Ex. No : 7 Date: 5/3/24.	IMPLEMENT ONLINE FRAUD DETECTION OF SHARE MARKET DATA USING ANY ONE OF THE DATA ANALYTICS TOOLS.

AIM

To implement a data analytics solution for real-time fraud detection in share market transactions, showcasing the application of data analytics in financial security.

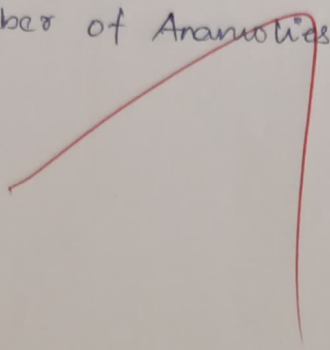
ALGORITHM

Step 1: Acquire a dataset of share market transaction data.
Step 2: Preprocess the data by cleaning, transforming, and aggregating relevant features.
Step 3: Choose a suitable data analytics tool, such as Pandas, R, or SQL.
Step 4: Implement data exploration and visualization to identify potential patterns of fraudulent transactions.
Step 5: Apply machine learning algorithms, like isolation forests or clustering, to detect anomalies.
Step 6: Evaluate the effectiveness of the fraud detection method using metrics like precision, recall, and F1-score.

PROGRAM

```
import pandas as pd
from sklearn.ensemble import IsolationForest
from sklearn.model_selection import train_test_split
data = pd.read_csv("C://Users//STUDENT//Downloads//archive(2)//
                                                AM2N.csv")
features = ["Adj close", "volume"]
train_data, test_data = train_test_split(data[features],
        test_size = 0.2, random_state = 42)
model = IsolationForest(contamination = 0.05)
model.fit(train_data)
```

```
predictions = model.predict(test_data)
num_anomalies = len(predictions[predictions == -1])
print('Number of Anamolies Detected:', num_anomalies)
```



OUTPUT

Number of Anamolies Detected = 16.

INFERENCE:

From the above program, I learnt the implementation of online fraud detection of share market data using any one of the data analytics tools.

VIVA QUESTIONS:

1) **What are data analytics tools, and why are they important in modern data-driven organizations?**

Data analytics tools are software applications used to analyze, manipulate and visualize data, crucial for during insights.

2) **Can you classify data analytics tools based on their functionality and use cases?**

It can be catgorized such as Business Intelligence tools for reporting and visualization, statistical tools for data analysis.

3) **What are some common features and capabilities of data analytics tools?**

* Visualization
* Statistical analysis
* Data mining
* Predictive modelling
* Interactive dashboards
* Decision support

4) **How do data analytics tools handle large-scale datasets and big data processing?**

It employ distributed computing frameworks like hadoop and spark to handle large scale-datasets leveraging parallel processing.

5) **Can you describe any popular open-source data analytics tools and their key features?**

Apache hadoop for distributed data storage and processing. Apache spark for in munory data processing.

RESULT

Thus the program for Develop Object Detection and Classification for Traffic Analysis using CNN was executed successfully and verified.

Ex. No : 8 Date: 12/3/24	IMPLEMENTATION OF RNN

AIM:

To write a program to perform language modeling using RNN.

ALGORITHM:

Step 1: Define a text corpus that you want to use for training. This text corpus can be any textual data you want to model.

Step 2: Create a vocabulary by extracting all unique characters from the text.

Step 3: Create a mapping (dictionaries) from characters to integers (char_to_index) and vice versa (index_to_char) to encode and decode characters during training and text generation.

Step 4: Create sequences of characters as training samples, where each sequence consists of a fixed number of characters (max_sequence_length) from the text. The target for each sequence is the character immediately following the sequence.

Step 5: Encode these sequences and targets as one-hot vectors to prepare the training data (x and y).

Step 6: Build an RNN model using Keras. In this example, an LSTM layer is used with 128 units, but you can modify the architecture as needed.

Step 7: Add a Dense layer with a softmax activation function to predict the next character.

