```
'''Train a simple deep NN on the MNIST dataset.
Get to 98.40% test accuracy after 20 epochs
(there is *a lot* of margin for parameter tuning).
2 seconds per epoch on a K520 GPU.
'''

  from __future__ import print_function
  import numpy as np
  np.random.seed(1337)  # for reproducibility

  from keras.datasets import mnist
  from keras.models import Sequential
  from keras.layers.core import Dense, Dropout, Activation
  from keras.optimizers import SGD, Adam, RMSprop
  from keras.utils import np_utils
  import keras.backend as K
  from itertools import product

  # Custom loss function with costs
  def w_categorical_crossentropy(y_true, y_pred, weights):
      nb_cl = len(weights)
      final_mask = K.zeros_like(y_pred[:, 0])
      y_pred_max = K.max(y_pred, axis=1)
      y_pred_max = K.expand_dims(y_pred_max, 1)
      y_pred_max_mat = K.equal(y_pred, y_pred_max)
      for c_p, c_t in product(range(nb_cl), range(nb_cl)):

          final_mask += (K.cast(weights[c_t, c_p],K.floatx()) *
K.cast(y_pred_max_mat[:, c_p] ,K.floatx())* K.cast(y_true[:,
c_t],K.floatx()))
      return K.categorical_crossentropy(y_pred, y_true) * final_mask
w_array = np.ones((3,3))
w_array[2,1] = 1.2
w_array[1,2] = 1.2
ncce = partial(w_categorical_crossentropy, weights=w_array)
ncce.__name__ ='w_categorical_crossentropy'

batch_size = 128
nb_classes = 10
nb_epoch = 20

# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))

rms = RMSprop()
model.compile(loss=ncce, optimizer=rms)

model.fit(X_train, Y_train,
          batch_size=batch_size, nb_epoch=nb_epoch,
          show_accuracy=True, verbose=1,
          validation_data=(X_test, Y_test))
score = model.evaluate(X_test, Y_test,
                       show_accuracy=True, verbose=1)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```