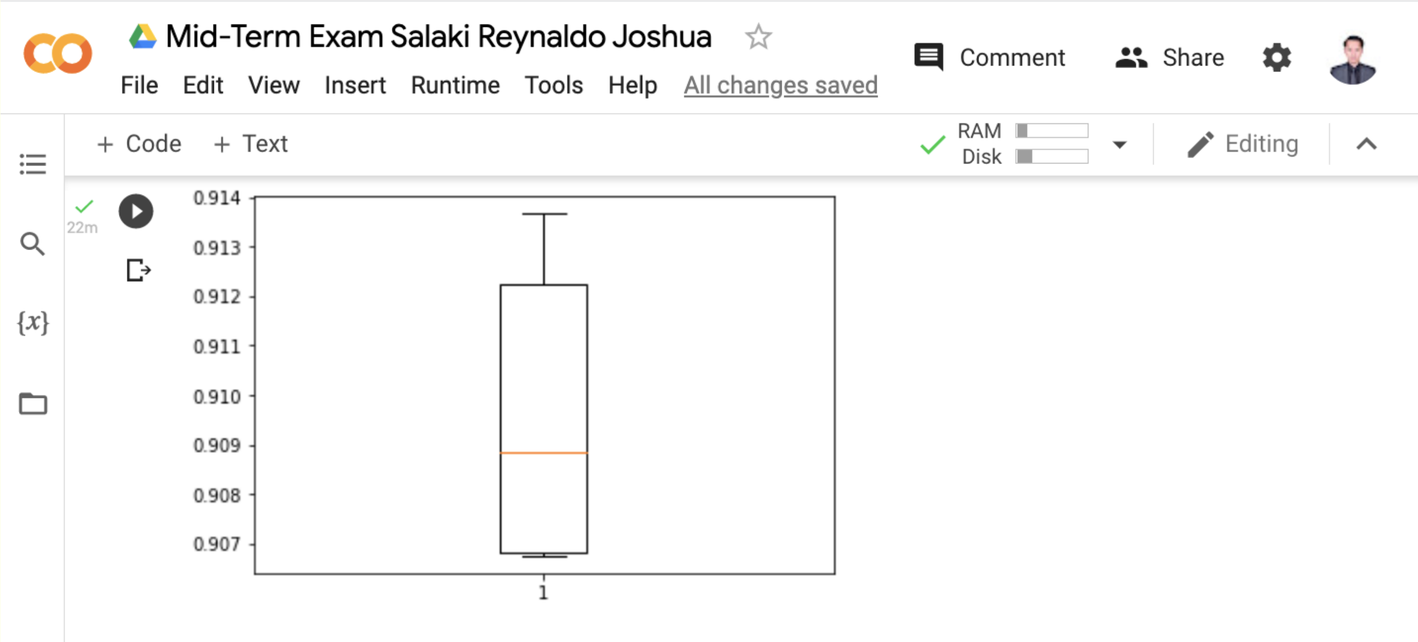
Next, a diagnostic plot is shown, giving insight into the learning behavior of the model across each fold.
In this case, we can see that the model generally achieves a good fit, with train and test learning curves converging. There may be some signs of slight overfitting.



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.



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.

