assignment06convnets6.1_6.2a

July 15, 2023

[75]:
```
from keras.datasets import mnist
from keras.utils import to_categorical
from keras import optimizers
import os, shutil
from keras.applications import VGG16
from keras.models import Sequential, load_model
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import imageio
from keras.layers.core import Dense, Dropout, Activation
from pathlib import Path
from keras import layers
from keras import models
```

[76]:
```
homebase=Path('/home/jovyan/DSC650/dsc650/')

myresults=Path('/home/jovyan/DSC650/dsc650/assignments/assignment06/').
 ↪joinpath('results')

myresults.mkdir(parents=True, exist_ok=True)
```

[77]:
```
model=models.Sequential()
model.add(layers.Conv2D(32,(3,3),activation='relu', input_shape=(28,28,1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_54 (Conv2D)          (None, 26, 26, 32)        320
```

```
 max_pooling2d_28 (MaxPoolin  (None, 13, 13, 32)       0
 g2D)

 conv2d_55 (Conv2D)          (None, 11, 11, 64)        18496

 max_pooling2d_29 (MaxPoolin  (None, 5, 5, 64)         0
 g2D)

 conv2d_56 (Conv2D)          (None, 3, 3, 64)          36928

=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```

[78]:
```
from contextlib import redirect_stdout

summaryFile = myresults.joinpath('Assignment6.1ModelSummary.txt')
with open(summaryFile, 'w') as f:
    with redirect_stdout(f):
        model.summary()
```

[79]:
```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

[80]:
```
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_54 (Conv2D)          (None, 26, 26, 32)        320

 max_pooling2d_28 (MaxPoolin  (None, 13, 13, 32)       0
 g2D)

 conv2d_55 (Conv2D)          (None, 11, 11, 64)        18496

 max_pooling2d_29 (MaxPoolin  (None, 5, 5, 64)         0
 g2D)

 conv2d_56 (Conv2D)          (None, 3, 3, 64)          36928

 flatten_9 (Flatten)         (None, 576)               0

 dense_18 (Dense)            (None, 64)                36928
```

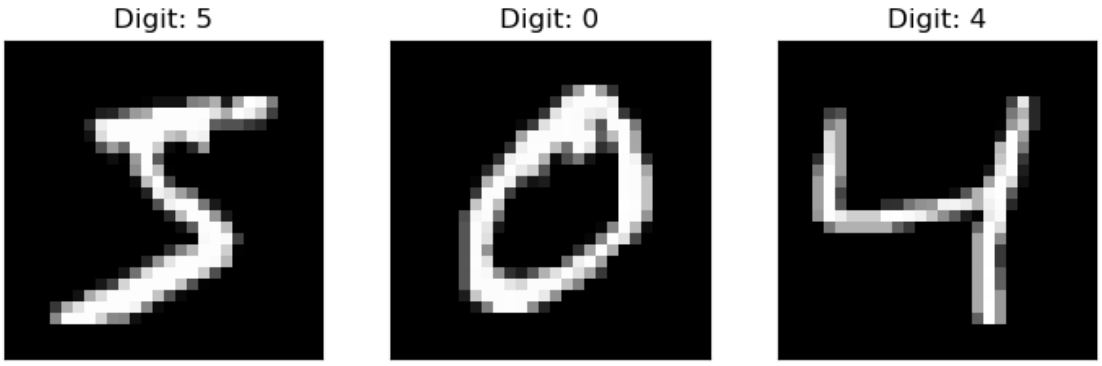
```
 dense_19 (Dense)            (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

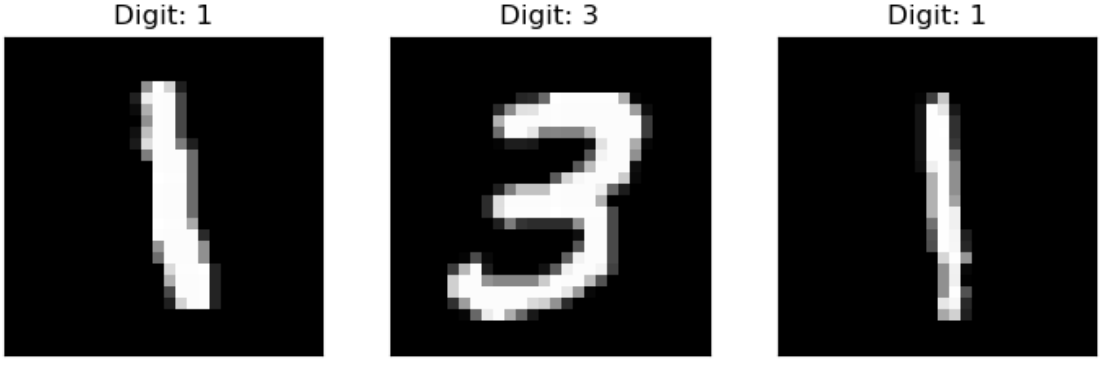
[81]:
```
summaryFile2 = myresults.joinpath('Assignment6.1ModelSummary2.txt')
with open(summaryFile2, 'w') as f:
    with redirect_stdout(f):
        model.summary()
```

[82]:
```
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