PROGRAM:

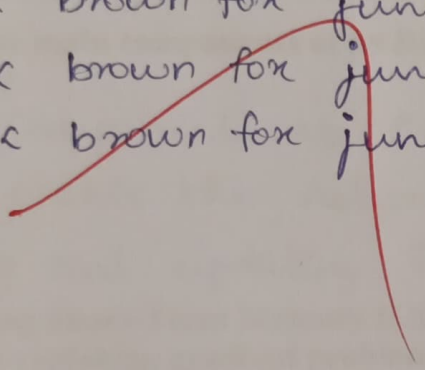
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
text = "The quick brown fox jumped over the lazy dog".
Vocab = sorted(set(text))
char_to_index = {char: index for index, char in enumerate(vocab)}
index_to_char = { index: char for index, char in enumerate(vocab)}
```

```
max_sequence_length = 40
step = 1
sequences = []
next_chars = []
for i in range(0, len(text) - max_sequence_length, step):
    input_sequence = text[i:i+max_sequence_length]
    target_char = text[i+max_sequence_length]
    sequences.append([char_to_index[char] for char in input_sequence])
    next_char.append(char_to_index[target_char])
x = np.zeros((len(sequences), max_sequence_length, len(vocab)), dtype=bool)
y = np.zeros((len(sequences), len(vocab)), dtype=bool)
for i, sequence in enumerate(sequences):
    for t, char_index in enumerate(sequence):
        x[i, t, char_index] = 1
    y[i, next_char[i]] = 1
model = sequential()
model.add(LSTM(128, input_shape=(max_sequence_length, len(vocab))))
model.add(Dense(len(vocab), activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
model.fit(x, y, batch_size=128, epochs=5)
def generate_text(seed_text, num_char_to_generate=100):
    generate_text = seed_text
    for_in range(num_char_to_generate):
        input_sequence = generated_text[-max_sequence_length:]
    x_pred = np.zeros((1, max_sequence_length, len(vocab)), dtype=bool)
    for t, char in enumerate(input_sequence):
        x_pred[0, t, char_to_index[char]] = 1
    predicted_char_index = np.argmax(model.predict(x_pred, verbose=0))
```

