**Weekly Assessment**

**NAME : DEVU VIJAYAN**

**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:**

   Windows OS

   VS Code

* PyTorch

**PROCEDURE:**

Step 1: Open VS code

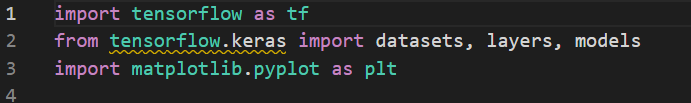
Step 2: Create a new Python file

Step 3: Rename the file and type the code to execute the program.

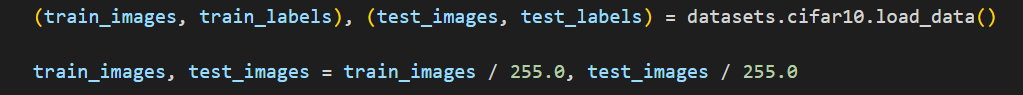
Step 4: Save and run the code

**CODE :-**

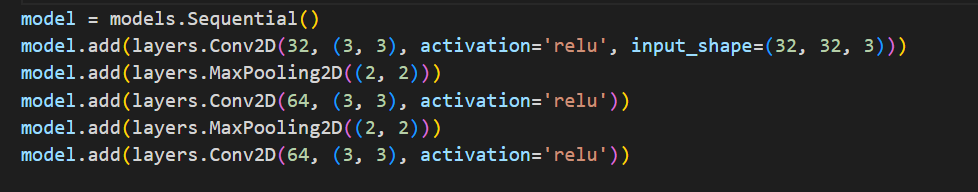
1. Import libraries



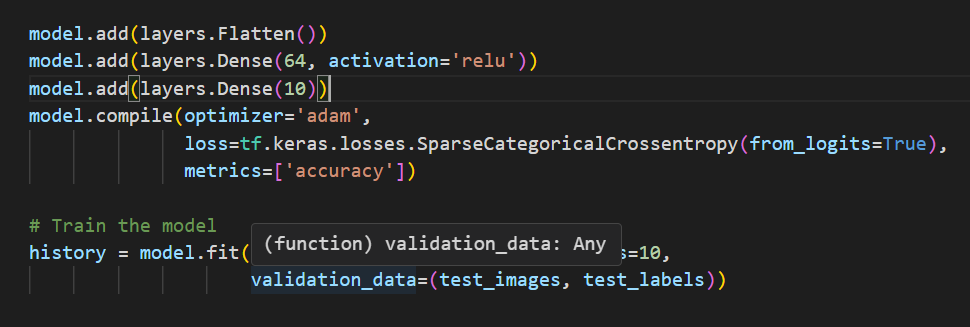
1. Load and Preprocess the CIFAR-10 Dataset



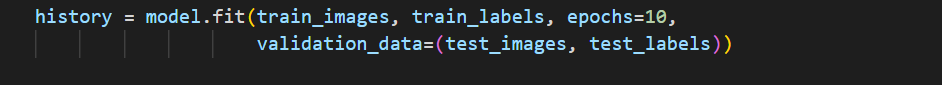
1. Define the CNN Model



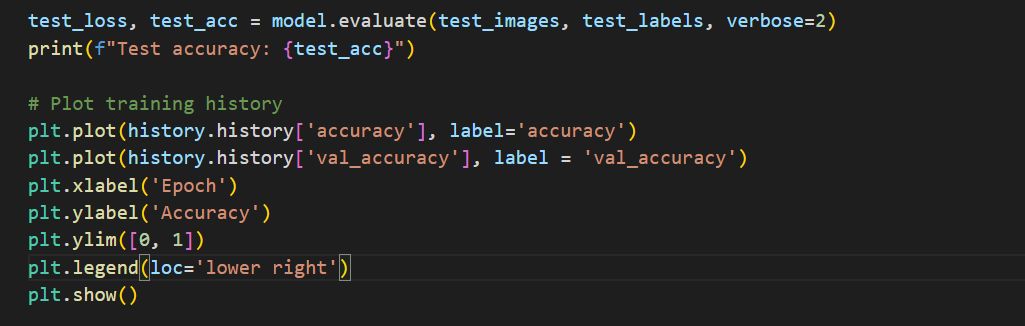
1. Define Loss Function and Optimizer



1. Train the Network



1. Evaluate the Network



CODE:

import tensorflow as tf

from tensorflow.keras import datasets, layers, models

import matplotlib.pyplot as plt

# 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

# 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))

model.compile(optimizer='adam',

              loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

              metrics=['accuracy'])

# Train the model

history = model.fit(train\_images, train\_labels, epochs=10,

                    validation\_data=(test\_images, test\_labels))

# Evaluate the model

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, verbose=2)

print(f"Test accuracy: {test\_acc}")

# Plot training history

plt.plot(history.history['accuracy'], label='accuracy')

plt.plot(history.history['val\_accuracy'], label = 'val\_accuracy')

plt.xlabel('Epoch')

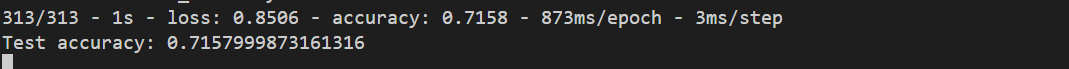
plt.ylabel('Accuracy')