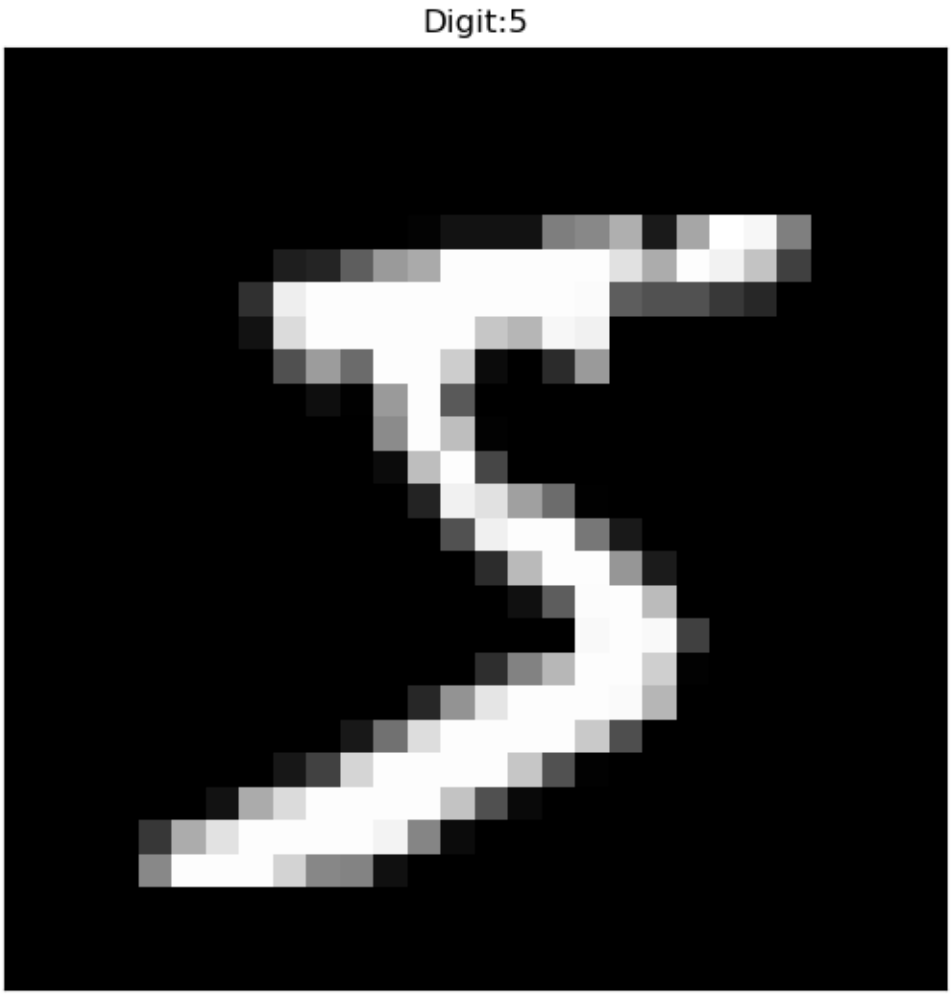
[83]:
```
print("Sneak Peek of the training images")
figure = plt.figure()
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(train_images[i], cmap='gray', interpolation='none')
    plt.title("Digit: {}".format(train_labels[i]))
    plt.tight_layout()
    plt.xticks([])
    plt.yticks([])
imgageFile = myresults.joinpath('Assignment_6.1ImagePeek.png')
plt.savefig(imgageFile)
plt.show()

import warnings
warnings.filterwarnings('ignore')
```

```
Sneak Peek of the training images
```


Digit: 5
Digit: 0
Digit: 4

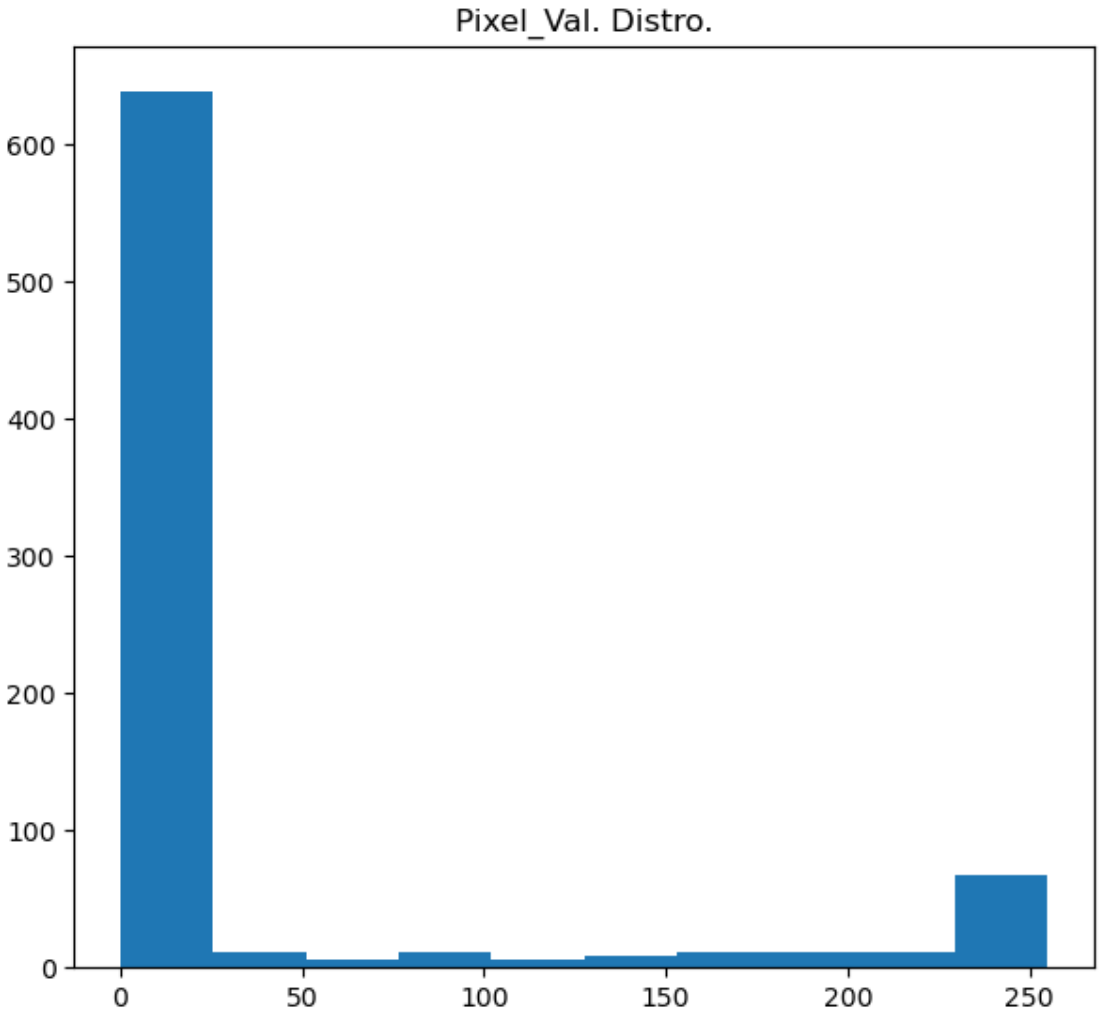

Digit: 1
Digit: 3
Digit: 1

[86]:
```
fig = plt.figure()
plt.subplot(2,1,1)
plt.imshow(train_images[0], cmap='gray', interpolation='none')
plt.title("Digit:{}".format(train_labels[0]))
plt.xticks([])
plt.yticks([])
Dig_images_file = myresults.joinpath('Assignment_6.1Digits.png')
plt.savefig(Dig_images_file)
plt.show()
```