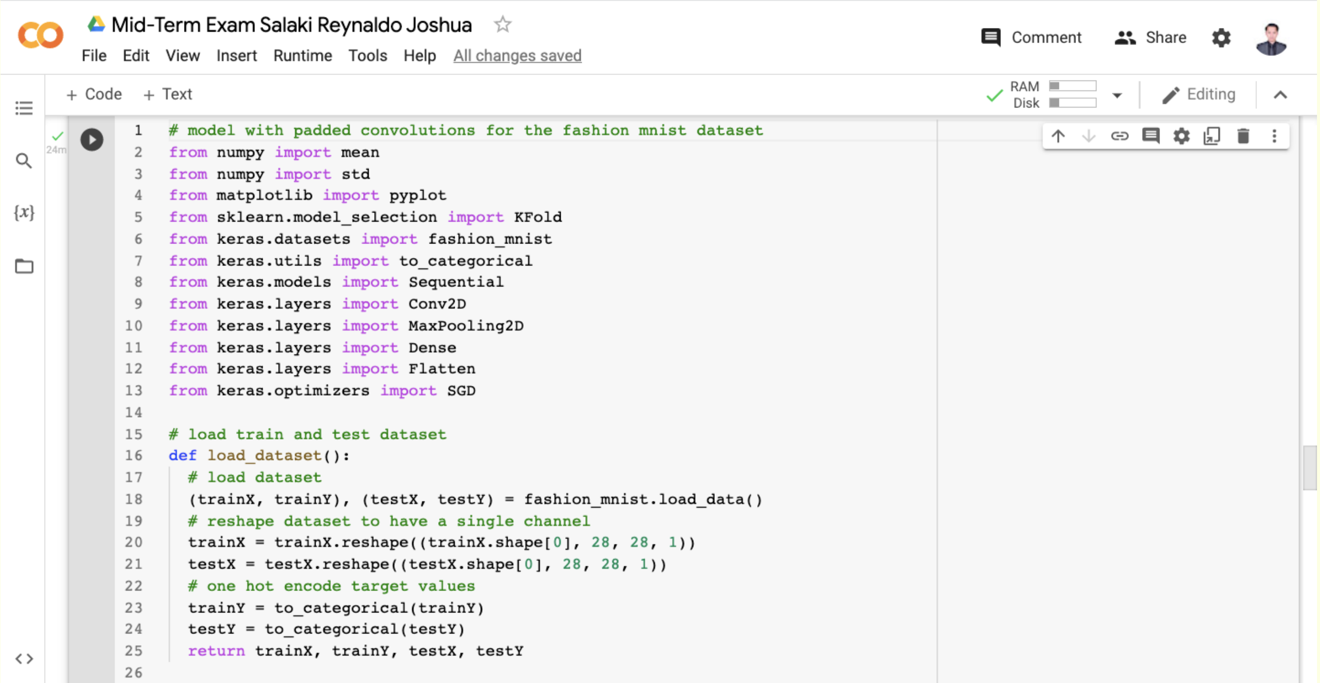
PADDING CONVOLUTIONS

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 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 ‘same‘ padding so that zero values are added around the input such that the output has the same size as the input.




```python
# model with padded convolutions for the fashion mnist dataset
from numpy import mean
from numpy import std
from matplotlib import pyplot
from sklearn.model_selection import KFold
from keras.datasets import fashion_mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD

# load train and test dataset
def load_dataset():
  # load dataset
  (trainX, trainY), (testX, testY) = fashion_mnist.load_data()
  # reshape dataset to have a single channel
  trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
  testX = testX.reshape((testX.shape[0], 28, 28, 1))
  # one hot encode target values
  trainY = to_categorical(trainY)
  testY = to_categorical(testY)
  return trainX, trainY, testX, testY

```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python

# 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(32, (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

```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
# evaluate a model using k-fold cross-validation
def evaluate_model(dataX, dataY, n_folds=5):
  scores, histories = list(), list()
  # prepare cross validation
  kfold = KFold(n_folds, shuffle=True, random_state=1)
  # enumerate splits
  for train_ix, test_ix in kfold.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

```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```python
# 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()

```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

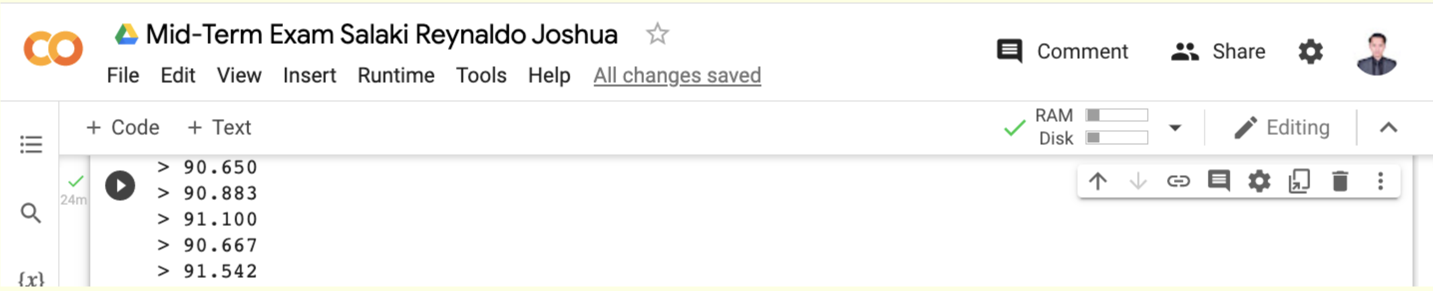
RAM Disk Editing

