**Practical 10**

**Denoising of images using autoencoder.**

**10**

**Aim:** **Denoising images using autoencoder.**

**Description:**

**autoencoder**

1. Unsupervised Learning: Autoencoders are a type of neural network used for unsupervised learning. They don't require labeled data for training. Instead, they learn by trying to reconstruct the input data itself.
2. Encoder-Decoder Structure: An autoencoder has two main parts: an encoder and a decoder. The encoder compresses the input data into a lower-dimensional representation, often called the latent space. This captures the essential features of the data.
3. Latent Space: The latent space is the compressed version of the input data created by the encoder. It's like a bottleneck that forces the network to learn efficient representations.
4. Reconstruction: The decoder then takes this latent space representation and tries to rebuild the original input data as accurately as possible.
5. Dimensionality Reduction: By forcing the data through the bottleneck of the latent space, autoencoders can learn to represent the data in a more compact way. This can be useful for tasks like data compression or anomaly detection.
6. Feature Extraction: The latent space representation can also be used as a new set of features for the data. These features can be helpful for other machine learning tasks like classification or clustering.
7. Variational Autoencoders (VAE): A variation of the autoencoder is the Variational Autoencoder (VAE). VAEs introduce randomness into the latent space, allowing them to learn more complex representations of the data.

**Code:**

# 10. Denoising of images using autoencoder.

import keras

from keras.datasets import mnist

from keras import layers

import numpy as np

from keras.callbacks import TensorBoard

import matplotlib.pyplot as plt

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

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

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

X\_train=np.reshape(X\_train,(len(X\_train),28,28,1))

X\_test=np.reshape(X\_test,(len(X\_test),28,28,1))

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)

X\_train\_noisy=np.clip(X\_train\_noisy,0.,1.)

X\_test\_noisy=np.clip(X\_test\_noisy,0.,1.)

n=10

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

for i in range(1,n+1):

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

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

plt.gray()

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

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

plt.show()

input\_img=keras.Input(shape=(28,28,1))

x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(input\_img)

x=layers.MaxPooling2D((2,2),padding='same')(x)

x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(x)

encoded=layers.MaxPooling2D((2,2),padding='same')(x)

x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(encoded)

x=layers.UpSampling2D((2,2))(x)

x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(x)

x=layers.UpSampling2D((2,2))(x)

decoded=layers.Conv2D(1,(3,3),activation='sigmoid',padding='same')(x)

autoencoder=keras.Model(input\_img,decoded)

autoencoder.compile(optimizer='adam',loss='binary\_crossentropy')

autoencoder.fit(X\_train\_noisy,X\_train, epochs=3, batch\_size=128, shuffle=True, validation\_data=(X\_test\_noisy,X\_test), callbacks=[TensorBoard(log\_dir='/tmo/tb',histogram\_freq=0,write\_graph=False)])

predictions=autoencoder.predict(X\_test\_noisy)

m=10

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

for i in range(1,m+1):

ax=plt.subplot(1,m,i)

plt.imshow(predictions[i].reshape(28,28))

plt.gray()

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

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

plt.show()

**Output:**



A screenshot of a computer

Description automatically generated

A black square with white numbers

Description automatically generated

**Learning:**

1. Denoising with Autoencoders: This code demonstrates using an autoencoder for image denoising. It adds artificial noise to the MNIST dataset and trains the autoencoder to remove the noise while reconstructing the original images.
2. Data Preprocessing: The code preprocesses the MNIST data by converting it to float32 format, normalizing the pixel values between 0 and 1, and reshaping it to include the channel dimension (as MNIST images are grayscale).
3. Convolutional Autoencoder Architecture: The autoencoder uses a convolutional neural network (CNN) architecture with alternating convolutional and pooling layers in the encoder and upsampling layers in the decoder. This allows the network to learn spatial features of the images.
4. Training and Validation: The code trains the autoencoder on the noisy training data with the Adam optimizer and binary crossentropy loss function. It also uses a validation set to monitor performance on unseen noisy data.
5. Visualization: The code visualizes both the noisy test images and the denoised predictions from the autoencoder. This allows qualitative assessment of the model's ability to remove noise.