[87]:
```
plt.subplot(2,1,2)
plt.hist(train_images[0].reshape(784))
plt.title("Pixel_Val. Distro.")
```

```
Distro_images_file = myresults.joinpath('Assignment_6.1PixelValueDistribution.
  ↪png')
plt.savefig(Distro_images_file)
plt.show()
```



[88]:
```
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

[89]:
```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[90]:
```
history = model.fit(train_images, train_labels,
          batch_size=128,
          epochs=20,
          verbose=2,
          validation_data=(test_images, test_labels))

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test Accuracy: %s ' % test_acc)

results4model = myresults.joinpath('Assignment_6.1Results4model.h5')
model.save(results4model)
print('Saved trained model at %s ' % results4model)
```

```
Epoch 1/20
469/469 - 11s - loss: 0.2442 - accuracy: 0.9241 - val_loss: 0.0594 -
val_accuracy: 0.9814 - 11s/epoch - 23ms/step
Epoch 2/20
469/469 - 10s - loss: 0.0575 - accuracy: 0.9821 - val_loss: 0.0481 -
val_accuracy: 0.9853 - 10s/epoch - 21ms/step
Epoch 3/20
469/469 - 10s - loss: 0.0388 - accuracy: 0.9871 - val_loss: 0.0374 -
val_accuracy: 0.9870 - 10s/epoch - 21ms/step
Epoch 4/20
469/469 - 9s - loss: 0.0286 - accuracy: 0.9910 - val_loss: 0.0314 -
val_accuracy: 0.9898 - 9s/epoch - 20ms/step
Epoch 5/20
469/469 - 10s - loss: 0.0227 - accuracy: 0.9930 - val_loss: 0.0413 -
val_accuracy: 0.9870 - 10s/epoch - 21ms/step
Epoch 6/20
469/469 - 10s - loss: 0.0186 - accuracy: 0.9942 - val_loss: 0.0304 -
val_accuracy: 0.9900 - 10s/epoch - 21ms/step
Epoch 7/20
469/469 - 9s - loss: 0.0137 - accuracy: 0.9958 - val_loss: 0.0293 -
val_accuracy: 0.9922 - 9s/epoch - 20ms/step
Epoch 8/20
469/469 - 10s - loss: 0.0115 - accuracy: 0.9964 - val_loss: 0.0326 -
val_accuracy: 0.9916 - 10s/epoch - 21ms/step
Epoch 9/20
469/469 - 10s - loss: 0.0097 - accuracy: 0.9968 - val_loss: 0.0347 -
val_accuracy: 0.9900 - 10s/epoch - 21ms/step
Epoch 10/20
469/469 - 10s - loss: 0.0078 - accuracy: 0.9976 - val_loss: 0.0374 -
val_accuracy: 0.9901 - 10s/epoch - 21ms/step
```

```
Epoch 11/20
469/469 - 10s - loss: 0.0065 - accuracy: 0.9980 - val_loss: 0.0428 -
val_accuracy: 0.9893 - 10s/epoch - 21ms/step
Epoch 12/20
469/469 - 10s - loss: 0.0057 - accuracy: 0.9982 - val_loss: 0.0337 -
val_accuracy: 0.9924 - 10s/epoch - 21ms/step
Epoch 13/20
469/469 - 10s - loss: 0.0047 - accuracy: 0.9985 - val_loss: 0.0386 -
val_accuracy: 0.9920 - 10s/epoch - 21ms/step
Epoch 14/20
469/469 - 10s - loss: 0.0037 - accuracy: 0.9990 - val_loss: 0.0385 -
val_accuracy: 0.9926 - 10s/epoch - 21ms/step
Epoch 15/20
469/469 - 10s - loss: 0.0036 - accuracy: 0.9987 - val_loss: 0.0354 -
val_accuracy: 0.9921 - 10s/epoch - 21ms/step
Epoch 16/20
469/469 - 10s - loss: 0.0028 - accuracy: 0.9990 - val_loss: 0.0434 -
val_accuracy: 0.9921 - 10s/epoch - 20ms/step
Epoch 17/20
469/469 - 10s - loss: 0.0026 - accuracy: 0.9992 - val_loss: 0.0445 -
val_accuracy: 0.9920 - 10s/epoch - 21ms/step
Epoch 18/20
469/469 - 10s - loss: 0.0019 - accuracy: 0.9994 - val_loss: 0.0467 -
val_accuracy: 0.9919 - 10s/epoch - 21ms/step
Epoch 19/20
469/469 - 10s - loss: 0.0027 - accuracy: 0.9991 - val_loss: 0.0604 -
val_accuracy: 0.9901 - 10s/epoch - 20ms/step
Epoch 20/20
469/469 - 10s - loss: 0.0019 - accuracy: 0.9994 - val_loss: 0.0454 -
val_accuracy: 0.9920 - 10s/epoch - 21ms/step
313/313 [==============================] - 2s 5ms/step - loss: 0.0454 -
accuracy: 0.9920
Test Accuracy: 0.9919999837875366
Saved trained model at /home/jovyan/DSC650/dsc650/assignments/assignment06/resul
ts/Assignment_6.1Results4model.h5
```