```
    predicted_char = index_to_char[predicted_char_index]
    generated_text += predicted_char
  return generated_text
seed_text = "The quick brown fox jumped over the lazy dog".
num_char_to_generate = 10
generated_text = ""
for i in range(num_char_to_generate):
    generated_text += seed_text + ". and"
generated_text = generated_text[:-6]
print(generated_text)
```

OUTPUT

The quick brown fox jumped over the lazy dog , and
The quick brown fox jumped over the lazy dog, and
The quick brown fox jumped over the lazy dog, and
The quick brown fox jumped over the lazy dog, and
The quick brown fox jumped over the lazy dog, and
The quick brown fox jumped over the lazy dog, and
The quick brown fox jumped over the lazy dog, and
The quick brown fox jumped over the lazy dog, and
The quick brown fox jumped over the lazy dog, and
The quick brown fox jumped over the lazy dog.



INFERENCE

From the above program, I learnt implimentation of language modeling using RNN.

VIVA VOICE:

1) **What is a Recurrent Neural Network (RNN), and how does it differ from feedforward neural networks?** RNNs are neural networks designed for sequential data processing with feedback connections allowing information persistence.

2) **What are the main components of an RNN cell, and what role do they play in processing sequential data?**

Main components are input, hidden state and output. They enable the network to process sequential data by retaining and updating information over time steps.

3) **How do Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) address the vanishing gradient problem?**

LSTM and GRU address the vanishing gradient problem by incorporating gating mechanisms allowing network to selectively update and retain information.

