Roll Number: 210701070

Exp No: 7

BUILD AUTOENCODERS WITH KERAS/TENSORFLOW

Aim:
To build autoencoders with Keras/TensorFlow.

Procedure:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

Program:

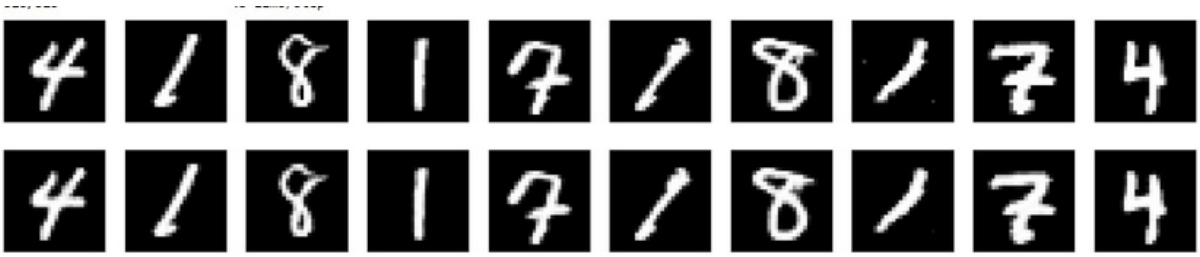
```
import numpy as np
import matplotlib.pyplot as plt

from keras import layers
from keras.datasets import mnist
from keras.models import Model

def preprocess(array):
"""Normalizes the supplied array and reshapes it."""
array = array.astype("float32") / 255.0
array = np.reshape(array, (len(array), 28, 28, 1))
return array
def noise(array):
    """Adds random noise to each image in the supplied array."""
```

```
    noise_factor = 0.4
    noisy_array = array + noise_factor * np.random.normal(
        loc=0.0, scale=1.0, size=array.shape
    )
    return np.clip(noisy_array, 0.0, 1.0)


def display(array1, array2):
    """Displays ten random images from each array."""
    n = 10
    indices = np.random.randint(len(array1), size=n)
    images1 = array1[indices, :]
    images2 = array2[indices, :]


    plt.figure(figsize=(20, 4))
    for i, (image1, image2) in enumerate(zip(images1, images2)):
        ax = plt.subplot(2, n, i + 1)
        plt.imshow(image1.reshape(28, 28))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)


        ax = plt.subplot(2, n, i + 1 + n)
        plt.imshow(image2.reshape(28, 28))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
    plt.show()
# Since we only need images from the dataset to encode and decode, we
```

```
# won't use the labels.
(train_data, _), (test_data, _) = mnist.load_data()

# Normalize and reshape the data
train_data = preprocess(train_data)
test_data = preprocess(test_data)

# Create a copy of the data with added noise
noisy_train_data = noise(train_data)
noisy_test_data = noise(test_data)

# Display the train data and a version of it with added noise
display(train_data, noisy_train_data) input =
layers.Input(shape=(28, 28, 1))

# Encoder
x = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(input) x
= layers.MaxPooling2D((2, 2), padding="same")(x)
x = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(x)
x = layers.MaxPooling2D((2, 2), padding="same")(x)

#  Decoder
x = layers.Conv2DTranspose(32, (3, 3), strides=2, activation="relu", padding="same")(x) x
= layers.Conv2DTranspose(32, (3, 3), strides=2, activation="relu", padding="same")(x) x =
layers.Conv2D(1, (3, 3), activation="sigmoid", padding="same")(x)

# Autoencoder
autoencoder = Model(input, x)
```

```
autoencoder.compile(optimizer="adam", loss="binary_crossentropy")
autoencoder.summary()
autoencoder.fit(
x=train_data,
y=train_data,
epochs=10,
batch_size=128,
shuffle=True,
validation_data=(test_data, test_data),
)
predictions = autoencoder.predict(test_data)
display(test_data, predictions)
autoencoder.fit(
    x=noisy_train_data,
    y=train_data,
    epochs=10,
    batch_size=128,
    shuffle=True, validation_data=(noisy_test_data,
    test_data),
)
predictions = autoencoder.predict(noisy_test_data)
display(noisy_test_data, predictions)
```

Output:



```
[9]: predictions = autoencoder.predict(noisy_test_data)
     display(noisy_test_data, predictions)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 8s 24ms/step
```

Result:

Autocoder has been successfully built using tensorflow/keras.