```
# run the test harness for evaluating a model
def run_test_harness():
  # load dataset
  trainX, trainY, testX, testY = load_dataset()
  # prepare pixel data
  trainX, testX = prep_pixels(trainX, testX)
  # evaluate model
  scores, histories = evaluate_model(trainX, trainY)
  # learning curves
  summarize_diagnostics(histories)
  # summarize estimated performance
  summarize_performance(scores)

# entry point, run the test harness
run_test_harness()
```

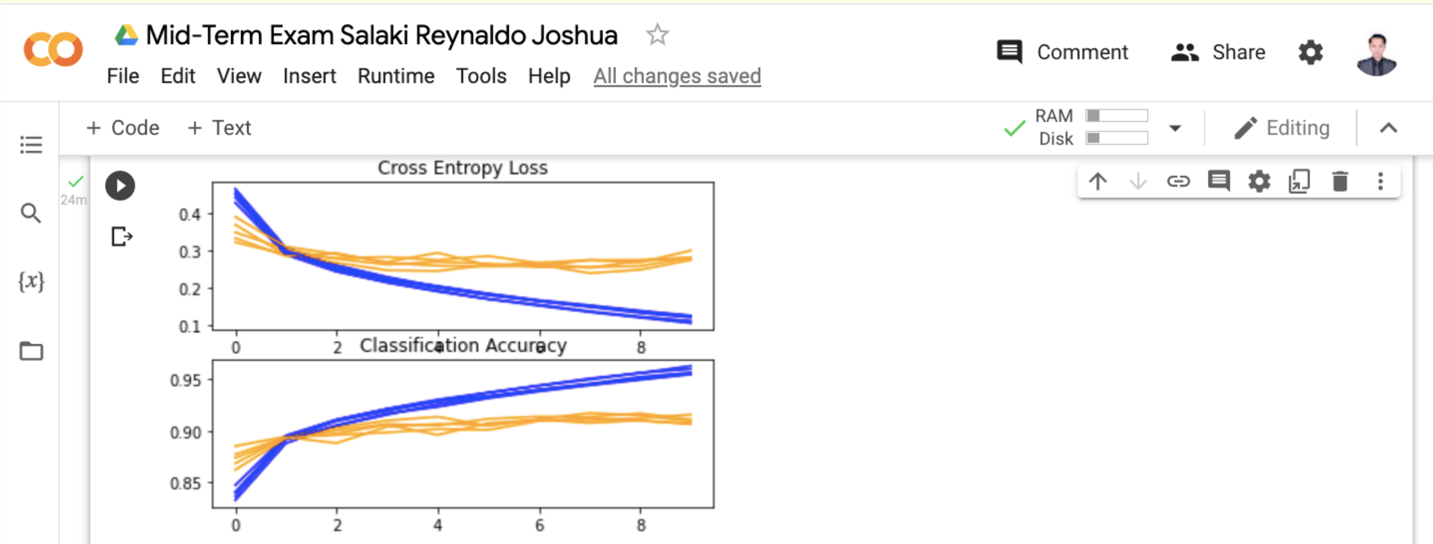
RUNNING RESULT

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

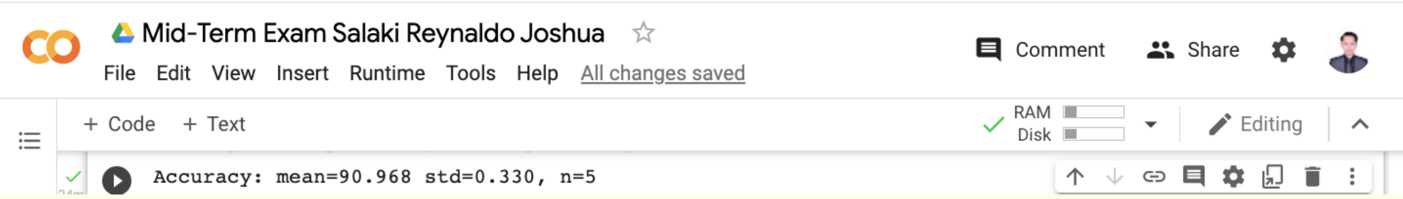