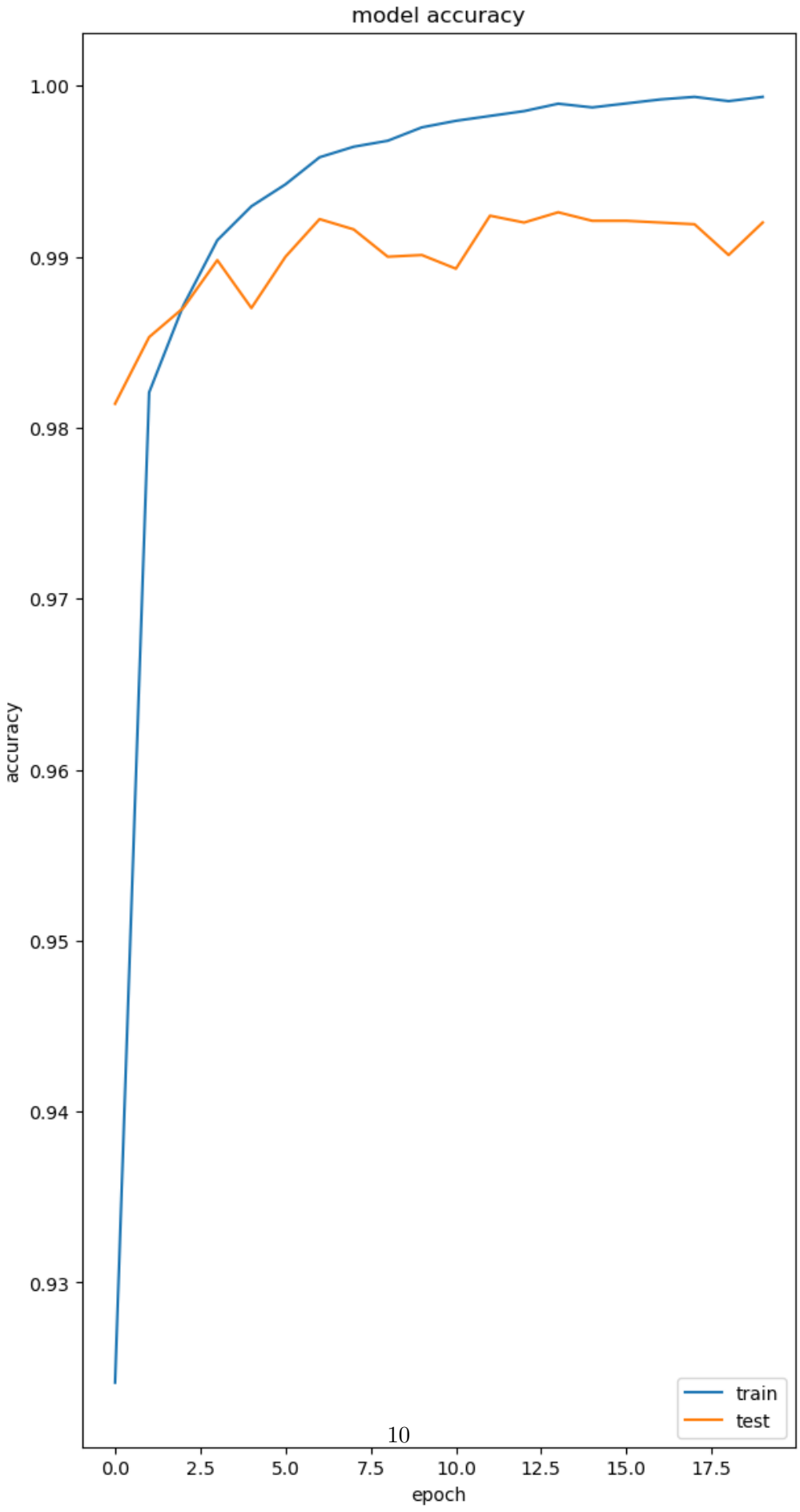
[91]:
```
results4model_file = myresults.joinpath('Assignment_6.1Results4Model.h5')
model.save(results4model_file)
print("I've Saved my trained model at %s " % results4model_file)
```

```
I've Saved my trained model at /home/jovyan/DSC650/dsc650/assignments/assignment
06/results/Assignment_6.1Results4Model.h5
```

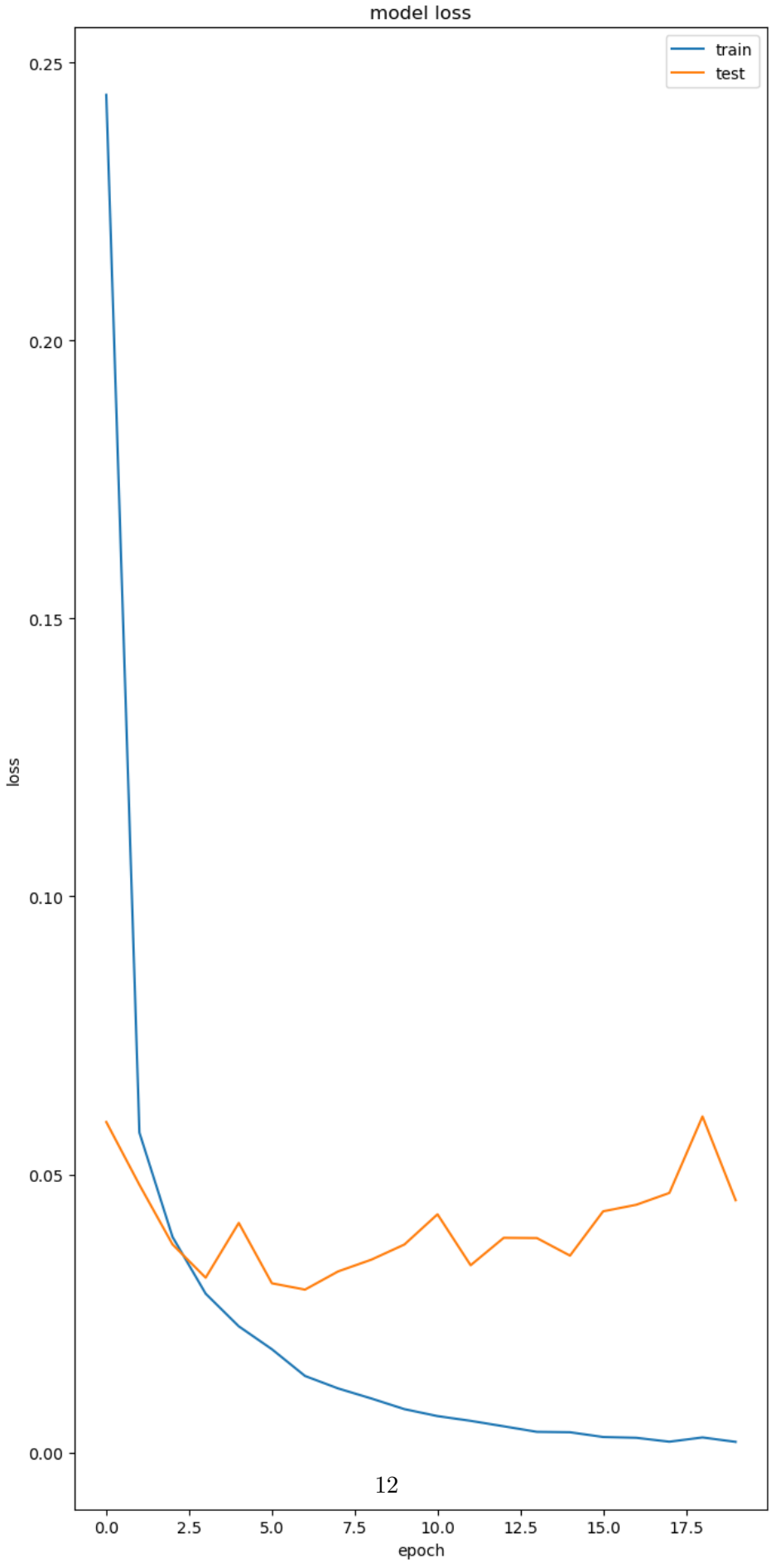
[92]:
```
#Let's take a look
fig = plt.figure()

plt.plot(history.history['accuracy'])
```

```
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')
```

