lab_9_1

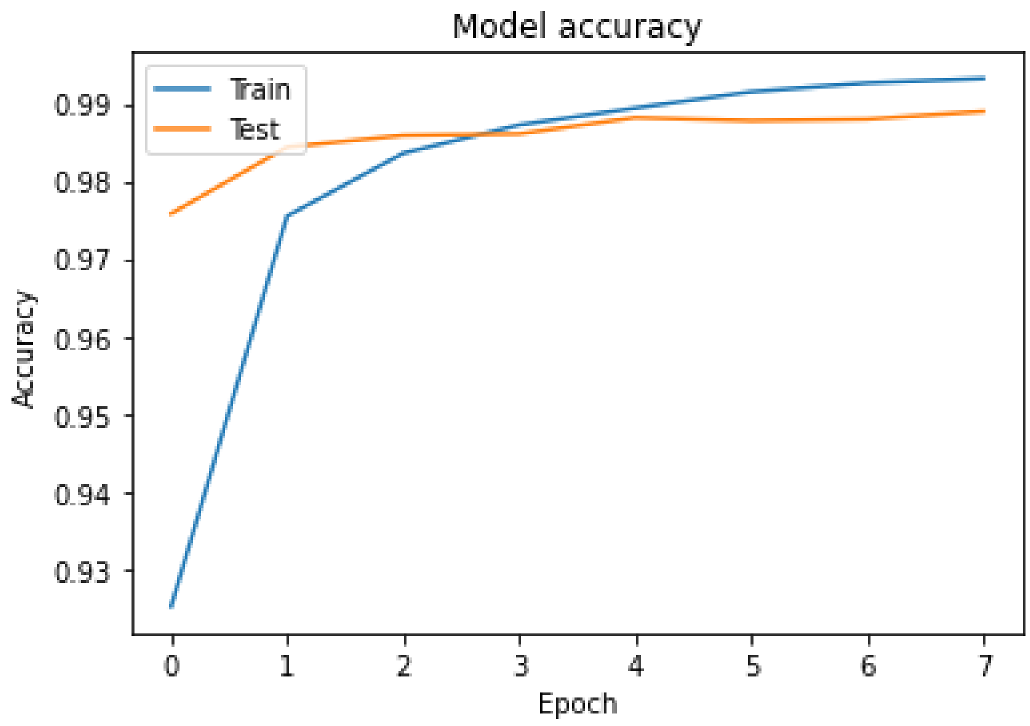
December 18, 2020

[1]:
```
# -*- coding: utf-8 -*-
"""
Running the example, the accuracy on the training and validation test is␣
 ↪printed
each epoch and at the end of the classification error rate is printed.

Note: Your results may vary given the stochastic nature of the algorithm or
evaluation procedure, or differences in numerical precision. Consider running␣
 ↪the example
a few times and compare the average outcome.

Epochs may take about 45 seconds to run on the GPU (e.g. on AWS).
You can see that the network achieves an error rate of 0.95%, which is better␣
 ↪than
our simple multi-layer perceptron model above.
"""

# Simple CNN for the MNIST Dataset
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from tensorflow.python.keras.callbacks import EarlyStopping
from numpy import argmax
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# reshape to be [samples][width][height][channels]
X_train = X_train.reshape((X_train.shape[0], 28, 28, 1)).astype('float32')
X_test = X_test.reshape((X_test.shape[0], 28, 28, 1)).astype('float32')
```

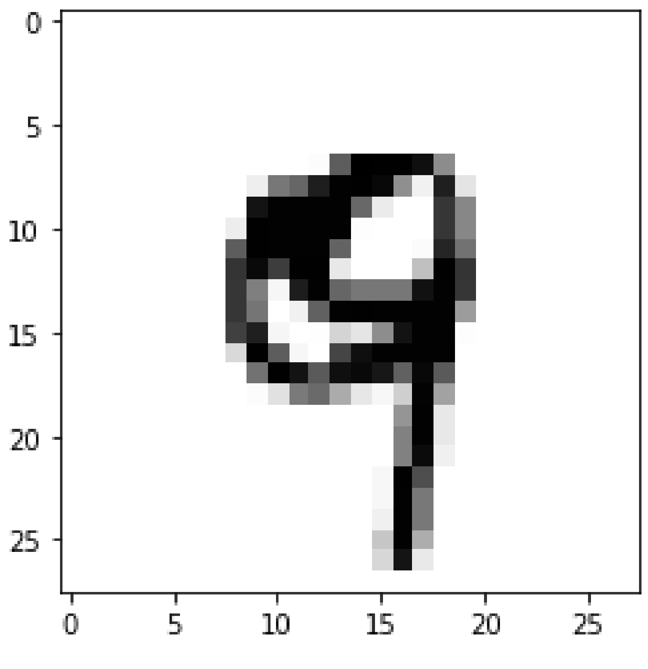
```python
# normalize inputs from 0-255 to 0-1
X_train = X_train / 255
X_test = X_test / 255

# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
# define a simple CNN model
def baseline_model():
        # create model
        model = Sequential()
        model.add(Conv2D(32, (5, 5), input_shape=(28, 28, 1),␣
 ↪activation='relu'))
        model.add(MaxPooling2D())
        model.add(Dropout(0.2))
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        model.add(Dense(num_classes, activation='softmax'))
        # Compile model
        model.compile(loss='categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
        return model

# build the model
model = baseline_model()

# simple early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)

# Fit the model
history=model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
 ↪epochs=10, batch_size=200, callbacks=[es])

#Process ploting
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='upper left')
plt.show()

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