We can see perhaps a small improvement in model performance as compared to the baseline across the cross-validation folds.

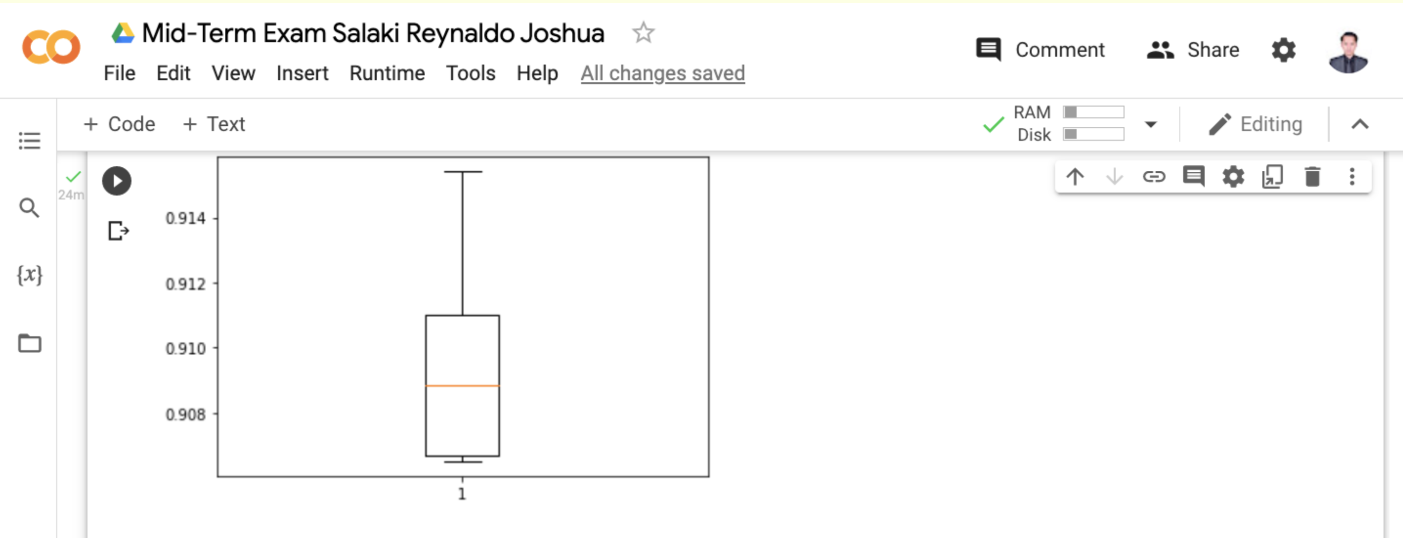


A plot of the learning curves is created. As with the baseline model, we may see some slight overfitting. This could be addressed perhaps with the use of regularization or the training for fewer epochs.



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.





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.

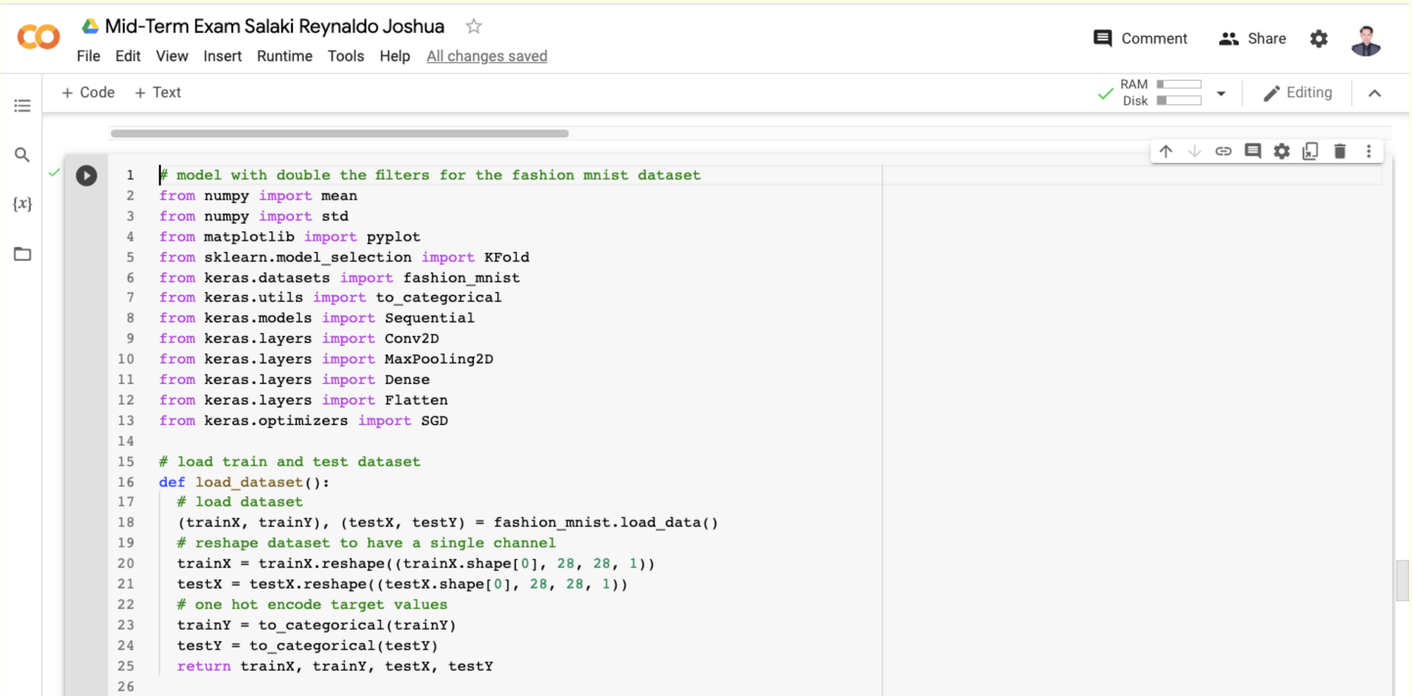
This is especially relevant when very small filters are used, such as 3×3 pixels.

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.



Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

