ICP11

Autoencoders::

File Code ad out put follows it.

from keras.layers import Input, Dense

from keras.models import Model

#load MNIST handwritten digits dataset

from keras.datasets import mnist

import numpy as np

import matplotlib.pyplot as plt

#Notice that we’re not loading any of the labels because autoencoders are unsupervised

(X\_train, \_), (X\_test, \_) = mnist.load\_data()

# rescale our images from 0 – 255 to 0 – 1 and flatten them out.

X\_train = X\_train.astype('float32') / 255.

X\_test = X\_test.astype('float32') / 255.

X\_train = X\_train.reshape((X\_train.shape[0], -1))

X\_test = X\_test.reshape((X\_test.shape[0], -1))

#constants for our input size and our encoding size.

INPUT\_SIZE = 784

ENCODING\_SIZE = 64

#educe our input from 784 -> 512 -> 256 -> 128 -> 64 (encoder path), then expand it back up 64 -> 128 -> 256 -> 512 -> 784(decoder path).

# Also notice the relu activation function

input\_img = Input(shape=(INPUT\_SIZE,))

encoded = Dense(512, activation='relu')(input\_img)

encoded = Dense(256, activation='relu')(encoded)

encoded = Dense(128, activation='relu')(encoded)

encoded = Dense(ENCODING\_SIZE, activation='relu')(encoded)

decoded = Dense(128, activation='relu')(encoded)

decoded = Dense(256, activation='relu')(decoded)

decoded = Dense(512, activation='relu')(decoded)

decoded = Dense(INPUT\_SIZE, activation='relu')(decoded)

autoencoder = Model(input\_img, decoded)

#using ADAM optimizer and mean squared error loss (the Euclidean distance/loss) between the input and reconstruction

autoencoder.compile(optimizer='adam', loss='mean\_squared\_error')

autoencoder.fit(X\_train, X\_train, epochs=50, batch\_size=256, shuffle=True, validation\_split=0.2)

#After our autoencoder has trained, we can try to encode and decode the test set to see how well our autoencoder can compress

decoded\_imgs = autoencoder.predict(X\_test)

#Finally, we can visualize our true values and reconstructions using matplotlib

plt.figure(figsize=(20, 4))

for i in range(10):

    # original

    plt.subplot(2, 10, i + 1)

    plt.imshow(X\_test[i].reshape(28, 28))

    plt.gray()

    plt.axis('off')

    # reconstruction

    plt.subplot(2, 10, i + 1 + 10)

    plt.imshow(decoded\_imgs[i].reshape(28, 28))

    plt.gray()

    plt.axis('off')

plt.tight\_layout()

plt.show()

Using TensorFlow backend.

Downloading data from <https://s3.amazonaws.com/img-datasets/mnist.npz>

11493376/11490434 [==============================] - 1s 0us/step

Train on 48000 samples, validate on 12000 samples

Epoch 1/50

48000/48000 [==============================] - 9s 191us/step - loss: 0.0400 - val\_loss: 0.0222

Epoch 2/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0189 - val\_loss: 0.0166

Epoch 3/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0154 - val\_loss: 0.0144

Epoch 4/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0138 - val\_loss: 0.0137

Epoch 5/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0128 - val\_loss: 0.0129

Epoch 6/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0121 - val\_loss: 0.0120

Epoch 7/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0116 - val\_loss: 0.0117

Epoch 8/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0112 - val\_loss: 0.0112

Epoch 9/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0108 - val\_loss: 0.0109

Epoch 10/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0106 - val\_loss: 0.0108

Epoch 11/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0103 - val\_loss: 0.0106

Epoch 12/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0101 - val\_loss: 0.0103

Epoch 13/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0099 - val\_loss: 0.0100

Epoch 14/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0097 - val\_loss: 0.0097

Epoch 15/50

48000/48000 [==============================] - 9s 185us/step - loss: 0.0093 - val\_loss: 0.0094

Epoch 16/50

48000/48000 [==============================] - 9s 185us/step - loss: 0.0091 - val\_loss: 0.0093

Epoch 17/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0090 - val\_loss: 0.0092

Epoch 18/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0089 - val\_loss: 0.0092

Epoch 19/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0088 - val\_loss: 0.0091

Epoch 20/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0087 - val\_loss: 0.0091

Epoch 21/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0087 - val\_loss: 0.0092

Epoch 22/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0086 - val\_loss: 0.0088

Epoch 23/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0085 - val\_loss: 0.0090

Epoch 24/50

48000/48000 [==============================] - 9s 180us/step - loss: 0.0084 - val\_loss: 0.0088

Epoch 25/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0084 - val\_loss: 0.0087

Epoch 26/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0083 - val\_loss: 0.0087

Epoch 27/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0082 - val\_loss: 0.0087

Epoch 28/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0082 - val\_loss: 0.0086

Epoch 29/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0081 - val\_loss: 0.0084

Epoch 30/50

48000/48000 [==============================] - 9s 181us/step - loss: 0.0081 - val\_loss: 0.0083

Epoch 31/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0081 - val\_loss: 0.0086

Epoch 32/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0080 - val\_loss: 0.0083

Epoch 33/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0079 - val\_loss: 0.0083

Epoch 34/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0079 - val\_loss: 0.0081

Epoch 35/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0079 - val\_loss: 0.0084

Epoch 36/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0078 - val\_loss: 0.0083

Epoch 37/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0078 - val\_loss: 0.0082

Epoch 38/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0078 - val\_loss: 0.0080

Epoch 39/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0077 - val\_loss: 0.0082

Epoch 40/50

48000/48000 [==============================] - 9s 185us/step - loss: 0.0077 - val\_loss: 0.0080

Epoch 41/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0076 - val\_loss: 0.0081

Epoch 42/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0076 - val\_loss: 0.0080

Epoch 43/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0076 - val\_loss: 0.0080

Epoch 44/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0076 - val\_loss: 0.0079

Epoch 45/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0075 - val\_loss: 0.0080

Epoch 46/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0075 - val\_loss: 0.0080

Epoch 47/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0075 - val\_loss: 0.0079

Epoch 48/50

48000/48000 [==============================] - 9s 183us/step - loss: 0.0075 - val\_loss: 0.0079

Epoch 49/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0074 - val\_loss: 0.0078

Epoch 50/50

48000/48000 [==============================] - 9s 182us/step - loss: 0.0074 - val\_loss: 0.0079