4) **What are the different variants of RNNs, and how do they differ in terms of architecture and functionality?** Variants of RNNs include LSTMs, GRUs and simple RNNs. LSTM and GRU have additional gating mechanisms while simple RNNs lack these mechanisms affecting their ability to capture long term dependencies.

5) **How do you handle variable-length sequences in RNNs during training and inference?**

Variable length sequences in RNNs are handled by padding sequence to a fixed length or using techniques like bucketing to group similar lengths.

RESULT:

Thus the program For Implementing RNN is executed and verified successfully.

EX. NO: 9 DATE: 19/3/24	IMPLEMENT IMAGE AUGMENTATION USING DEEP RBM.

AIM

To enhance the dataset by generating augmented images using deep restricted Boltzmann machines (RBMs), emphasizing the importance of data augmentation in improving model robustness.

ALGORITHM

Step 1: Collect a dataset of images for augmentation.
Step 2: Design a deep restricted Boltzmann machine (RBM) architecture.
Step 3: Train the RBM using the input images to learn patterns and features.
Step 4: Implement image augmentation techniques using the learned RBM representations, such as noise injection or distortion.
Step 5: Apply the augmented images for training a separate deep learning model (e.g., CNN).
Step 6: Compare the performance of the model trained with and without image augmentation.

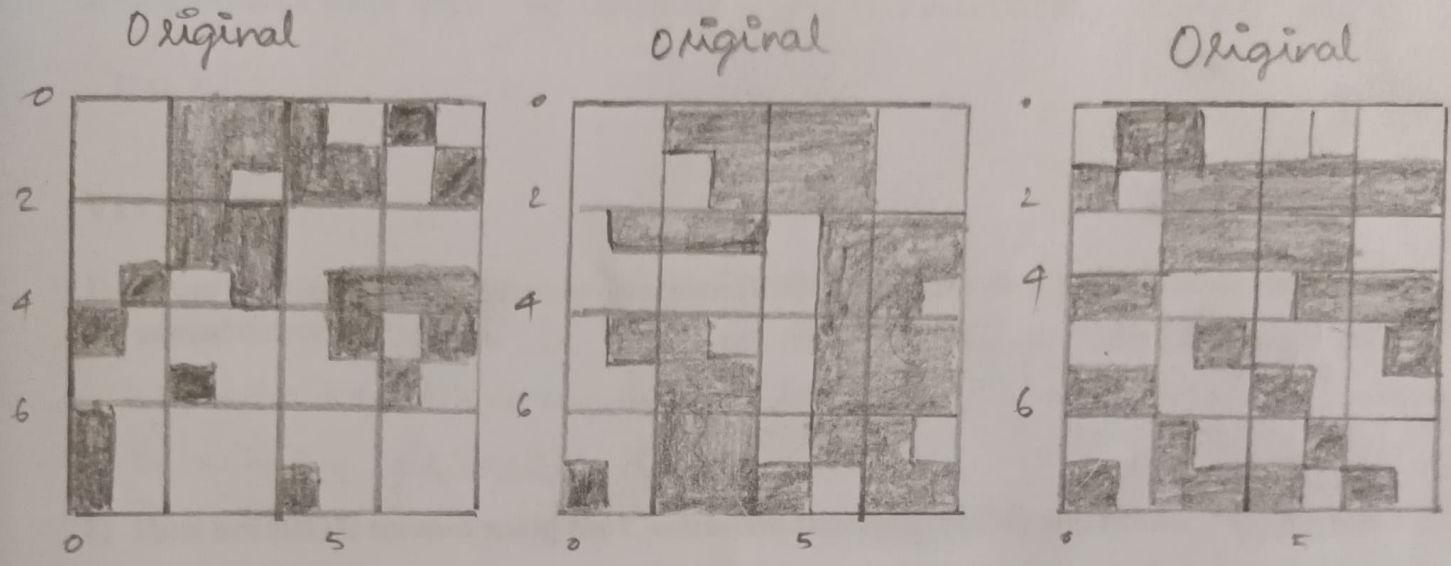
PROGRAM

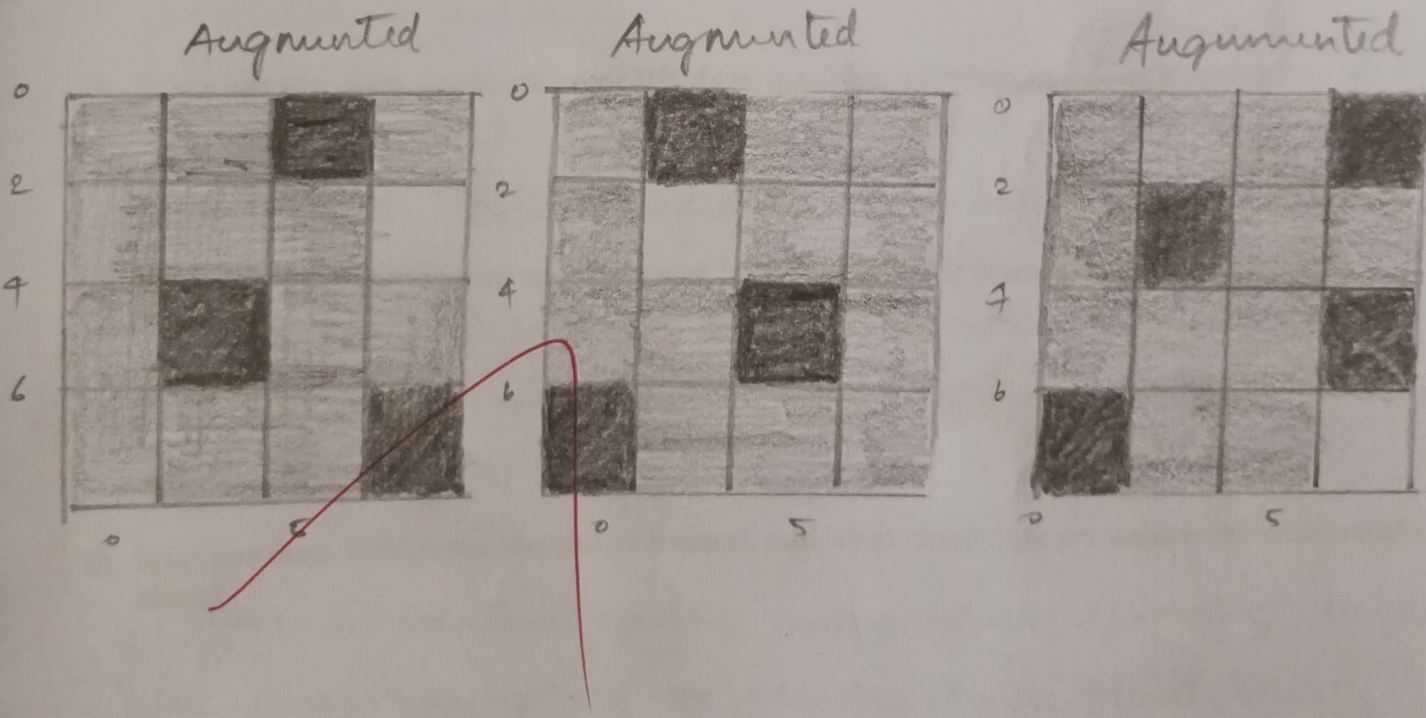
```
import numpy as np
from sklearn.neural_network import BernoulliRBM.
import matplotlib.pyplot as plt.
num_samples = 100
image_size = 8*8
original_images = np.random.randint(0,2, size=(num_samples,
                                    image_size))
rbm = BernoulliRBM(n_components = 64, n_iter = 10, batch_size = 10)
rbm2 = BernoulliRBM(n_components = 64, n_iter = 10, batch_size = 10)
rbm.fit(original_images)
```