```python
# 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
```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

```python
# evaluate a model using k-fold cross-validation
def evaluate_model(dataX, dataY, n_folds=5):
  scores, histories = list(), list()
  # prepare cross validation
  kfold = KFold(n_folds, shuffle=True, random_state=1)
  # enumerate splits
  for train_ix, test_ix in kfold.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
```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

```python
# 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()
```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

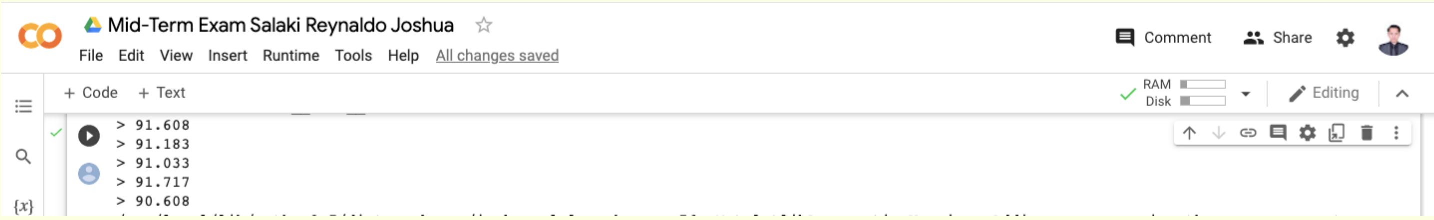
+ Code + Text

RAM Disk Editing

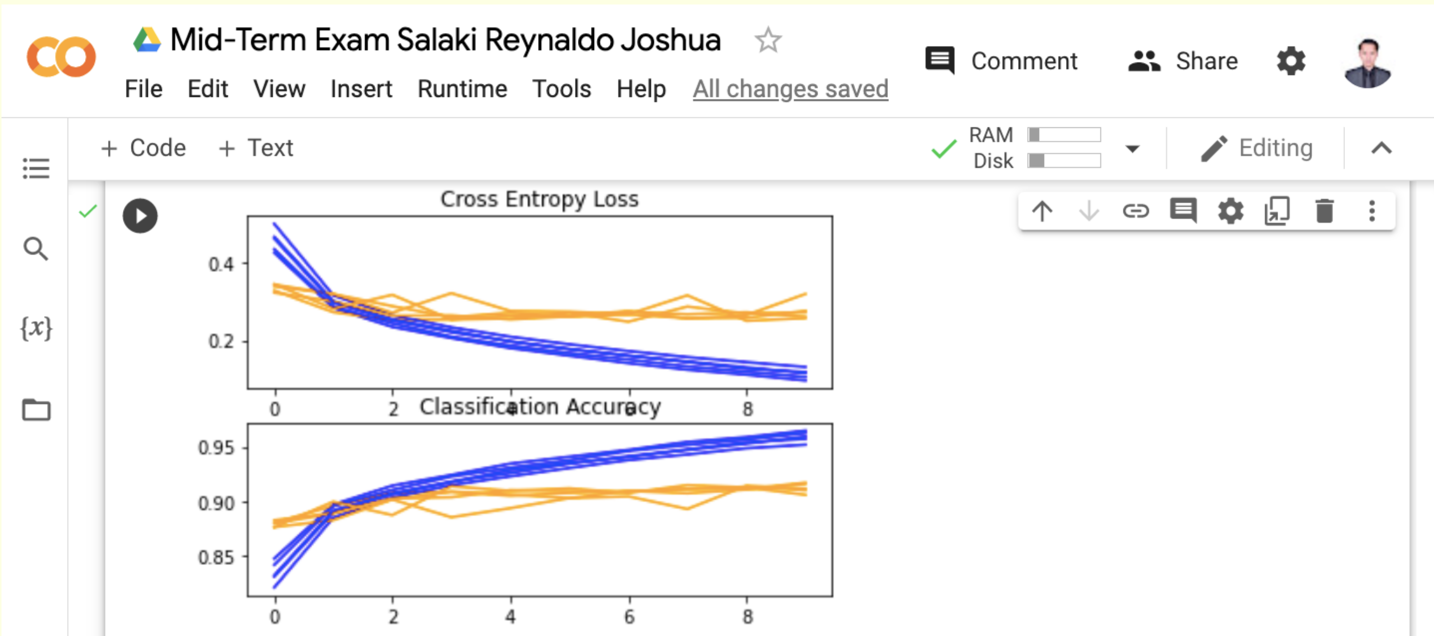
```
# plot diagnostic learning curves
def summarize_diagnostics(histories):
  for i in range(len(histories)):
    # plot loss
    pyplot.subplot(211)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(histories[i].history['loss'], color='blue', label='train')
