Lesson 16 (Early Stopping MNIST) Load and preprocess the dataset `MNIST_train_1000.npg`. For the following questions, experiment with two optimizers: Adam and SGD.

(a) Which is a more accuracte measure of loss: (i) accuracy rate or (ii) cross-entropy. Explain.

(b) Establish a baseline accuracy rate. Then use Keras and logistic regression to predict the digit from its image. Use 100 epochs and a 20% validation set split. Identify the epoch having the lowest validation loss.

(c) Add a dense layer of 64 nodes and repeat part (b).

(d) Use early stopping to regularize your network and repeat part(c).

(e) Design a "deep" network of multiple dense layers with decreasing numbers of nodes, e.g. 32, 16, 8, 4, 2. Repeat part (d).

(f) Load and preprocess the dataset `MNIST_test_1000.npg`. What is the accuracy rate of your final network design on the test set? Display images of all the incorrect predictions. How many of the incorrect predictions would a human have correctly labeled? Use this number to establish an accuracy rate for human performance for the MNIST labeling task. (For simplicity, assume a human would have correctly labeled any correctly labeled image.)

```
import numpy as np
import pandas as pd
from   keras.models import Sequential
from   keras.layers.core import Dense, Activation
from   keras.optimizers import Adam
from   keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
%matplotlib inline

results = pd.DataFrame()
results['epoch']           = hist.epoch
results['epoch']           = results['epoch'] + 1
results['training loss']   = hist.history['loss']
results['validation loss'] = np.sqrt(hist.history['val_loss'])
results['training acc']    = hist.history['acc']
results['validation acc']  = np.sqrt(hist.history['val_acc'])

ix = results['validation loss'].idxmin()
ce_training   = results['training loss'].iloc[ix]
ce_validation = results['validation loss'].iloc[ix]
acc_training   = results['training acc'].iloc[ix]
acc_validation = results['validation acc'].iloc[ix]
print()
print('minimum validation loss index',ix,'of',epochs)
print('cross-entropy')
print('        training =',ce_training)
print('      validation =',ce_validation)
print('accuracy rate')
print('        training =',acc_training)
print('      validation =',acc_validation)
print('        baseline =',acc_baseline)
```

(continued on back side)

```
ax = results.plot.line(x='epoch',y='validation loss')
results.plot.line(x='epoch',y='training loss',ax=ax)
results.plot.line(x='epoch',y='validation loss')

hist = model.fit(X,P,epochs=epochs,validation_split=0.2,verbose=0,
                    callbacks=[EarlyStopping(patience=patience)])

Ph_test = model.predict(X_test)
ix_errors = (Ph_test.argmax(axis=1) != P_test.argmax(axis=1))
images_errors = images_train[ix_errors,:,:]

# image grid code
N = 7
M = 7
plt.figure(figsize=(10,10))
for i in range(len(images_errors)):
    plt.subplot(M, N, i+1)
    plt.imshow(images_errors[i,:,:], cmap='Greys_r')
    plt.title(labels_train[i])
plt.show()
```

(Image grid code curtesy Professor Yoder.)