```
hidden_features = rbm.transform(original_images)
rbm2.fit(hidden_features)
augmented_features = rbm2.transform(hidden_features)
reconstructed_hidden_features_tranformed = rbm.tranform
                    (reconstructed_hidden_features)
reconstructed_images = rbm.transform(reconstructed_hidden_
                    features_tranformed)

plt.figure[figsize=(10,4))
for i in range(5):
    plt.subplot(2,5,i+1)
    plt.imshow(original_images[i].reshape(8,8),cmap='gray')
    plt.title("original")
    plt.subplot(2,5,i+6)
    plt.imshow(reconstructed_images[i].reshape(8,8);cmap='gray')
    plt.title("Augmented")
plt.tight_layout()
plt.show()
```

OUTPUT


Original
Original
Original


Augmented
Augmented
Augmented

INFERENCE

From the above program, I learnt the implementation of image agmentation using deep RBM.

VIVA VOICE:

1) **What is a Restricted Boltzmann Machine (RBM), and how does it differ from other types of neural networks?** RBM is a type of generative stochastic neural network used for unsupervised learning, unlike other neural networks which are primarily discriminative.

2) **How are RBMs trained using the Contrastive Divergence (CD) algorithm?**
RBMs are trained using CD algorithm by updating weights to minimize the difference between the model predicted probabilities.

3) **What are the main applications of RBMs in machine learning and deep learning?**
RBMs find applications in dimensionality reduction collaborative filtering - feature learning and generative.

4) **What is a deep Boltzmann machine, and how does it differ from a single-layer RBM?**
A deep boltzman machine consists of multilayer of RBMs allowing for hirrarchical feature learning unlike single layer RBM.

5) **How are deep Boltzmann machines trained, and what challenges are associated with training them?**
It is trained using unsupervised learning techniques like Gibbs sampling or Markov chain Monte Carlo.

V. R. 2/3/24

RESULT

Thus the program to Implement Image Augmentation using Deep RBM was executed and verified successfully.

Ex.No:10 Date: 26/3/24.	IMPLEMENT SENTIMENT ANALYSIS USING LSTM.

AIM

To develop a sentiment analysis model using long short-term memory (LSTM) networks, demonstrating the use of recurrent neural networks in understanding and classifying emotions in text data.

ALGORITHM

Step 1: Gather a dataset of text samples labeled with sentiment labels.
Step 2: Preprocess the text data by tokenizing, padding, and converting to word embeddings.
Step 3: Design a long short-term memory (LSTM) neural network architecture for sentiment analysis.
Step 4: Choose a suitable loss function, like binary cross-entropy, for sentiment prediction.
Step 5: Train the LSTM on the preprocessed text data.
Step 6: Evaluate the model's performance in terms of sentiment classification accuracy.

PROGRAM

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
                                        pad_sequences.
texts = [
    "I love this product",
    "This is terrible",
    "The movie was amazing",
    "I'm not sure how I feel about it".
]
```

```
labels = np.array([1,0,1,0])
tokenizer = Tokenizer(num_words = 1000, oov_token = "<OOV>")
tokenizer.fit_on_text(texts)
word_index = tokenizer.word_index
sequences = tokenizer.text_to_sequences(texts)
padded_sequences = pad_sequences(sequences, maxlen=10,
                                 padding='post')
model = tf.keras.Sequential([tf.keras.layers.Embedding(
    input_dim = len(word_index)+1, output_dim = 16,
    input_length =10),
    tf.keras.layers.LSTM(16), tf.keras.layers.Dense
                                 (1, activation ='sigmoid')])
model.compile(optimizer ='adam', loss ='binary_crossentropy',
              metrics = ["accuracy"])
model.fit(padded_sequences, labels, epochs = 5, verbose=1)
sample_test_text = ["This is a great product!"]
prediction = model.predict(sample_padded_sequence)
print("Predicted sentiment:", "Positive" if prediction>0.5
                                 else "Negative")
```

OUTPUT

Epoch 1/3

1/1 [==========] - 1s 1s/step - loss: 0.6957 - accuracy: 0.2500

Epoch 2/3

1/1 [==========] - 0s 4ms/step - loss: 0.6950 - accuracy: 0.2500

Epoch 3/3

1/1 [==========] - 0s 4ms/step - loss: 0.6943 - accuracy: 0.5000

1/1 [==========] - 0s 227 ms/step

Predicted sentiment : Positive.

INFERENCE

From the above program, I learnt the implementation of sentiment Analysis using LSTM.

VIVA VOICE:

1) What is sentiment analysis, and why is it important in natural language processing (NLP)?