    pyplot.plot(histories[i].history['val_loss'],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

```

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.

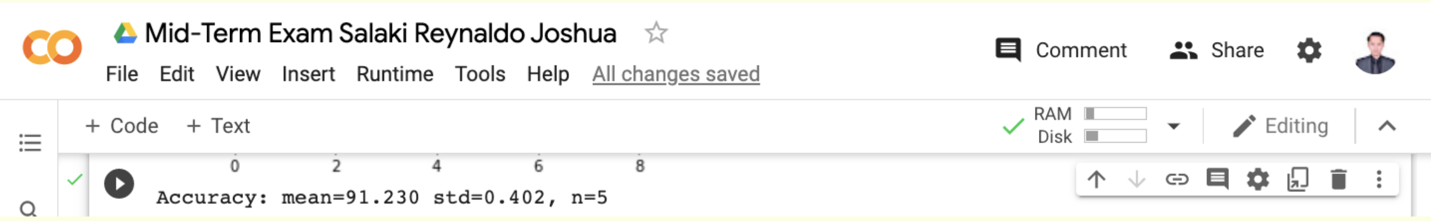


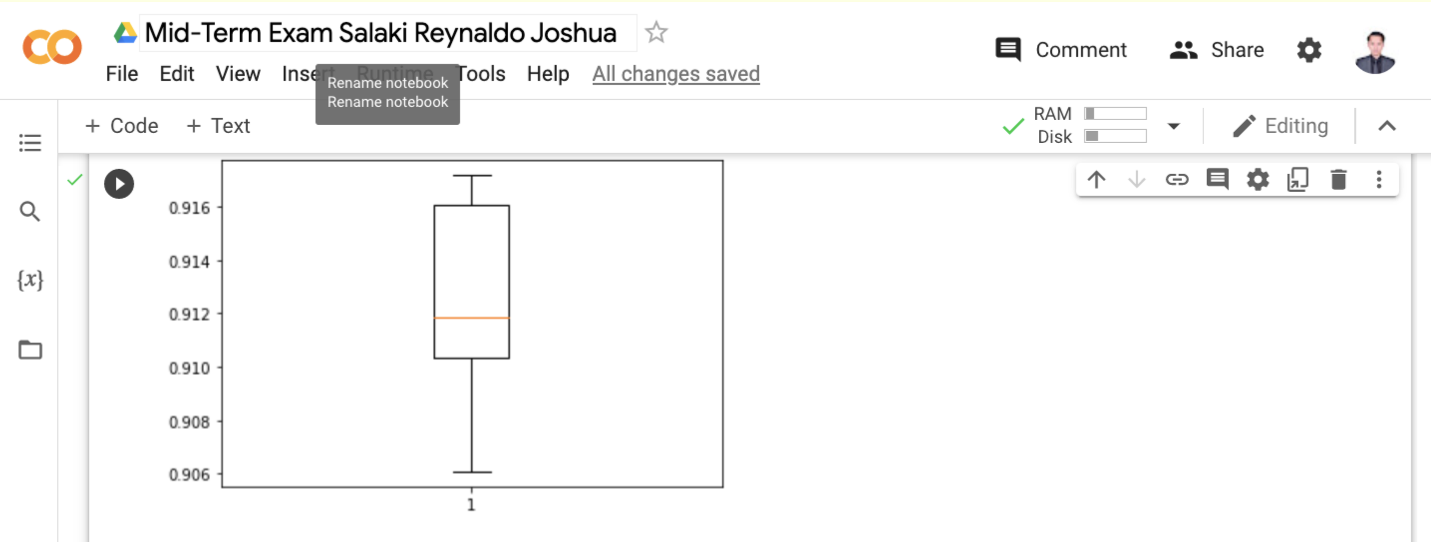
A plot of the learning curves is created, in this case showing that the models still have a reasonable fit on the problem, with a small sign of some of the runs overfitting.



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

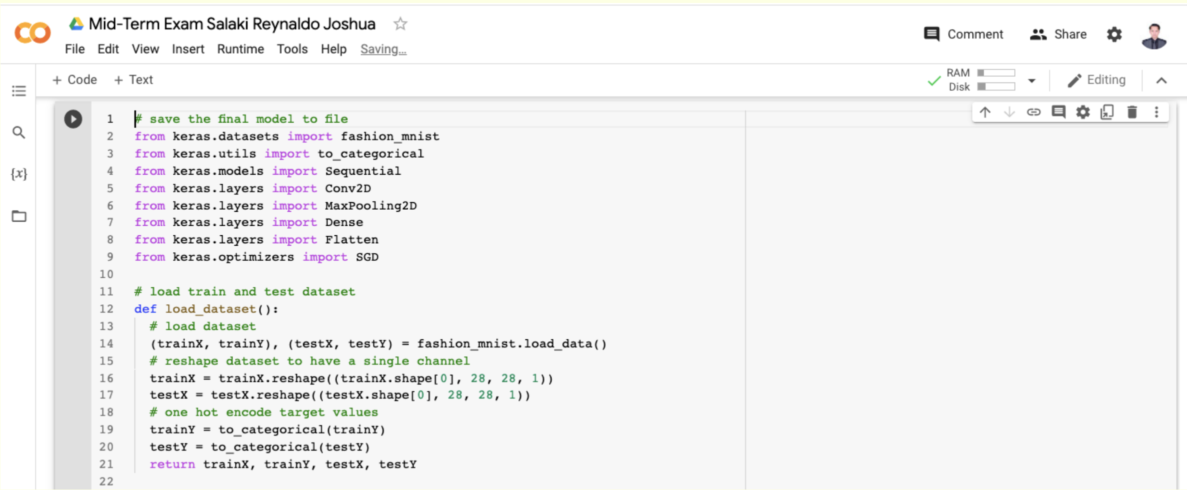
5

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.




```python
# save the final model to file
from keras.datasets import fashion_mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD

# load train and test dataset
def load_dataset():
  # load dataset
  (trainX, trainY), (testX, testY) = fashion_mnist.load_data()
  # reshape dataset to have a single channel
  trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
  testX = testX.reshape((testX.shape[0], 28, 28, 1))
  # one hot encode target values
  trainY = to_categorical(trainY)
  testY = to_categorical(testY)
  return trainX, trainY, testX, testY

```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```python
# 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(32, (3, 3), 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

```

Mid-Term Exam Salaki Reynaldo Joshua

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```python
# run the test harness for evaluating a model
def run_test_harness():
  # load dataset
  trainX, trainY, testX, testY = load_dataset()
  # prepare pixel data
  trainX, testX = prep_pixels(trainX, testX)
  # define model
  model = define_model()
  # fit model
  model.fit(trainX, trainY, epochs=10, batch_size=32, verbose=0)
  # save model
  model.save('final_model.h5')

# entry point, run the test harness
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