[92]: <matplotlib.legend.Legend at 0x7f67b45bb040>


model accuracy
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch
train
test

[93]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.tight_layout()
img_lossfile = myresults.joinpath('Assignment_6.1ModelLoss.png')
plt.savefig(img_lossfile)
plt.show()
```


model loss
train
test
loss
epoch
0.25
0.20
0.15
0.10
0.05
0.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

[94]:
```
import time
start_time = time.time()

my_mnisty_model = load_model(results4model)
loss_and_metrics = my_mnisty_model.evaluate(test_images, test_labels)
print("Test Loss", loss_and_metrics[0])
print("Test Accuracy", loss_and_metrics[1])
print("Time = %s seconds " % (time.time() - start_time))
```

```
313/313 [==============================] - 2s 5ms/step - loss: 0.0454 -
accuracy: 0.9920
Test Loss 0.045355696231126785
Test Accuracy 0.9919999837875366
Time = 1.91609525680542 seconds
```

[95]:
```
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
```

[96]:
```
CIFAR= tf.keras.datasets.cifar10.load_data()
```

[97]:
```
(x_train, y_train), (x_test, y_test) = CIFAR
```

[98]:
```
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

[98]:
```
((50000, 32, 32, 3), (50000, 1), (10000, 32, 32, 3), (10000, 1))
```

[99]:
```
def load_dataset():
    (x_train, y_train), (x_test, y_test) = CIFAR
    trainY=to_categorical(y_train)
    testY=to_categorical(y_test)

    return x_train, trainY, x_test, testY
```

[100]:
```
#homemade scale
def prep_pixels(train, test):
    train_norm = train.astype('float32')
```

```
    test_norm = test.astype('float32')
    train_norm = train_norm / 255.0
    test_norm = test_norm / 255.0

    return train_norm, test_norm
```

[101]:
```
def sum_diagnostics(history):
    plt.subplot(211)
    plt.title('Cross Entropy Loss')
    plt.plot(history.history['loss'], color='blue',
                label='train')
    plt.plot(history.history['val_loss'], color='orange',
                label='test')
    plt.subplot(212)
    plt.title('Classification Accuracy')
    plt.plot(history.history['accuracy'], color='blue',
                label='train')
    plt.plot(history.history['val_accuracy'], color='orange',
                label='test')

    #resultsRdirect=Path('/home/jovyan/DSC650/dsc650/assignments/assignment06/
↪').joinpath('results')
    filedAway=myresults.joinpath('6.2A_Summary_Plot.png')
    plt.savefig(filedAway)
    plt.close()
```

[102]:
```
def defineDAmodel():
    model=Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu',
                     kernel_initializer='he_uniform',
                     padding='same',
                     input_shape=(32, 32, 3)))
    model.add(Conv2D(32, (3, 3), activation='relu',
                     kernel_initializer='he_uniform',
                     padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu',
                     kernel_initializer='he_uniform',
                     padding='same'))
    model.add(Conv2D(64, (3, 3), activation='relu',
                     kernel_initializer='he_uniform',
                     padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu',
                     kernel_initializer='he_uniform',
                     padding='same'))
    model.add(Conv2D(128, (3, 3), activation='relu',
```

```
                    kernel_initializer='he_uniform',
                    padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu',
                    kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    from keras.optimizers import SGD
    optimum=SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=optimum,
                  metrics=['accuracy'],
                  loss='categorical_crossentropy')
    return model
```

[103]:
```
def loader(filename):
    img=tf.keras.utils.load_img(filename, target_size=(32, 32))
    img=img_to_array(img)
    img=img.reshape(1, 32, 32, 3)

    img=img.astype('float32')
    img=img/255.0
    return img
```

[104]:
```
def plotdaconfusion(cm, classes,
                    normalize=False,
                    cmap=plt.cm.Blues):
    import itertools
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title("Da Confusion Matrix")
    plt.colorbar()
    tickSmarked = np.arange(len(classes))
    plt.xticks(tickSmarked, classes, rotation=45)
    plt.yticks(tickSmarked, classes)

    if normalize:
        cm=cm.astype('float') / cm.sum(axis=1)[:,np.newaxis]

    thresholds=cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
                                  range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresholds else "black")

        plt.tight_layout()
        plt.ylabel('True')
        plt.xlabel('Predicted')
```

```
    #resultsRdirect=Path('home/jovyan/DSC650/dsc650/assignments/assignment06/').
↪joinpath('results')
    imged_file=myresults.joinpath('6.2A_Confusion.png')
    plt.savefig(imged_file)
    plt.show()
```

