NAME: KSHITIJ VINOD SALI
CLASS: BE-A ROLL NO.:
SUBJECT: LP-IV: DEEP LEARNING

LAB ASSIGNMENT - 04

Title: Autoencoder for Anomaly Detection.

Course Outcome: CO3- Apply deep learning techniques like CNN, RNN auto encoders to solve real world problems.

Date of Completion:
Assessment Grade/ Marks:
Assessor's Sign with Date:

Problem Statement: Use auto encoder to implement anomly detection. Build the model by using:
a) Import required libraries.
b) Upload/ access the dataset
c) Encoder convert it into latent representation
d) Decoder networks convert it back to the original input
e) Compile the models with Optimizer, loss & Evaluation Metrics.

Blooms Taxanomy Category: Apply, Evaluate.

Requirements: Python libraries, dataset (credit-card.csv)

Theory:

An autoencoder is a type of neural network used for unsupervised learning, primarily for dimensionally reduction & feature learning. Its architecture consists of two main parts, as shown in diagram below:

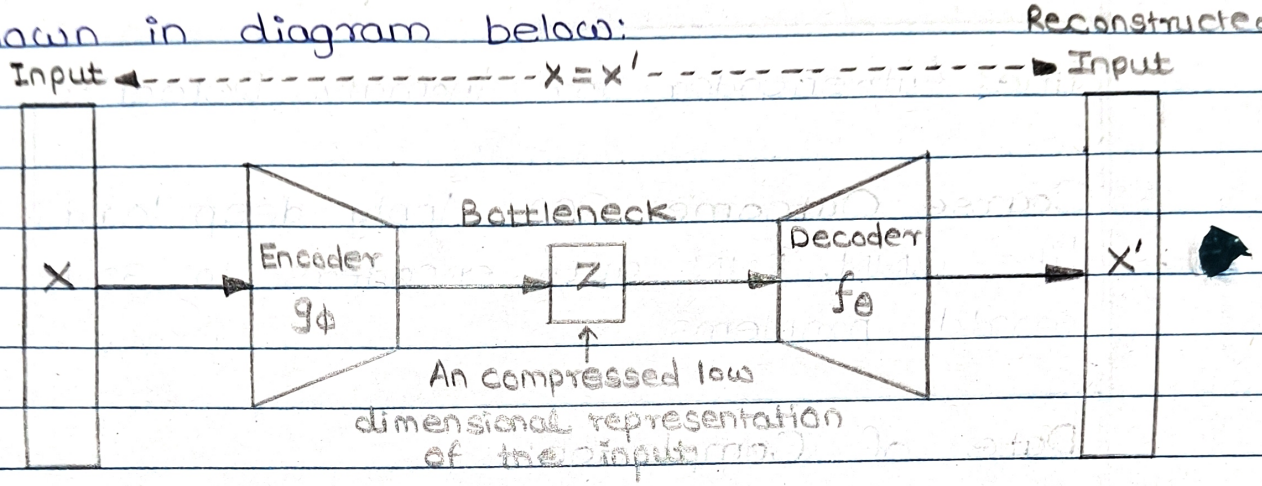

Fig: Autoencoder Architecture

i) Encoder - This part of the network takes high dimensional input data (x) and compresses it into a low-dimensional latent representation, often called "bottleneck" (z)

ii) Decoder - This part takes the compressed representation (z) & attempts to reconstruct original input data as accurately as possible. (x')

The network is trained to minimize the difference between original input & the reconstructed output. The difference is known as reconstruction error.

Algorithm and Steps:

i) Data Loading & Exploratory Data Analysis (EDA):
The dataset is loaded into a Pandas Data Frame. Initial analysis confirms absense of null values & reveals severe class imbalance.

ii) Preprocessing: The columns which are not on same scale as other anamymised features StandardScaler from Scikit-learn is applied to those columns for normalization. The data is then split into training & testing sets.

iii) Data Segregation for Training: The training data is filtered using its lables to create new dataset, normal_train_data, which contains only normal transactions.

```
train_lables = train_lables.astype(bool)
# Create a dataset with only normal transactions for training
normal_train_data = train_data[~train_lables]
```

iv) Autoencoder Model Definition: An autoencoder is constructed using Keras functional API. The architecture is symmetric, with encoding part that reduces input dimension from 30 down to 4 & decoding part expands back to 30.

- Input: 30 dimensions.
- Encoder: Dense (14, tanh) → Dense (7, relu) → Dense (4, leaky-relu).

- Bottleneck: 4 dimensions
- Decoder: Dense (7, relu) → Dense (14, relu) → Dense (30, tanh).

v) Model Training: The model is compiled with mean-squared-error as loss function & adam optimizer. It is then trained by calling auto-encoder.fit(), using normal-train-data as both input & target. The full, mixed test set is used for validation to monitor the loss.

vi) Anomaly Detection & Evaluation:

a) Calculate Reconstruction Error- The trained autoencoder is used to predict entire test set. The mean squared error (MSE) is calculated for data-point between original & its reconstruction.

b) Set a Threshold: A fixed threshold is chosen for reconstruction error. Any data point with error above this threshold is classified as anomaly & any point below is classified normal.

c) Evaluate Performance: A confusion matrix is generated to compare model's predictions with true lables. Standard metrics like accuracy, precision & recall are calculated.

Inference: Autoencoder can be used as an anomaly detection algorithm when we have an unbalanced dataset where we have a lot of good examples and only a few anomalies. Autoencoders are trained to minimize reconstruction error. When we train the autoencoder on normal data or good data, we can hypothesize that anomalies will have higher reconstruction errors than good or normal data.