Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department

Deep Learning/Odd Sem 2023-23/Experiment 3a

Name: Subrato Tapaswi	Class/Roll No.: D16AD/60	Grade:

Title of Experiment: Autoencoder for image compression.

Objective of Experiment: To develop and implement an autoencoder-based image compression system that efficiently reduces input image sizes while preserving essential visual information, exploring the potential of autoencoders in image compression.

Outcome of Experiment: Implement an autoencoder model trained on a dataset of images, which has learned to encode and decode images efficiently.

Problem Statement: The problem is to develop an autoencoder based image compression system that addresses this issue by compressing images into a lower-dimensional representation while minimizing the loss of critical visual details.

Description / Theory:
Autoencoders, a form of artificial neural network employed in machine learning and deep learning, have multiple applications, including image compression. The fundamental concept behind utilizing autoencoders for image compression is to acquire a condensed representation of the input image while retaining its crucial characteristics.

1. Architecture: An autoencoder consists of two main parts.
   Encoder: The encoder takes an input image and maps it to a lower-dimensional representation called the “latent space” or “encoding”. This encoding typically has fewer dimensions than the original image, which leads to compression.
   Decoder: The decoder takes the encoded representation and attempts to reconstruct the original image from it.

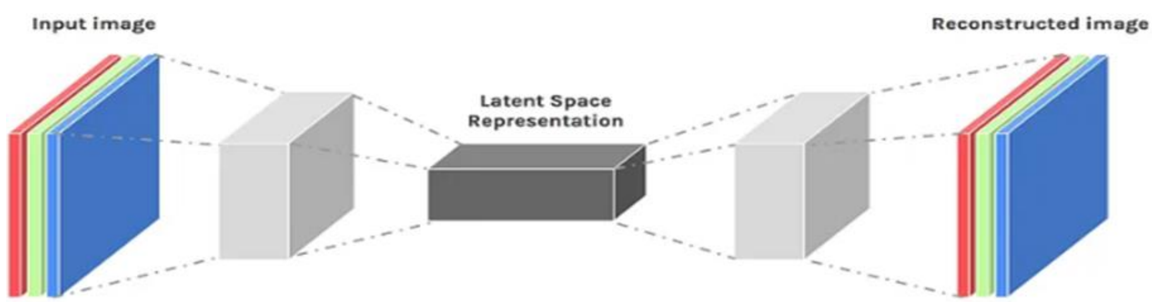
2. Training: Autoencoders are trained using unsupervised learning. The training objective is to minimize the reconstruction error, which measures how well the decoder can reconstruct the input image from its encoding.
   Common loss functions for image compression tasks include mean squared error (MSE) or binary cross-entropy, depending on whether the images are continuous or binary (e.g., grayscale or black-and-white).

3. Compression: The key to image compression with autoencoders is the reduction in the dimensionality of the encoding compared to the original image.
   By encoding the image in a lower-dimensional space, it's possible to represent the image using fewer bits, which results in compression.

Applications -

Autoencoders for image compression are used in various fields, including medical imaging, video streaming, and image transmission in low-bandwidth environments. They are also used as a preprocessing step for other computer vision tasks, where reducing the dimensionality of images can improve efficiency.

Algorithm/ Pseudo Code:



Autoencoder architecture

Program:

```
In [1]: import numpy as np
        import tensorflow as tf
        import matplotlib.pyplot as plt
        from tensorflow.keras import layers
        from tensorflow.keras.datasets import mnist
        from tensorflow.keras.models import Model
        import keras
        from keras import layers
```

```
In [2]: (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.reshape((len(x_train), np.prod(x_train.shape[1:])))
        x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
```

```
In [3]: encoding_dim = 32
        input_img = keras.Input(shape=(784,))
        encoded = layers.Dense(encoding_dim, activation='relu')(input_img)
        decoded = layers.Dense(784, activation='sigmoid')(encoded)
        autoencoder = keras.Model(input_img, decoded)
```

Deep Learning/Odd Sem 2023-23/Experiment 3a

jupyter D16AD_60_DL_Exp-3a_Code Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

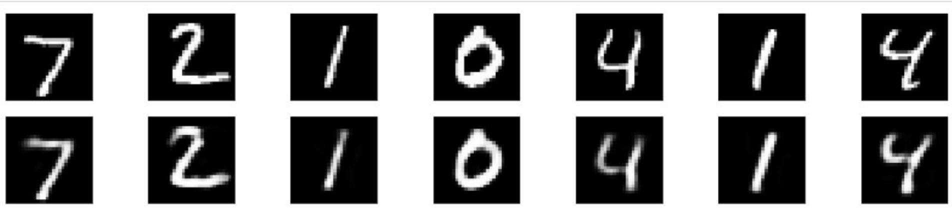
In [5]:
```
autoencoder.fit(x_train, x_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))
```

```
Epoch 1/50
235/235 [==============================] - 2s 5ms/step - loss: 0.2745 - val_loss: 0.1851
Epoch 2/50
235/235 [==============================] - 1s 4ms/step - loss: 0.1674 - val_loss: 0.1511
Epoch 3/50
235/235 [==============================] - 1s 4ms/step - loss: 0.1422 - val_loss: 0.1324
Epoch 4/50
235/235 [==============================] - 1s 5ms/step - loss: 0.1277 - val_loss: 0.1212
Epoch 5/50
235/235 [==============================] - 1s 4ms/step - loss: 0.1185 - val_loss: 0.1133
Epoch 6/50
235/235 [==============================] - 1s 4ms/step - loss: 0.1117 - val_loss: 0.1075
Epoch 7/50
235/235 [==============================] - 1s 4ms/step - loss: 0.1065 - val_loss: 0.1029
```

Output:

In [7]:
```
n = 7
plt.figure(figsize=(20, 4))
for i in range(n):
    #original
    ax = plt.subplot(2, 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)

    #reconstruction
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```



Results and Discussions: The autoencoder successfully learns a compressed representation of the input images in its encoded layer. The encoded images capture the essential features of the original images, albeit with a lower dimensionality, The decoded images show that the model can reconstruct the input data with reasonable accuracy, although there may be some loss of fine details. This type of autoencoder can be used for various applications, including image denoising, dimensionality reduction, and feature extraction.