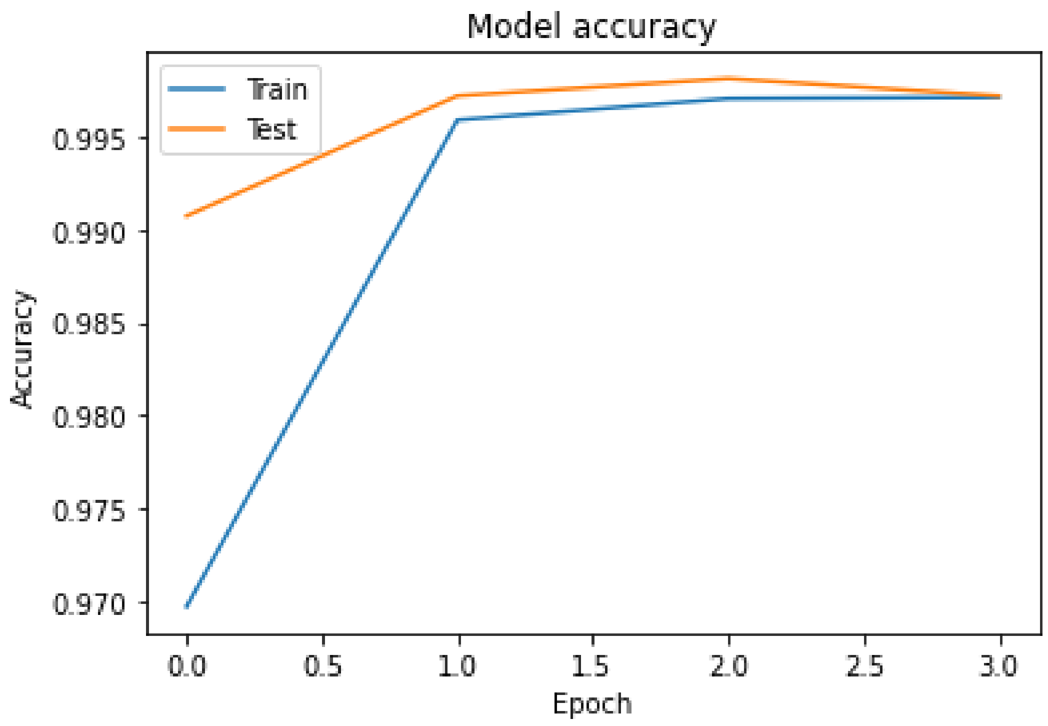
lab_9_13

December 18, 2020

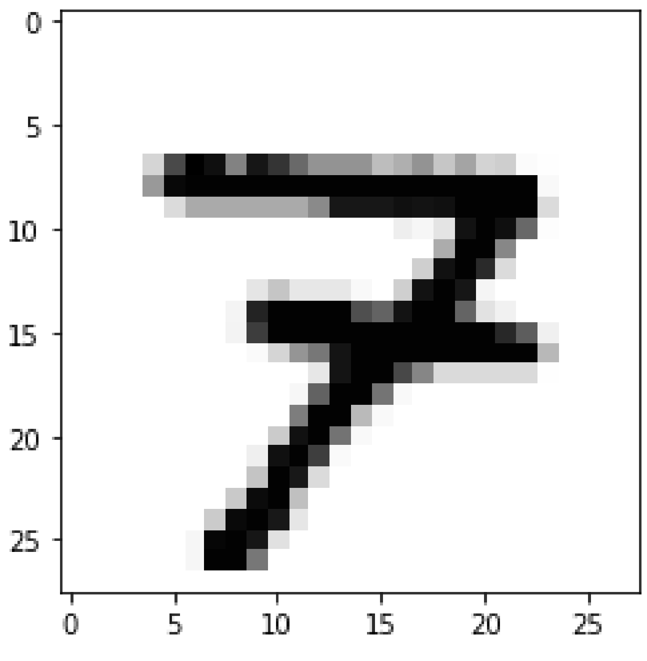
[1]:
```python
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from tensorflow.python.keras.callbacks import EarlyStopping
from numpy import argmax
from numpy import array

# 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')

# 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)
# new dataset
def subset_keep_only_seven_one(x,y):
    lambda_return_to_value = lambda y_i: argmax(y_i, axis=None, out=None)
    lambda_second_parm_seven_or_one = lambda x: lambda_return_to_value(x[1]) ==␣
 ↪7 or lambda_return_to_value(x[1]) == 1
    data = list(zip(x , y))
    data = list(filter(lambda_second_parm_seven_or_one,data))
    x, y = list(zip(*data))
    x = array(x)
    y = array(y)
    return x,y

# subset dataSet
X_train, y_train = subset_keep_only_seven_one(X_train, y_train)
X_test, y_test = subset_keep_only_seven_one(X_test,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 = 17
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("7.png")
```

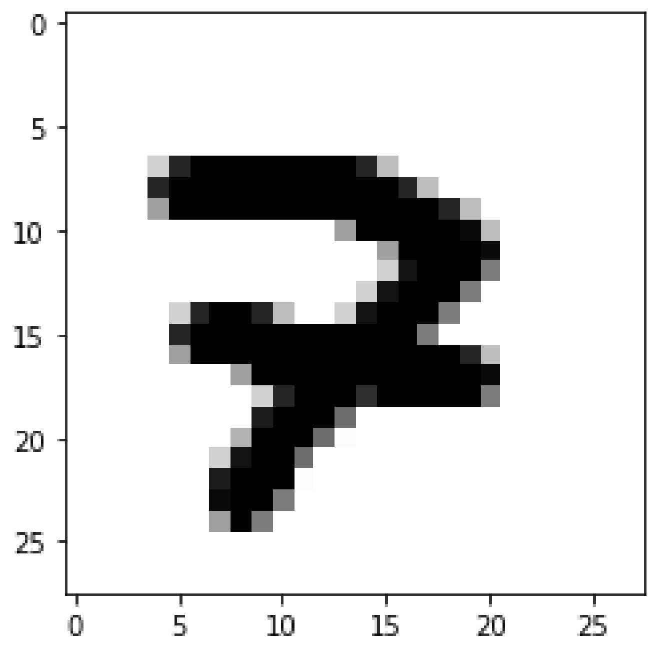
```
Epoch 1/10
66/66 [==============================] - 4s 55ms/step - loss: 0.1335 - accuracy:
0.9697 - val_loss: 0.0265 - val_accuracy: 0.9908
Epoch 2/10
66/66 [==============================] - 3s 52ms/step - loss: 0.0143 - accuracy:
0.9959 - val_loss: 0.0093 - val_accuracy: 0.9972
Epoch 3/10
66/66 [==============================] - 3s 52ms/step - loss: 0.0094 - accuracy:
0.9971 - val_loss: 0.0082 - val_accuracy: 0.9982
Epoch 4/10
66/66 [==============================] - 4s 54ms/step - loss: 0.0089 - accuracy:
0.9972 - val_loss: 0.0088 - val_accuracy: 0.9972
Epoch 00004: early stopping
```



```
CNN Error: 0.28%
```



```
7
```



```
The predicted result 1
```