[105]:
```
def run_test_harness():
    classy = ('airplane', 'automobile', 'bird', 'cat', 'deer',
              'dog', 'frog', 'horse', 'ship', 'truck')

    print("loading...")
    x_train, trainY, x_test, testY = load_dataset()
    print("Time: %s seconds" % (time.time() - start_time))

    print("preparing...")
    x_train, x_test=prep_pixels(x_train, x_test)

    for i in range(9):
        plt.subplot(330 + 1 + i)
        z = x_train[i]
        z = np.reshape(z, (32, 32, 3))
        plt.imshow(z)

    #resultsRdirect=Path('/home/jovyan/DSC650/dsc650/assignments/assignment06/
↪').joinpath('results')
    img_files = myresults.joinpath('6.2A_Sampled_CIFAR.png')
    plt.savefig(img_files)
    plt.show()
    print("Time: %s seconds" % (time.time() - start_time))

    print("defining...")
    model = defineDAmodel()
    summary_file = myresults.joinpath('6.2A_ModelSummary.txt')
    with open(summary_file, 'w') as f:
        with redirect_stdout(f):
            model.summary()
    print("Time: %s seconds" % (time.time() - start_time))

    print("fitting...")
    history = model.fit(x_train, trainY, epochs=20,
                        batch_size=64,
                        validation_data=(x_test, testY),
                        verbose=0)
    print("Time: %s seconds" % (time.time() - start_time))

    print("evaluating...")
```

```
    _, accuracy = model.evaluate(x_test, testY, verbose=0)
    print('> %.2f' % (accuracy*100.0))
    print("Time: %s seconds" % (time.time() - start_time))

    #resultsRdirect = Path('home/jovyan/DSC650/dsc650/assignments/assignment06/
↪')).joinpath('results')
    resultsOmodel_file = myresults.joinpath('6.2A_model.h5')
    model.save(resultsOmodel_file)
    print("I've Saved the trained model at %s " % resultsOmodel_file)

    print("preparing diagnosis summary")
    sum_diagnostics(history)
    print("Time: %s seconds" % (time.time() - start_time))

    print("predicting... ")
    Y_pred = model.predict(x_test)
    Y_pred_classes = np.argmax(Y_pred, axis=1)
    Y_true = np.argmax(testY, axis=1)
    from sklearn.metrics import confusion_matrix
    confusion=confusion_matrix(Y_true, Y_pred_classes)
    plotdaconfusion(confusion, classes=range(10))
    print("Time: %s seconds" % (time.time() - start_time))

    print("preparing classifications")
    classes=('airplane', 'automobile', 'bird', 'cat', 'deer',
             'dog', 'frog', 'horse', 'ship', 'truck')
    correct_in = np.nonzero(Y_pred_classes == Y_true)[0]
    incorrect_in = np.nonzero(Y_pred_classes != Y_true)[0]
    print(len(Y_pred_classes))
    print(len(Y_true))
    print(len(correct_in)," correctly classed")
    print(len(incorrect_in)," incorrectly classed")

    plt.rcParams['figure.figsize'] = (7, 14)
    figure_evaluation = plt.figure()

    for i, correct in enumerate(correct_in[:14]):
        plt.subplot(6, 3, i + 1)
        plt.imshow(x_test[correct], cmap='gray',
                   interpolation='none')
        plt.title("Predicted: {}, True: {}".format(
            classes[Y_pred[correct].argmax()],
            classes[testY[correct].argmax()]))
        plt.xticks([])
        plt.yticks([])

    images_file =myresults.joinpath('6.2A_CorrectPredictions.png')
```

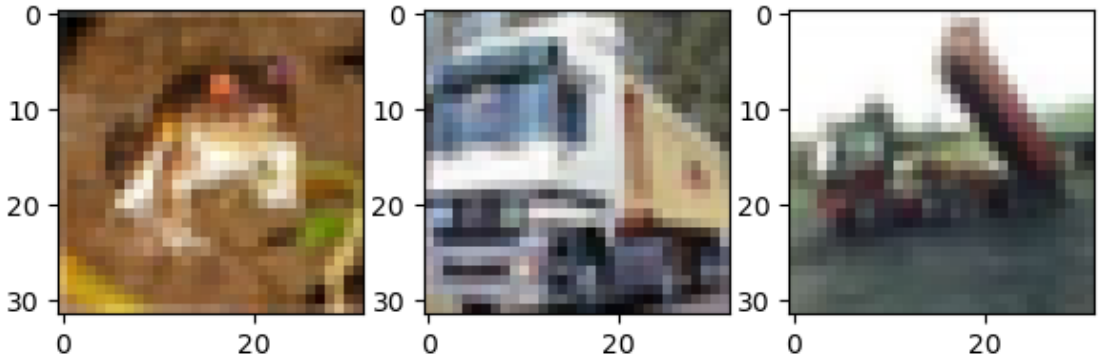
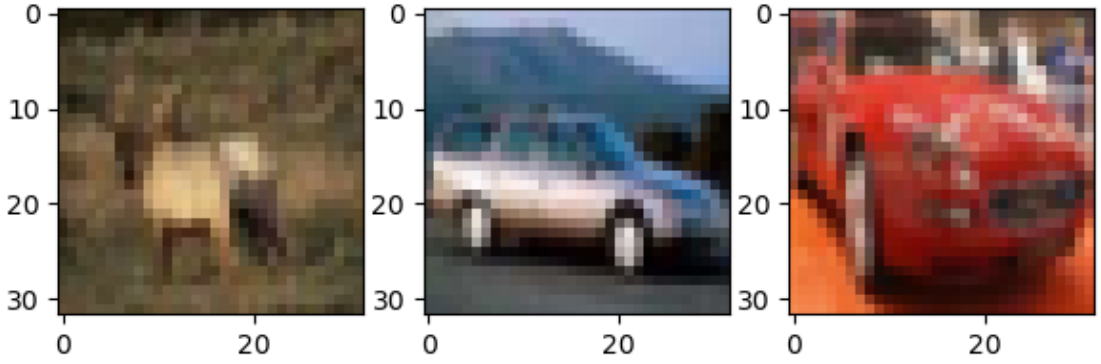
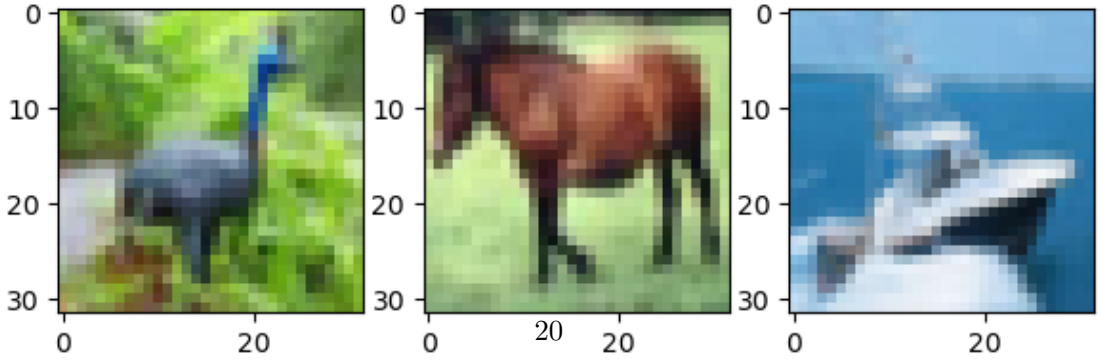
```
    plt.savefig(images_file)
    plt.show()


    for i, incorrect in enumerate(incorrect_in[:9]):
        plt.subplot(6, 3, i + 10)
        plt.imshow(x_test[incorrect], cmap='gray',
                   interpolation='none')
        plt.title(
            "Predicted {}, True: {}".format(
                classes[Y_pred[incorrect].argmax()],
                classes[testY[incorrect].argmax()]))
        plt.xticks([])
        plt.yticks([])

    images_file =myresults.joinpath('6.2A_IncorrectPredictions.png')
    plt.savefig(images_file)
    plt.show()
    print("Time: %s seconds" % (time.time() - start_time))
```