Sentiment analysis is the process of determining the emotional tone behind a piece of text, crucial for NLP in understanding opinions, attitudes.

2) How do LSTMs handle sequential data, such as text, and why are they suitable for sentiment analysis?

LSTMs utilize memory cells to retain and selectively forget information over time, making them adept at capturing long range dependencies in sequential data like text.

3) What are the main components of an LSTM cell, and how do they contribute to capturing long-range dependencies in text data?

The main components of an LSTMs cell include input, forget and output gates, along with the cell state. They work together to regulate information flow, retaining relevant content and disregarding noise.

4) How do you handle text preprocessing and tokenization before feeding data into an LSTM network for sentiment analysis?

The main component of an LSTM cell include like removing punctuation, lowercasing and handling stopwords followed by tokenization where text is split into individual token.

5) What are some common loss functions used in training LSTM networks for sentiment analysis?

Common loss function for training LSTM include binary cross-entropy for binary classification tasks.

RESULT

Thus the program for implement sentiment analysis using LSTM was executed & verified successfully.

Ex.No:11 Date: 2/4/24.	IMPLEMENT VARIATIONAL AUTO ENCODERS.

AIM

To write a program to implement Variational Auto Encoders

ALGORITHM:

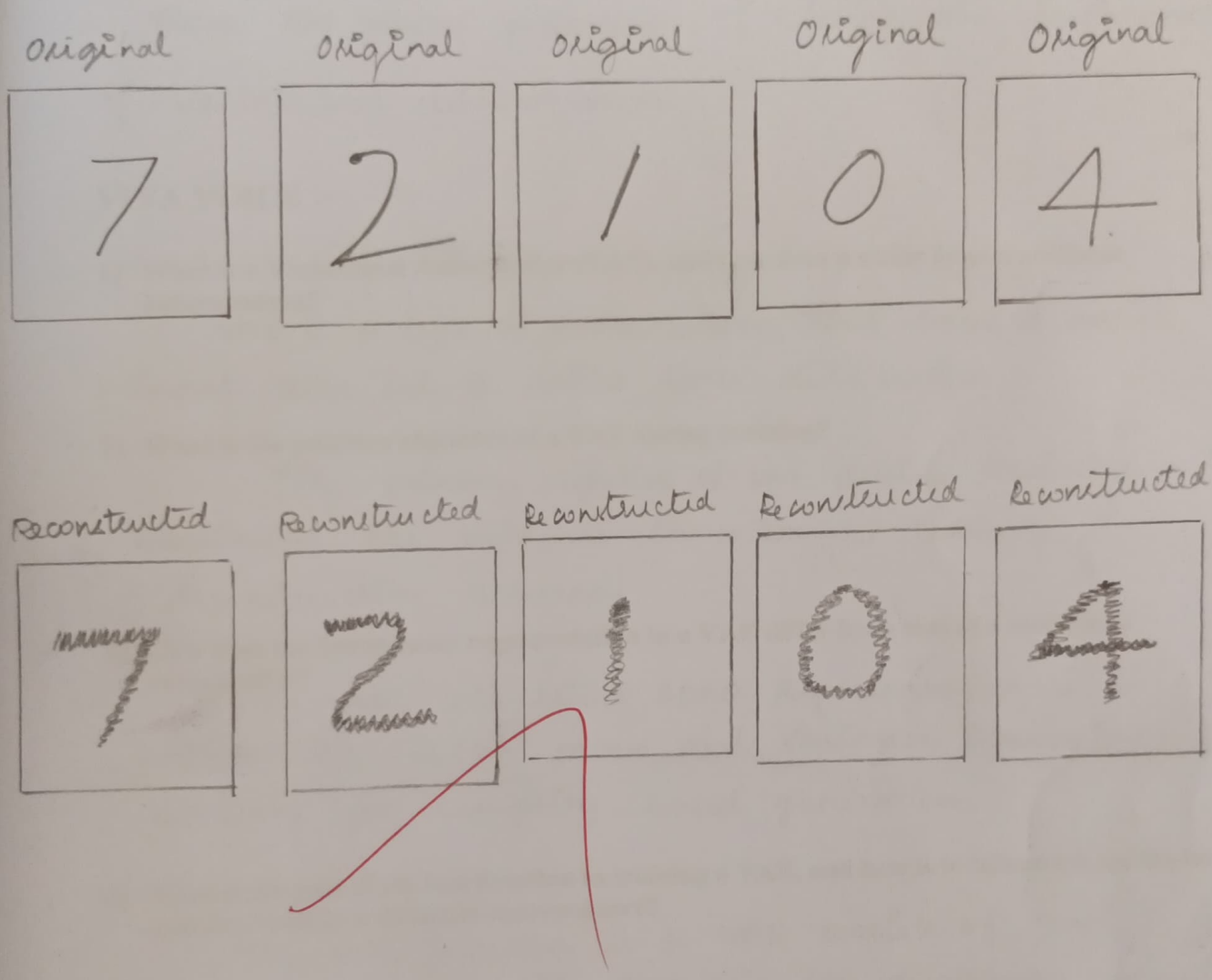
Step 1: Import specific modules and functions from TensorFlow and Keras.
Step 2: Set parameters such as **latent_dim** for the dimensionality of the latent space and **input_shape** for the shape of input images.
Step 3: Define the encoder network using convolutional layers (**Conv2D**), flatten the output, and compute the mean and log variance of the latent space (**z_mean** and **z_log_var**).
Step 4: Construct the VAE model by connecting the encoder and decoder networks.
Step 5: Evaluate the trained VAE model on unseen data (e.g., test dataset).

