Practical Technical Assessment AI

SREETHU MOHAN

Activity 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:

1. Windows OS
2. VS Code
3. PyTorch

PROCEDURE:

Step 1: Open Vscode

Step 2: Create a new Python file from "new file" option from Vscode

Step 2: save the file name with extension of ".py".

Step 4: write and save the code

Step 5: Run the code to get the output.

CODE :-

1. Importing libraries

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

2. Load and Preprocess the CIFAR-10 Dataset

```
# 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

```python
# 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'))
```

4. Define Loss Function and Optimizer

```python
# Add Dense layers on top
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

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

5. Train the Network

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

6. Evaluate the Network

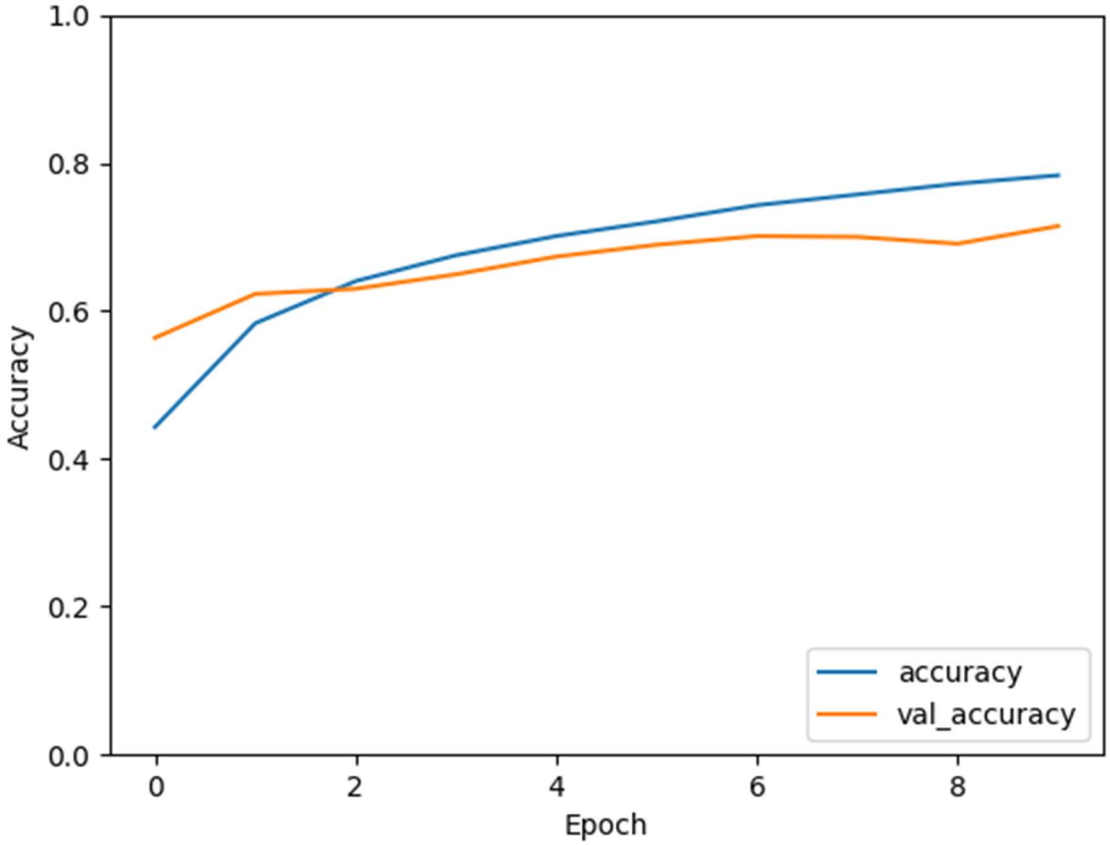
```python
# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")
```

7. Plot the model

```python
# 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: 0.7146999835968018
```



Result:

The program is successfully completed

Activity 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:

1. Windows OS
2. VS Code
3. PyTorch

PROCEDURE:

Step 1: Open Vscode

Step 2: Create a new Python file from "new file" option from Vscode

Step 2: save the file name with extension of ".py".

Step 4: write and save the code

Step 5: Run the code to get the output.

Code:

1. import Libraries

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras import layers, models
```

2. Load the Dataset

```
# Load the dataset
data = pd.read_csv('housing_price.csv')
# Preprocess the data
X = data.drop('Price', axis=1)
y = data['Price']
```

3. Encode Categorical Variables

```
# One-hot encode the 'Location' feature
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['Bedrooms', 'Bathrooms', 'SquareFootage', 'Age']),
        ('cat', OneHotEncoder(), ['Location'])
    ])
X = preprocessor.fit_transform(X)
```

4. Splitting the data

```
# 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
```

6. Train the Model

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

7. Evaluate the Model

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.

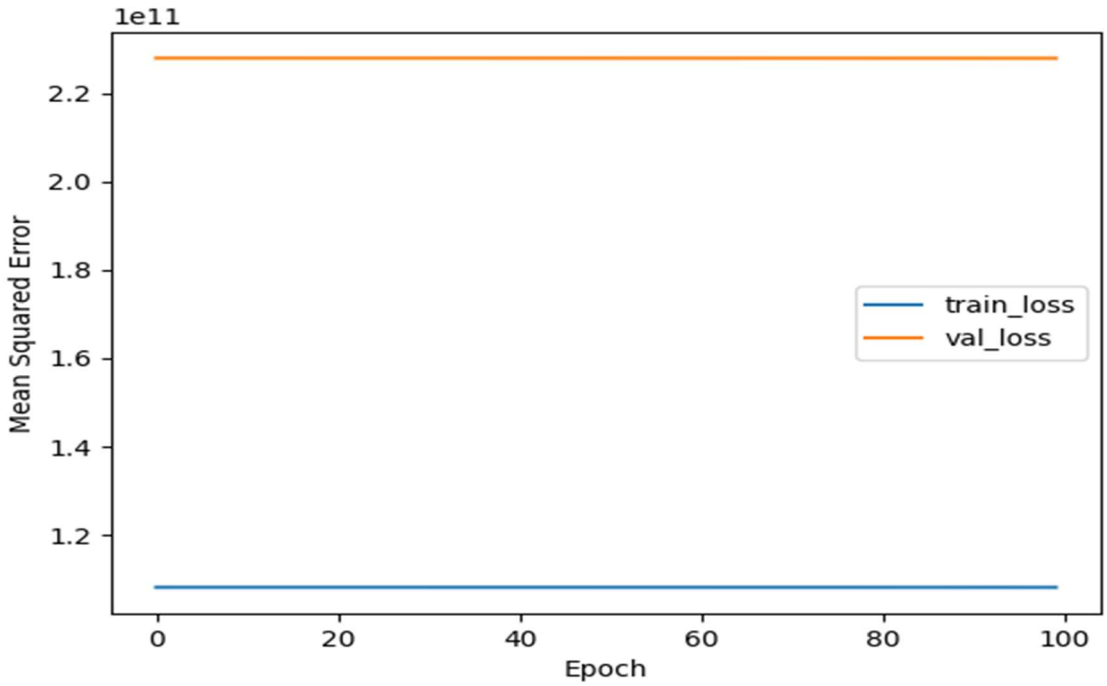
```
# Evaluate the model
y  pred = model.predict(X  test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

8. Plot the model

```
# 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()
```

OUTPUT:

```
Mean Squared Error: 96185794980.51886
```



Result:

The program is successfully completed