DL exp image deoising

Deep Learning (University of Mumbai)

DL exp image deoising

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
[1]  #import
     import numpy
     import matplotlib.pyplot as plt
     from keras.models import Sequential
     from keras.layers import Dense
     from keras.datasets import mnist
```

```
[2]  (X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

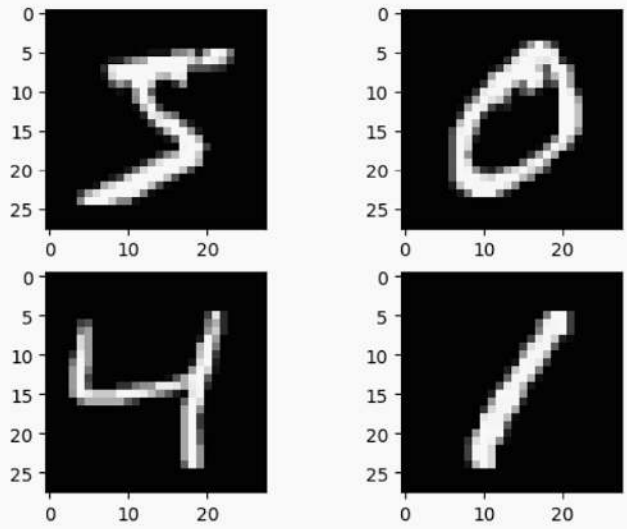
```
[3]  X_train.shape
```

```
(60000, 28, 28)
```

```
[4]  X_test.shape
```

```
(10000, 28, 28)
```

```
[5]  plt.subplot(221)
     plt.imshow(X_train[0], cmap=plt.get_cmap('gray'))
     plt.subplot(222)
     plt.imshow(X_train[1], cmap=plt.get_cmap('gray'))
     plt.subplot(223)
     plt.imshow(X_train[2], cmap=plt.get_cmap('gray'))
     plt.subplot(224)
     plt.imshow(X_train[3], cmap=plt.get_cmap('gray'))
     # show the plot
     plt.show()
```




```
[6]  num_pixels = X_train.shape[1] * X_train.shape[2]
     X_train = X_train.reshape(X_train.shape[0], num_pixels).astype('float32')
     X_test = X_test.reshape(X_test.shape[0], num_pixels).astype('float32')
     X_train = X_train / 255
     X_test = X_test / 255
```

```
[7]  X_train.shape
```

```
(60000, 784)
```

```
[8]  X_test.shape
```

```
(10000, 784)
```

```
[9] noise_factor = 0.2
    x_train_noisy = X_train + noise_factor * numpy.random.normal(loc=0.0, scale=1.0, size=X_train.shape)
    x_test_noisy = X_test + noise_factor * numpy.random.normal(loc=0.0, scale=1.0, size=X_test.shape)
    x_train_noisy = numpy.clip(x_train_noisy, 0., 1.)
    x_test_noisy = numpy.clip(x_test_noisy, 0., 1.)
```

```
[10] # create model
     model = Sequential()
     model.add(Dense(500, input_dim=num_pixels, activation='relu'))
     model.add(Dense(300, activation='relu'))
     model.add(Dense(100, activation='relu'))
     model.add(Dense(300, activation='relu'))
     model.add(Dense(500, activation='relu'))
     model.add(Dense(784, activation='sigmoid'))
```

```
[11] # Compile the model
     model.compile(loss='mean_squared_error', optimizer='adam')
```

```
[12] # Training model
     model.fit(x_train_noisy, X_train, validation_data=(x_test_noisy, X_test), epochs=2, batch_size=200)
```

```
Epoch 1/2
300/300 [==============================] - 16s 50ms/step - loss: 0.0415 - val_loss: 0.0191
Epoch 2/2
300/300 [==============================] - 15s 49ms/step - loss: 0.0160 - val_loss: 0.0133
<keras.src.callbacks.History at 0x79f695ce8df0>
```

```
[13] # Final evaluation of the model
     pred = model.predict(x_test_noisy)
     pred.shape
```

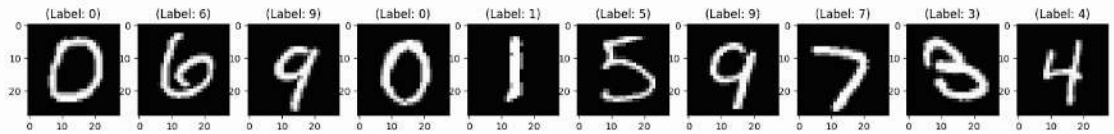
```
313/313 [==============================] - 2s 6ms/step
(10000, 784)
```

```
[14] X_test.shape
```

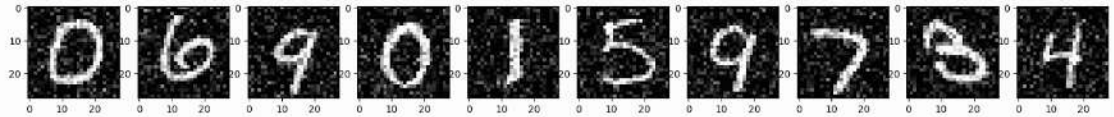
```
(10000, 784)
```

```
X_test = numpy.reshape(X_test, (10000,28,28)) *255
pred = numpy.reshape(pred, (10000,28,28)) *255
x_test_noisy = numpy.reshape(x_test_noisy, (-1,28,28)) *255
plt.figure(figsize=(20, 4))
print("Test Images")
for i in range(10,20,1):
    plt.subplot(2, 10, i+1)
    plt.imshow(X_test[i,:,:], cmap='gray')
    curr_lbl = y_test[i]
    plt.title("(Label: " + str(curr_lbl) + ")")
plt.show()
plt.figure(figsize=(20, 4))
print("Test Images with Noise")
for i in range(10,20,1):
    plt.subplot(2, 10, i+1)
    plt.imshow(x_test_noisy[i,:,:], cmap='gray')
plt.show()
plt.figure(figsize=(20, 4))
print("Reconstruction of Noisy Test Images")
for i in range(10,20,1):
    plt.subplot(2, 10, i+1)
    plt.imshow(pred[i,:,:], cmap='gray')
plt.show()
```

```
Test Images
```




```
Test Images with Noise
```



```
Reconstruction of Noisy Test Images
```