**Module 17**

Write a code to denoise an image using autoencoders. You Have to use MNIST dataset and use salt and pepper noise to add noise to the image. This noisy image has to be denoised using autoencoders.

**Code:**

# -\*- coding: utf-8 -\*-

"""

Created on Thu May 16 21:00:44 2024

@author: Lenovo

"""

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import tensorflow as tf

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras import layers, losses

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Model

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

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

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

x\_train = x\_train[..., tf.newaxis]

x\_test = x\_test[..., tf.newaxis]

print(x\_train.shape)

noise\_factor = 0.2

x\_train\_noisy = x\_train + noise\_factor \* tf.random.normal(shape=x\_train.shape)

x\_test\_noisy = x\_test + noise\_factor \* tf.random.normal(shape=x\_test.shape)

x\_train\_noisy = tf.clip\_by\_value(x\_train\_noisy, clip\_value\_min=0., clip\_value\_max=1.)

x\_test\_noisy = tf.clip\_by\_value(x\_test\_noisy, clip\_value\_min=0., clip\_value\_max=1.)

n = 10

plt.figure(figsize=(20, 2))

for i in range(n):

ax = plt.subplot(1, n, i + 1)

plt.title("original + noise")

plt.imshow(tf.squeeze(x\_test\_noisy[i]))

plt.gray()

plt.show()

class Denoise(Model):

def \_\_init\_\_(self):

super(Denoise, self).\_\_init\_\_()

self.encoder = tf.keras.Sequential([

layers.Input(shape=(28, 28, 1)),

layers.Conv2D(16, (3, 3), activation='relu', padding='same', strides=2),

layers.Conv2D(8, (3, 3), activation='relu', padding='same', strides=2)])

self.decoder = tf.keras.Sequential([

layers.Conv2DTranspose(8, kernel\_size=3, strides=2, activation='relu', padding='same'),

layers.Conv2DTranspose(16, kernel\_size=3, strides=2, activation='relu', padding='same'),

layers.Conv2D(1, kernel\_size=(3, 3), activation='sigmoid', padding='same')])

def call(self, x):

encoded = self.encoder(x)

decoded = self.decoder(encoded)

return decoded

autoencoder = Denoise()

autoencoder.compile(optimizer='adam', loss=losses.MeanSquaredError())

autoencoder.fit(x\_train\_noisy, x\_train,

epochs=10,

shuffle=True,

validation\_data=(x\_test\_noisy, x\_test))

autoencoder.encoder.summary()

autoencoder.decoder.summary()

encoded\_imgs = autoencoder.encoder(x\_test\_noisy).numpy()

decoded\_imgs = autoencoder.decoder(encoded\_imgs).numpy()

n = 10

plt.figure(figsize=(20, 4))

for i in range(n):

# display original + noise

ax = plt.subplot(2, n, i + 1)

plt.title("original + noise")

plt.imshow(tf.squeeze(x\_test\_noisy[i]))

plt.gray()

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

# display reconstruction

bx = plt.subplot(2, n, i + n + 1)

plt.title("reconstructed")

plt.imshow(tf.squeeze(decoded\_imgs[i]))

plt.gray()

bx.get\_xaxis().set\_visible(False)

bx.get\_yaxis().set\_visible(False)

plt.show()

**Output:** 