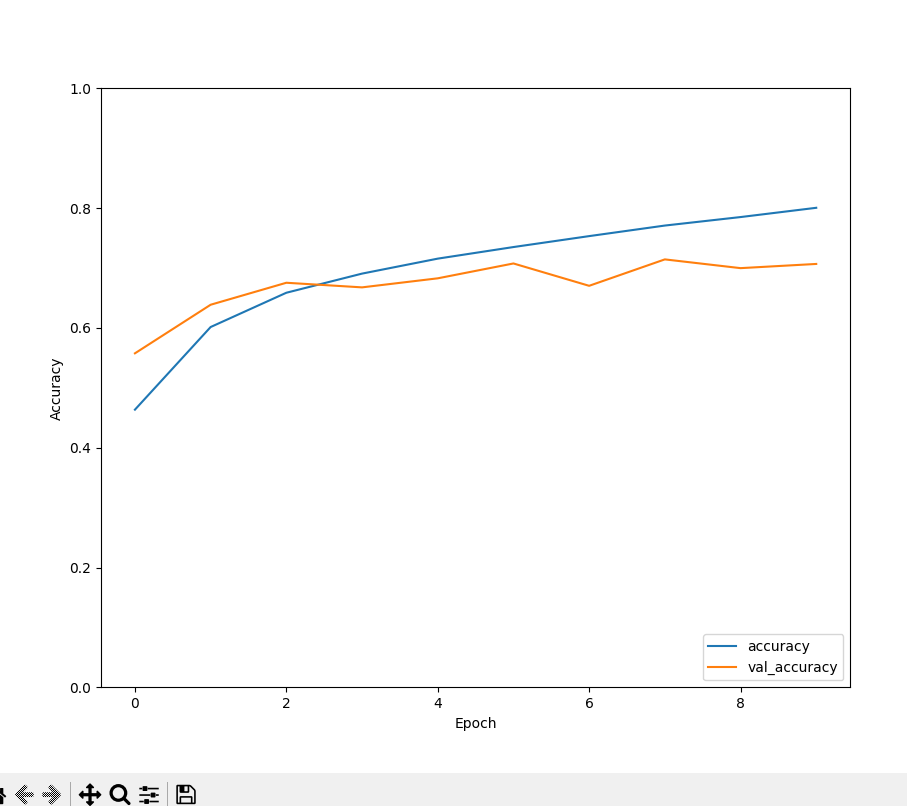
plt.ylim([0, 1])

plt.legend(loc='lower right')

plt.show()

**Output:**

****



**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:**

* + - Windows OS
    - VS code

**PROCEDURE:**

#### 1: Open Visual Studio Code

#### 2: Create a new Python file and name it housing\_prices\_prediction.py

#### 3: Install required Python libraries

#### 

#### 4: Load the Dataset

#### 

#### Use pandas to load the dataset from a CSV file.

#### 

#### 5: Encode Categorical Variables.Convert categorical variables into numerical values using one-hot encoding.

#### 

#### 6: Apply standardization to features using StandardScaler

#### 

#### 7: Define the Neural Network Model.Build the neural network using PyTorch.

#### 

#### 8: Train the Model Train the model using backpropagation.

#### 

#### 9: 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.

#### 

#### Code:

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

# Load the dataset

data = pd.read\_csv('housing\_prices.csv')

# Preprocess the data

X = data.drop('Price', axis=1)

y = data['Price']

# One-hot encode the 'Location' feature

preprocessor = ColumnTransformer(

    transformers=[

        ('num', StandardScaler(), ['Bedrooms', 'Bathrooms', 'SquareFootage', 'Age']),

        ('cat', OneHotEncoder(), ['Location'])

    ])

X = preprocessor.fit\_transform(X)

# 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)

# 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')

# 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}")

# 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')

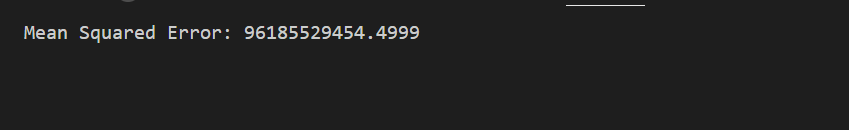
plt.xlabel('Epoch')

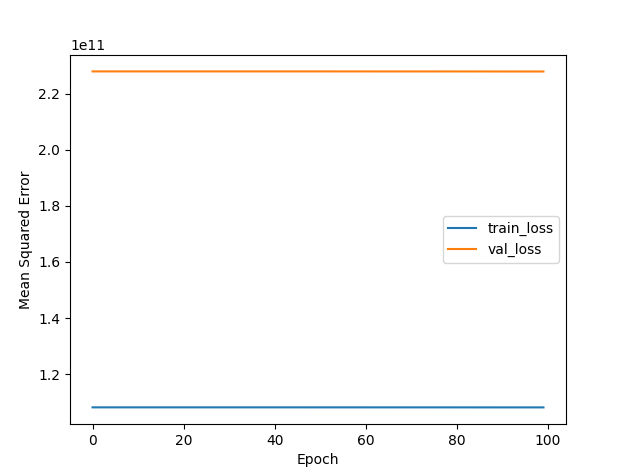
plt.ylabel('Mean Squared Error')

plt.legend()

plt.show()

**OUTPUT:**





**RESULT:**

The program is successfully completed