PROGRAM

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Flatten,
    Reshape
from tensorflow.keras.models import Model
from tensorflow.keras.dataset import mnist
(x_train,_),(x_test,_) = mnist.load_data()
x_train = x_train.astype('float32')/255-0
x_test = x_test.astype('float32')/255.0
```

```
x_train_float = x_train.reshape((len(x_train), np.prod
                (x_train.shape[1:])))
x_test_flat = x_test.reshape((len(x_test), np.prod
              (x_test.shape[1:])))
input_img = Input(shape=(784,))
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)
decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
autoencoder.fit(x_train_flat, x_train_flat, epochs=5, batch_size=256,
                shuffle=True
validation_data=(x_test_flat, x_test_flat))
encoded_imgs = autoencoder.predict(x_test_flat)
encoded_imgs = encoded_imgs.reshape((len(x_test),28,28))
n=10
plt.figure(figsize=(20,4))
for i in range(n):
    ax = plt.subplot(2,n,(+1)
    plt.imshow(x_test[i])
    plt.title("Original")
    plt.gray()
plt.show()
```

OUTPUT


Original
Original
Original
Original
Original
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed

INFERENCE

From the above program, I learnt the implementation of Variational autoencoders.

VIVA VOICE

1) **What is a Variational Autoencoder (VAE), and how does it differ from traditional autoencoders?**

VAE is a type of autoencoders that learns to encode input data into a latent space distribution.

2) **What is the primary objective of a VAE during training?**

The primary objective of VAE during training is to maximize the evidence lower bound, balancing reconstruction accuracy.

3) **How does the latent space representation in a VAE differ from that of a traditional autoencoder?**

In VAE, the latent space representation is probabilistic, with mean and variance parameters, allowing for sampling based generation.

4) **What is the role of the loss function in training a VAE, and how is it different from the loss function used in traditional autoencoders?**

The loss function in a VAE consists of two components: reconstruction loss and KL divergence, ensuring faithful reconstruction and regulation.

5) **What are some common applications of VAEs in machine learning and deep learning?**

VAEs find application in generative modelling, data compression, anamoly detection, semi-learning in fields like image.

RESULT

Thus the program for implementing Variational Auto Encoders was executed & verified successfully.

Ex.No:	1	NUMBER PLATE RECOGNITION OF TRAFFIC VIDEO ANALYSIS
Date:	16/4/24.	

AIM

To create a system that can automatically recognize and extract number plate information from traffic videos, showcasing the practical application of computer vision techniques in traffic management and law enforcement.

ALGORITHM

Step 1: Collect a dataset of traffic videos containing scenes with number plates.
Step 2: Extract frames from the videos and preprocess them.
Step 3: Design a pipeline that combines object detection for locating number plates and optical character recognition (OCR) for reading the characters.
Step 4: Implement a suitable OCR algorithm, such as Tesseract, to extract characters from number plates.
Step 5: Train and fine-tune the OCR model using labeled character data.
Step 6: Apply the pipeline to the video frames, recognize number plates, and output the results with accuracy scores.

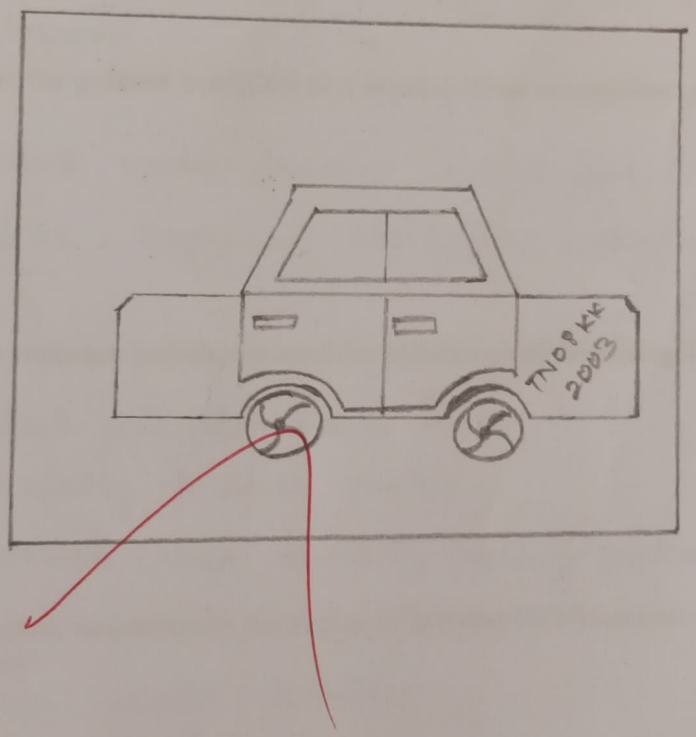
PROGRAM

```
import cv2
import numpy as np
import pytesseract
video_path = 'sample-traffic_video.mp4'.
cap = cv2.VideoCapture(video_path)
pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\
                            tesseract.exe'
```

```
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    processed_frame = cv2.GaussianBlur(gray, (5,5), 0)
    edges = cv2.Canny(processed_frame, 50, 150)
    number_plate_text = pytesseract.image_to_string(edges,
                                    config='psm1')
    cv2.imshow('Number plate Recognition', frame)
    print("Recognized Number plate:", number_plate_text.
                                        strip())
if cv2.waitKey(1) & 0xFF == ord("q"):
    break
cap.release()
cv2.destroyAllWindows()
```

OUTPUT

TN 08 KK 2003



INFERENCE

From the above program, I learnt the number plate recognition of traffic video analysis.

VIVA VOICE

1) What is number plate recognition, and why is it important in traffic video analysis?

Number plate recognition is the automated process of identifying and reading vechile license plates from video stream.

2) Can you outline the general workflow of a number plate recognition system for traffic video analysis?

Preprocess video frames, detect and localize license plate regions, segment characters, perform OCR.

3) What are some common techniques used for detecting and localizing number plates in traffic videos?

* select potential plate region
* Precisely localise plates
* Enhance image quality using contrast.

4) How do you extract and enhance the region of interest (ROI) containing the number plate from a video frame?

Haar cascade classifier
Edge detection
Color based segmentation.

5) What are some feature extraction methods used for recognizing characters within number plates?

* Histogram of Oriented Gradients * Template matching
* Convolutional neural network * Statistical moments.

RESULT

Thus the program for implementation of number plate recognition of traffic video analysis was executed successfully & verified.