```python
#make individual predictions
image_index = 4444
plt.imshow(X_test[image_index].reshape(28, 28), cmap='Greys')
plt.show()
pred = model.predict(X_test[image_index].reshape(1, 28, 28, 1))
print(pred.argmax())

#load image
def pred_image(fileName):
    img = mpimg.imread(fileName)
    gr_img =  0.2989*img[:,:,0] + 0.5870*img[:,:,1] + 0.1140*img[:,:,2]
    gr_img = 255 - gr_img

    plt.imshow(gr_img,cmap='Greys')
    plt.show()
    gr_img = gr_img.reshape(1, 28, 28, 1)
    pred = model.predict(gr_img)
    print(f"The predicted result {pred.argmax()}")

pred_image("8.png")
pred_image("7.png")
```

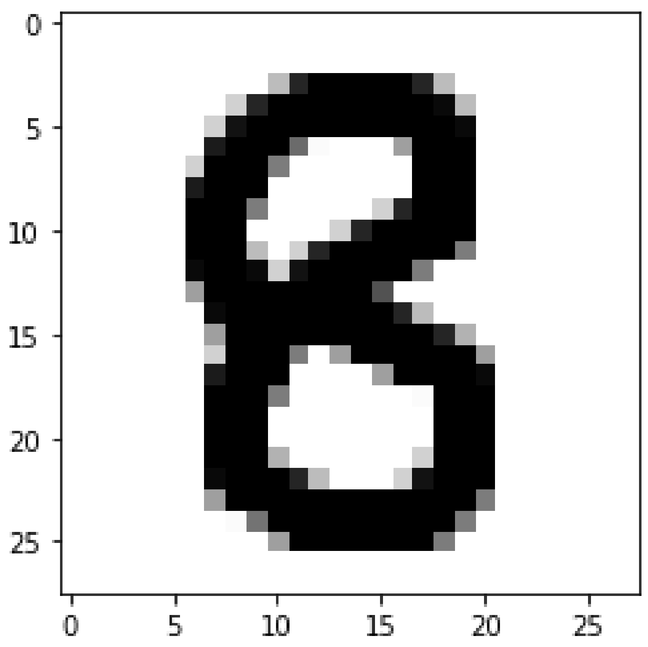
```
Epoch 1/10
300/300 [==============================] - 17s 57ms/step - loss: 0.2601 -
accuracy: 0.9251 - val_loss: 0.0817 - val_accuracy: 0.9758
Epoch 2/10
300/300 [==============================] - 17s 56ms/step - loss: 0.0796 -
accuracy: 0.9755 - val_loss: 0.0523 - val_accuracy: 0.9844
Epoch 3/10
300/300 [==============================] - 17s 56ms/step - loss: 0.0543 -
accuracy: 0.9836 - val_loss: 0.0413 - val_accuracy: 0.9859
Epoch 4/10
300/300 [==============================] - 17s 56ms/step - loss: 0.0413 -
accuracy: 0.9872 - val_loss: 0.0405 - val_accuracy: 0.9861
Epoch 5/10
300/300 [==============================] - 17s 57ms/step - loss: 0.0337 -
accuracy: 0.9894 - val_loss: 0.0354 - val_accuracy: 0.9882
Epoch 6/10
300/300 [==============================] - 17s 57ms/step - loss: 0.0270 -
accuracy: 0.9915 - val_loss: 0.0341 - val_accuracy: 0.9878
Epoch 7/10
300/300 [==============================] - 17s 57ms/step - loss: 0.0231 -
accuracy: 0.9926 - val_loss: 0.0321 - val_accuracy: 0.9880
Epoch 8/10
300/300 [==============================] - 18s 61ms/step - loss: 0.0213 -
accuracy: 0.9932 - val_loss: 0.0325 - val_accuracy: 0.9890
Epoch 00008: early stopping
```



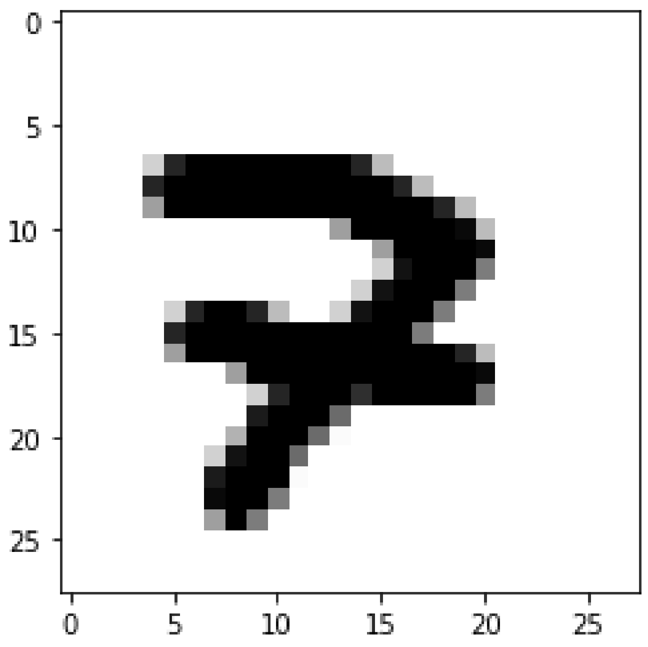
```
CNN Error: 1.10%
```



```
9
```



```
The predicted result 8
```



```
The predicted result 8
```