Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	[@KLWKS_bot] THANOS

Experiment 13: Denoise Autoencoder for Feature Extraction on MNIST Data.

Aim/Objective: To implement an autoencoder on the MNIST dataset for feature extraction, aiming to learn a compressed representation of the input data.

Description: Autoencoders can be used for feature extraction by training the model to learn a compact representation of the input data. In this experiment, we'll utilize the MNIST dataset to build an autoencoder that extracts meaningful features from the handwritten digit images.

Pre-Requisites: Basic knowledge of Understanding of autoencoders and their architecture, PyTorch Basics, Python Programming, Familiarity with the library, Basic knowledge of the MNIST dataset.

Pre-Lab:

1. What is the primary purpose of using an autoencoder for feature extraction?

Extracts essential features by compressing and reconstructing data.

2. How does the size of the bottleneck layer (latent space) affect the quality of features extracted by an autoencoder?

Smaller size forces key features; larger size retains more details but may include noise.

3. What is the role of activation functions in the context of feature extraction using autoencoders?

Adds non-linearity, helping capture complex patterns.

4. How does an autoencoder differ from traditional dimensionality reduction techniques like PCA when it comes to feature extraction?

Autoencoders handle non-linear features, while PCA is limited to linear relationships.

In-Lab:
Program 1: Build a Denoise Autoencoders for Feature Extraction for the MNIST data set. Display the feature set.
Procedure/Program:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Conv2DTranspose, Flatten, Reshape
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import mnist

(x_train, _), (x_test, _) = mnist.load_data()

x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train[..., np.newaxis]
x_test = x_test[..., np.newaxis]

noise_factor = 0.5
x_train_noisy = np.clip(x_train + noise_factor * np.random.randn(*x_train.shape), 0., 1.)
x_test_noisy = np.clip(x_test + noise_factor * np.random.randn(*x_test.shape), 0., 1.)

inp = Input((28, 28, 1))

x = Conv2D(32, 3, activation='relu', padding='same')(inp)
x = Conv2D(64, 3, activation='relu', padding='same', strides=2)(x)
x = Conv2D(128, 3, activation='relu', padding='same', strides=2)(x)
encoded = Flatten()(x)

x = Reshape((7, 7, 128))(encoded)
x = Conv2DTranspose(64, 3, activation='relu', padding='same', strides=2)(x)
x = Conv2DTranspose(32, 3, activation='relu', padding='same', strides=2)(x)
decoded = Conv2D(1, 3, activation='sigmoid', padding='same')(x)

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

```
autoencoder.fit(x_train_noisy, x_train, epochs=10, batch_size=128,
validation_data=(x_test_noisy, x_test))


encoder = Model(inp, encoded)
features = encoder.predict(x_test_noisy)

plt.figure(figsize=(10, 5))

for i in range(10):
   plt.subplot(2, 5, i + 1)
   plt.imshow(features[i].reshape(16, -1), cmap='viridis')
   plt.axis('off')

plt.show()
```

- **Data and Results:**

Data

The MNIST dataset contains handwritten digit images for deep learning.

Result

The autoencoder successfully denoised images and extracted meaningful features.

- **Analysis and Inferences:**

Analysis

Feature representations show compressed and abstract patterns from noisy images.

Inferences

Denoising autoencoders effectively enhance image quality and feature learning.

Sample VIVA-VOCE Questions (In-Lab):

1. How can the features extracted by an autoencoder be used for downstream tasks?

Features from the bottleneck layer can be used for classification, clustering, anomaly detection, etc. Example: Feeding extracted features into an SVM for classification.

2. Discuss the trade-off between the size of the bottleneck layer and the quality of features extracted by an autoencoder.

A smaller bottleneck improves generalization but may lose details; a larger one retains more information but risks redundancy and overfitting.

3. Can an autoencoder be used for feature extraction in non-image data? Provide an example.

Yes, e.g., in NLP, autoencoders learn text embeddings for sentiment analysis. Also used in anomaly detection for network security.

4. Explain the concept of unsupervised feature learning and its relevance to autoencoders.

Autoencoders learn meaningful representations without labeled data, useful for clustering and semi-supervised learning.

5. How does the choice of loss function impact the quality of features learned by an autoencoder?

MSE ensures smooth reconstructions but may blur details, while other losses (e.g., binary cross-entropy, adversarial loss) affect feature quality differently.

Post-Lab:

Program 2: Hyperparameter Tuning for Feature Extraction, Explore the impact of changing hyperparameters, such as the size of the bottleneck layer, number of hidden layers, and activation functions, on the quality of features extracted by the autoencoder.

Procedure/Program:

1. **Bottleneck Size:** Small → better generalization; Large → richer features but risk overfitting.
2. **Hidden Layers:** Fewer → simpler, less expressive; More → deeper representation, higher risk of overfitting.
3. **Activation:** ReLU (fast), Leaky ReLU (better for dying neurons), Sigmoid/Tanh (vanishing gradients risk).
4. **Batch Size:** Small → better generalization; Large → stable but may overfit.
5. **Learning Rate:** High → faster but unstable; Low → slow but stable.
6. **Regularization:** L1 (sparse), L2 (prevents overfitting), Dropout (better generalization).

Tuning Strategy: Grid/random search, reconstruction error, clustering, and classification accuracy.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	[@KLWKS_bot] THANOS

- **Data and Results:**

Data

Collected input samples, preprocessed, normalized, and fed into autoencoder model.

Results

Feature extraction improved with optimized hyperparameters, reducing reconstruction error significantly.

- **Analysis and Inferences:**

Analysis

Smaller bottleneck compressed features efficiently, deeper networks captured complex representations.

Inferences

Balanced hyperparameters enhance feature quality, avoiding overfitting or underfitting issues.

Evaluator Remark (if Any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.