Untitled4

November 17, 2023

[3]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPooling2D,␣
 ↪UpSampling2D
from tensorflow.keras.models import Model
import pandas as pd
```

[4]:
```
data=pd.read_csv('mnist.csv')
data.head()
```

[4]:

	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	\
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	

	pixel10	…	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	\
0	0	…	0	0	0	0	0	0	
1	0	…	0	0	0	0	0	0	
2	0	…	0	0	0	0	0	0	
3	0	…	0	0	0	0	0	0	
4	0	…	0	0	0	0	0	0	

	pixel782	pixel783	pixel784	class
0	0	0	0	5
1	0	0	0	0
2	0	0	0	4
3	0	0	0	1
4	0	0	0	9

[5 rows x 785 columns]

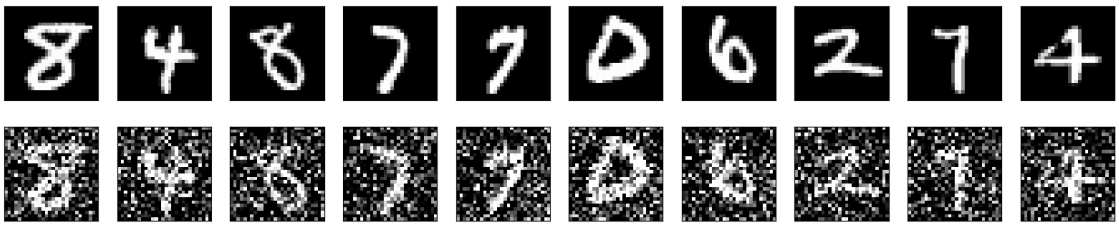
[7]:
```
x=data.drop(['class'],axis=1)
```

[9]:
```
from sklearn.model_selection import train_test_split
x_train,x_test=train_test_split(x,test_size=0.2,random_state=42)
```

[10]:
```
# Normalize pixel values to be between 0 and 1
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# Add random noise to the images
noise_factor = 0.5
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0, scale=1.0,␣
 ↪size=x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0,␣
 ↪size=x_test.shape)
```

[11]:
```
x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)
```

[16]:
```
# Display original and noisy images
n = 10  # Number of images to display
plt.figure(figsize=(20, 4))
for i in range(n):
    # Display original images
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')  # Specify the colormap
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display noisy images
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(x_test_noisy[i].reshape(28, 28), cmap='gray')  # Specify the␣
 ↪colormap
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

plt.show()
```



[17]:
```
# Define the autoencoder model
input_img = Input(shape=(28, 28, 1))
x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
```

```
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

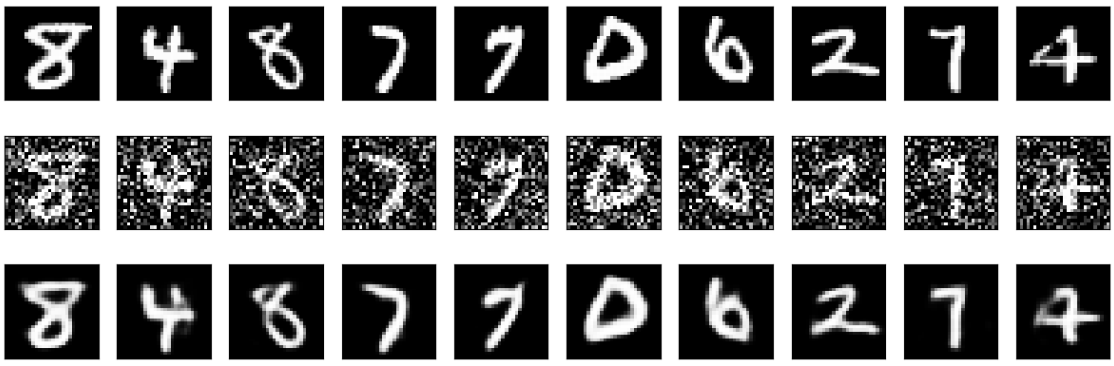
[18]:
```
# Reshape the data for convolutional autoencoder
x_train = np.reshape(x_train, (len(x_train), 28, 28, 1))
x_test = np.reshape(x_test, (len(x_test), 28, 28, 1))
x_train_noisy = np.reshape(x_train_noisy, (len(x_train_noisy), 28, 28, 1))
x_test_noisy = np.reshape(x_test_noisy, (len(x_test_noisy), 28, 28, 1))
```

[19]:
```
autoencoder.fit(x_train_noisy, x_train, epochs=10, batch_size=128,␣
↪shuffle=True, validation_data=(x_test_noisy, x_test))

# Denoise test images
denoised_images = autoencoder.predict(x_test_noisy)
```

```
Epoch 1/10
438/438 [==============================] - 124s 278ms/step - loss: 0.1602 -
val_loss: 0.1179
Epoch 2/10
438/438 [==============================] - 123s 281ms/step - loss: 0.1135 -
val_loss: 0.1098
Epoch 3/10
438/438 [==============================] - 133s 303ms/step - loss: 0.1083 -
val_loss: 0.1063
Epoch 4/10
438/438 [==============================] - 117s 266ms/step - loss: 0.1051 -
val_loss: 0.1035
Epoch 5/10
438/438 [==============================] - 88s 200ms/step - loss: 0.1031 -
val_loss: 0.1021
Epoch 6/10
438/438 [==============================] - 84s 193ms/step - loss: 0.1017 -
val_loss: 0.1009
Epoch 7/10
438/438 [==============================] - 89s 204ms/step - loss: 0.1005 -
val_loss: 0.0997
Epoch 8/10
438/438 [==============================] - 81s 186ms/step - loss: 0.0996 -
val_loss: 0.0989
```

```
Epoch 9/10
438/438 [==============================] - 89s 204ms/step - loss: 0.0988 -
val_loss: 0.0986
Epoch 10/10
438/438 [==============================] - 83s 189ms/step - loss: 0.0982 -
val_loss: 0.0979
438/438 [==============================] - 8s 18ms/step
```

[20]:
```python
# Display original, noisy, and denoised images
plt.figure(figsize=(15, 5))
for i in range(n):
    # Display original images
    ax = plt.subplot(3, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display noisy images
    ax = plt.subplot(3, n, i + 1 + n)
    plt.imshow(x_test_noisy[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display denoised images
    ax = plt.subplot(3, n, i + 1 + 2 * n)
    plt.imshow(denoised_images[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

plt.show()
```



[ ]: