PRACTICAL TECHNICAL ASSESSMENT

NAME: SOORYA S
REG NO: ADIT22AP05426

Q.1.

AIM:

Using a deep learning framework of your choice (TensorFlow, PyTorch, etc.), implement a CNN to classify images from the CIFAR-10 dataset. Ensure your network includes convolutional layers, pooling layers, and fully connected layers. Evaluate the performance of your model and discuss any improvements you could make

List of Hardware/Software used:

- Windows OS
- VS Code
- PyTorch

PROCEDURE:

Step 1: Open Visual studio code

Step 2: Create a new Python file.

Step 3: Type the code for execution

Step 4: Save and run the code

CODE

1. Import libraries

```
estion_one.py > ...
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

2. Load and Preprocess the CIFAR-10 Dataset

```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

train_images, test_images = train_images / 255.0, test_images / 255.0
```

3. Define the CNN Model

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

4. Define Loss Function and Optimizer

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

5. Train the Network

```
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

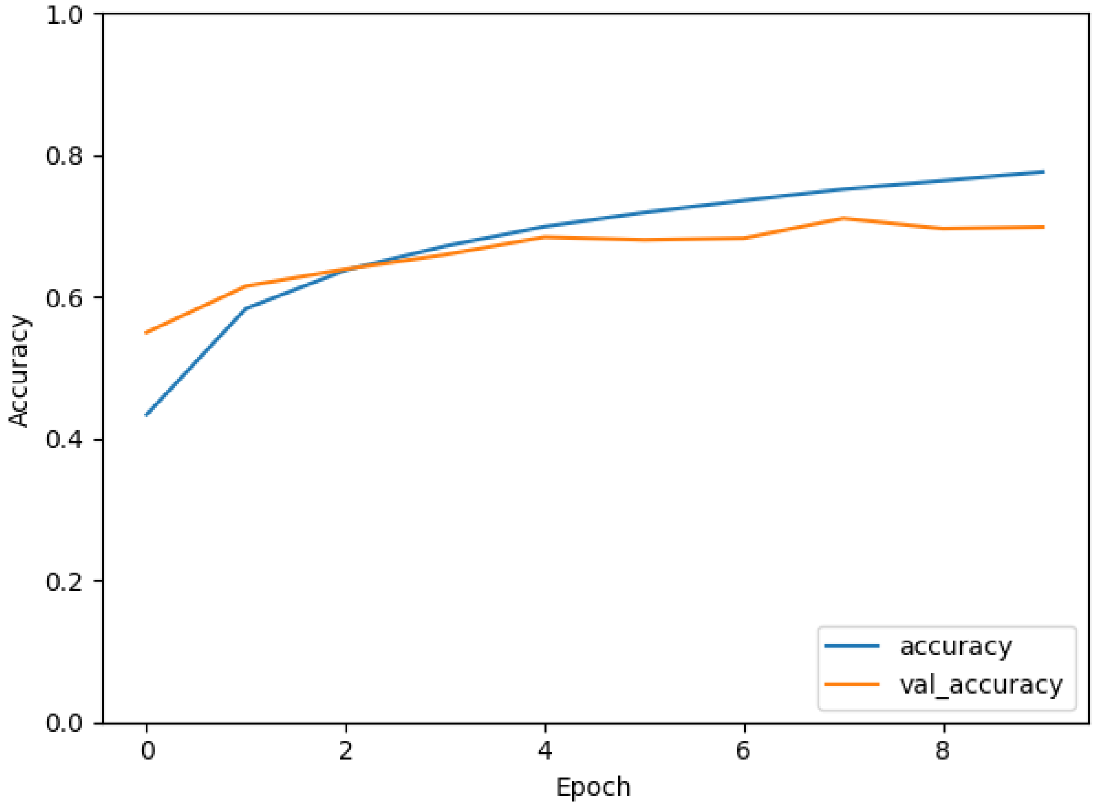
6. Evaluate the Network

```
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")
```

7. Plot training history

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```

OUTPUT:




```
Test accuracy: 0.6987000107765198
313/313 - 1s - 3ms/step - accuracy: 0.6987 - loss: 0.8997
313/313 - 1s - 3ms/step - accuracy: 0.6987 - loss: 0.8997
313/313 - 1s - 3ms/step - accuracy: 0.6987 - loss: 0.8997
Test accuracy: 0.6987000107765198
```

RESULT:

The program is successfully completed

Q 2.

AIM: Construct a feedforward neural network to predict housing prices based on the provided dataset. Include input normalization, hidden layers with appropriate activation functions, and an output layer. Train the network using backpropagation and evaluate its performance using Mean Squared Error (MSE)

List of Hardware/Software used:

- Windows OS
- VS Code

PROCEDURE:

Step 1: Open Visual studio code

Step 2: Create a new Python file.

Step 3: Type the code for execution

Step 4: Load Dataset

Step 5: Save and run the code

CODE

1. Load the Dataset

question_two.py 1 question_one.py 1 housing_prices.csv ×

housing_prices.csv

```
Bedrooms,Bathrooms,SquareFootage,Location,Age,Price
3,2,1500,Urban,10,300000
4,3,2000,Suburban,5,400000
2,1,800,Rural,20,150000
3,2,1600,Urban,12,310000
4,3,2200,Suburban,8,420000
2,1,900,Rural,25,160000
5,4,3000,Urban,3,600000
3,2,1400,Suburban,15,290000
3,2,1300,Rural,30,180000
4,3,2500,Urban,7,500000
```

```
# Load the dataset
data = pd.read_csv('housing_prices.csv')
```

2. Encode categorical data

```
# Preprocess the data
X = data.drop('Price', axis=1)
y = data['Price']
```

3. Scale/Normalize Features

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Bedrooms', 'Bathrooms', 'SquareFootage', 'Age']),
        ('cat', OneHotEncoder(), ['Location'])
    ])

X = preprocessor.fit_transform(X)
```

4. Train the model

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5. Define the Neural Network Model

```
# Build the feedforward neural network model
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1))  # Output layer

# Compile the model
model.compile(optimizer='adam', loss='mse')
```

6. Train and evaluate model

```
# Train the model
history = model.fit(X_train, y_train, epochs=100, validation_split=0.2)

# Evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

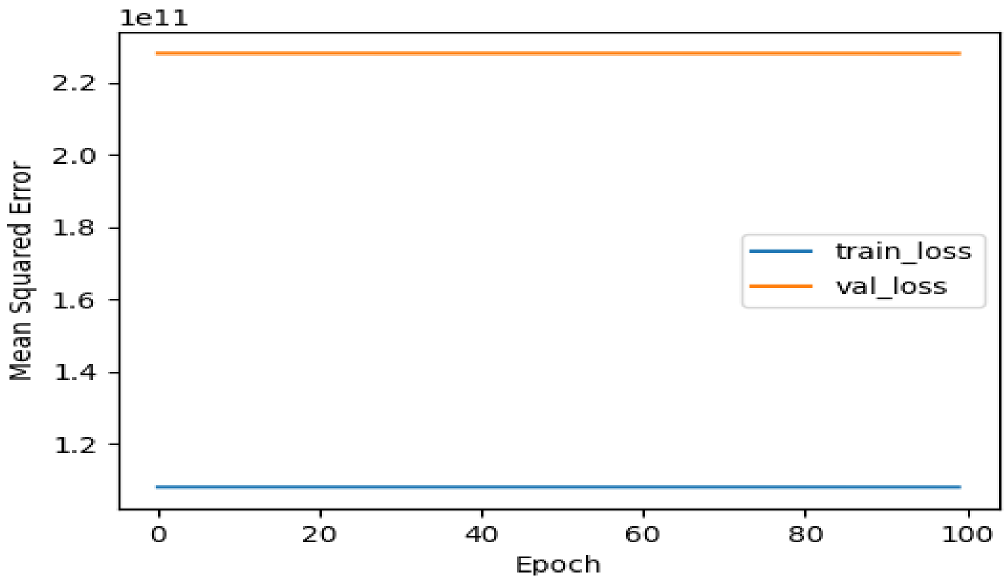
7. Plot the training history

```
# Plot the training history
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error')
plt.legend()
plt.show()
```

8. **Mean Squared Error (MSE):** The MSE of the model on the test dataset will be printed after training. This metric helps to understand the model's performance.

<u>OUTPUT:</u>



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 108081356800.0000 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
Mean Squared Error: 96181935777.23343
```

RESULT:

The program is successfully completed