[106]:
```
def run_example_prediction():
    classes=('airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck')
    print("Attempting to predict image: 6.2A_Sampled_CIFAR.png")
    #resultsRdirect=Path('home/jovyan/DSC650/dsc650/assignments/assignment06/').
↪joinpath('results')
    result_model_file =myresults.joinpath('6.2A_model.h5')
    model=loader(result_model_file)
    summary_file=resultsRdirect.joinpath('6.2A_ModelSummaryLoaded.txt')
    with open(summary_file, 'w') as f:
        with redirect_stdout(f):
            model.summary()

    #resultsRdirect = Path('home/jovyan/DSC650/dsc650/assignments/assignment06/
↪').joinpath('results')
    filenamed =myresults.joinpath('6.2asample_image.png')
    img = loader(filenamed)

    result = model.predict_classes(img)
    print("The picture prediction is:.....")
    print(classes[result[0]])
    print("Time: %s seconds " % (time.time() - start_time))
```

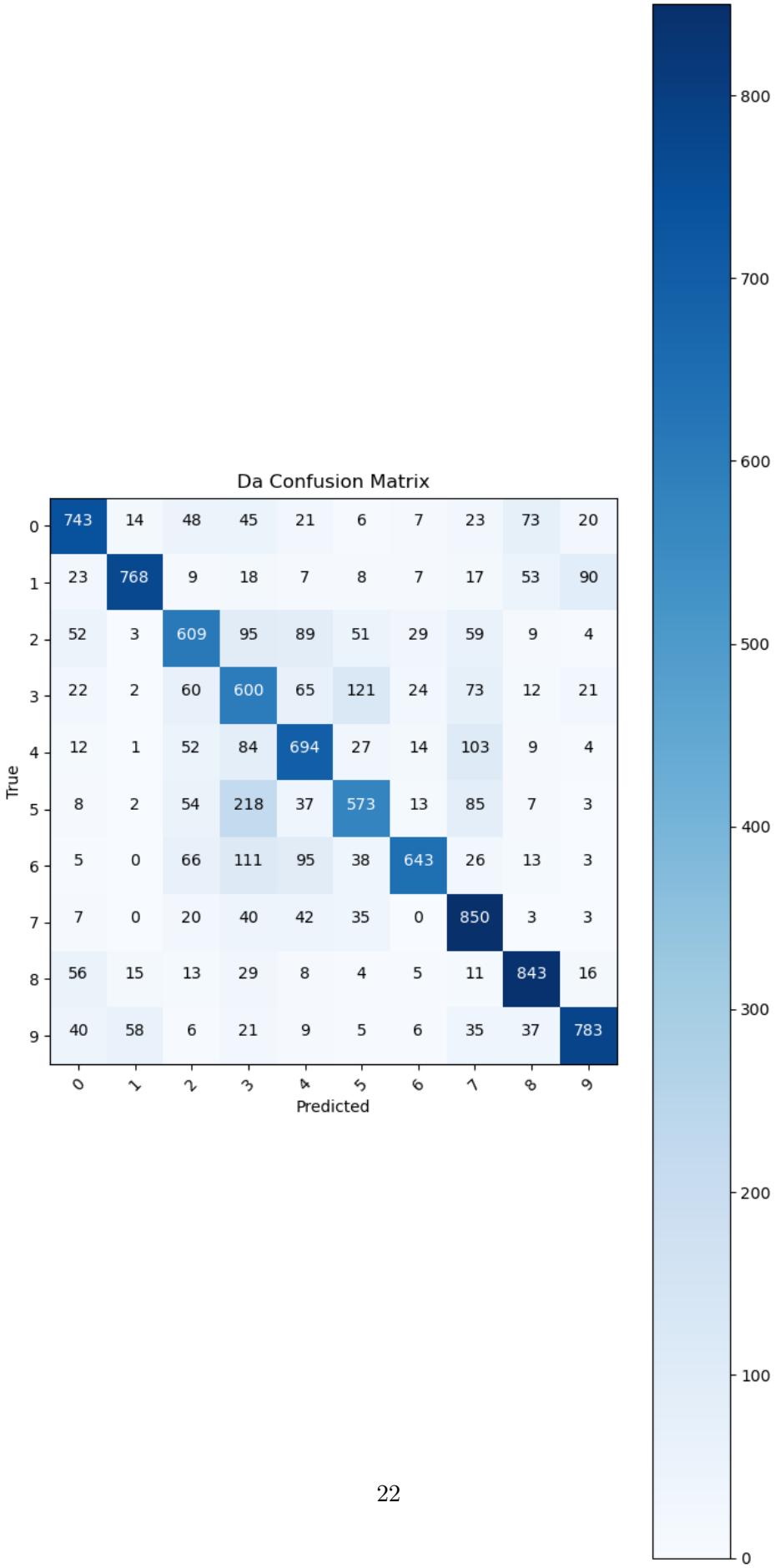
[107]:
```
run_test_harness()
```

```
loading…
Time: 53.20997881889343 seconds
```

```
preparing…
```

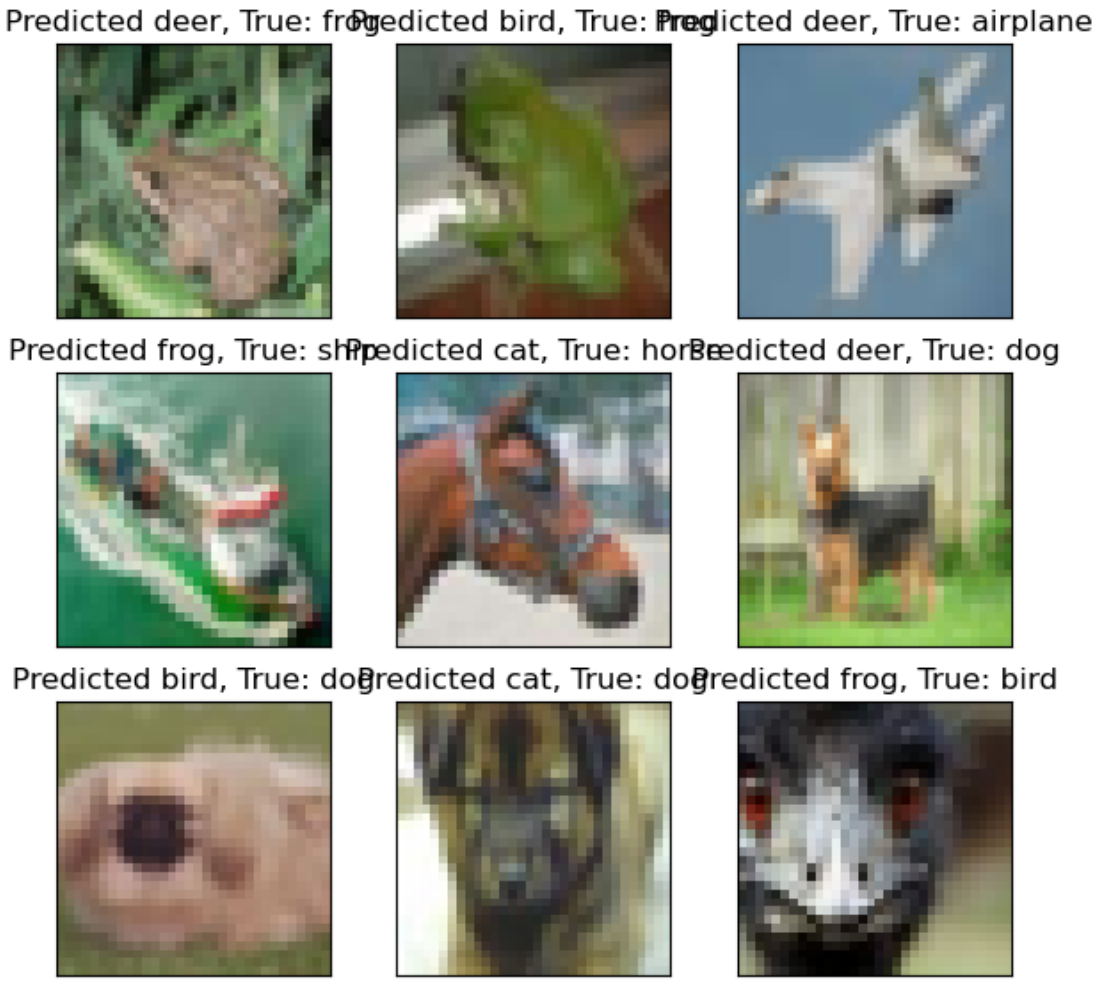




```
Time: 55.4164354801178 seconds
defining…
Time: 55.559701442718506 seconds
fitting…
Time: 945.1389605998993 seconds
evaluating…
> 71.06
Time: 949.323487997055 seconds
I've Saved the trained model at
/home/jovyan/DSC650/dsc650/assignments/assignment06/results/6.2A_model.h5
preparing diagnosis summary
Time: 949.7736599445343 seconds
predicting…
313/313 [==============================] - 4s 12ms/step
```


Da Confusion Matrix
True
Predicted
0 1 2 3 4 5 6 7 8 9
743 14 48 45 21 6 7 23 73 20
23 768 9 18 7 8 7 17 53 90
52 3 609 95 89 51 29 59 9 4
22 2 60 600 65 121 24 73 12 21
12 1 52 84 694 27 14 103 9 4
8 2 54 218 37 573 13 85 7 3
5 0 66 111 95 38 643 26 13 3
7 0 20 40 42 35 0 850 3 3
56 15 13 29 8 4 5 11 843 16
40 58 6 21 9 5 6 35 37 783
800
700
600
500
400
300
200
100
0

```
Time: 961.192111492157 seconds
preparing classifications
10000
10000
7106  correctly classed
2894  incorrectly classed
```


Predicted: cat, True: cat
Predicted: ship, True: ship
Predicted: ship, True: ship
Predicted: airplane, True: airplane
Predicted: frog, True: frog
Predicted: automobile, True: automobile
Predicted: cat, True: cat
Predicted: automobile, True: automobile
Predicted: truck, True: truck
Predicted: dog, True: dog
Predicted: horse, True: horse
Predicted: truck, True: truck
Predicted: dog, True: dog
Predicted: ship, True: ship




```
Time: 963.8997311592102 seconds
```

[108]:
```
def run_example_prediction():
    classes=('airplane', 'automobile', 'bird', 'cat', 'deer',
             'dog', 'frog', 'horse', 'ship', 'truck')
    print("Attempting to predict image: 6.2A_Sampled_CIFAR.png")

    result_model_file=myresults.joinpath('6.2A_model.h5')
    from keras.models import load_model
    model=load_model(result_model_file)
    summary_file=myresults.joinpath('6.2A_ModelSummaryLoaded.txt')
    with open(summary_file, 'w') as f:
        with redirect_stdout(f):
            model.summary()
```

```
    filenamed=myresults.joinpath('6.2asample_image.png')
    img = loader(filenamed)

    results=model.predict_classes(img)
    print("The picture prediction is:.....")
    print(classes[results[0]])
    print("Time: %s seconds " % (time.time() - start_time))
```

[ ]:

[ ]: