**ASSESMENT-3**

**PROBLEM STATEMENT-** Applying the Convolution Neural Network on computer vision problems

**SOURCE CODE:**

* Open the folder in the Spyder IDE
* The Command to install

**“pip install tensorflow matplotlib”**

**File Name: program3.py**

import tensorflow as tf

from tensorflow.keras import layers, models, datasets

import matplotlib.pyplot as plt

# Load and preprocess 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 CNN architecture

model = models.Sequential([

    layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.Flatten(),

    layers.Dense(64, activation='relu'),

    layers.Dense(10)

])

# Compile the model

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("\nTest 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()

**Explanation:**

Here's a breakdown of the code:

1. **Load and Preprocess Data:**

* It loads the CIFAR-10 dataset using TensorFlow's datasets module.
* Preprocesses the images by scaling pixel values to the range [0, 1].

1. **Define CNN Architecture:**

* It defines a CNN architecture using the Sequential API in Keras.
* The architecture consists of three convolutional layers with ReLU activation, followed by max-pooling layers and dense layers.

1. **Compile the Model:**

* It compiles the model using the Adam optimizer and sparse categorical cross-entropy loss function.

1. **Train the Model:**

* It trains the model on the training data for 10 epochs.

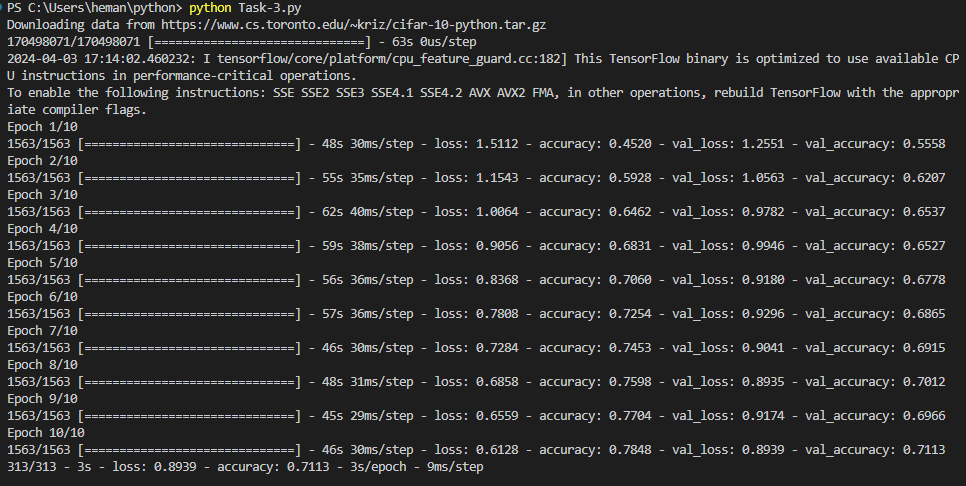
1. **Evaluate the Model:**

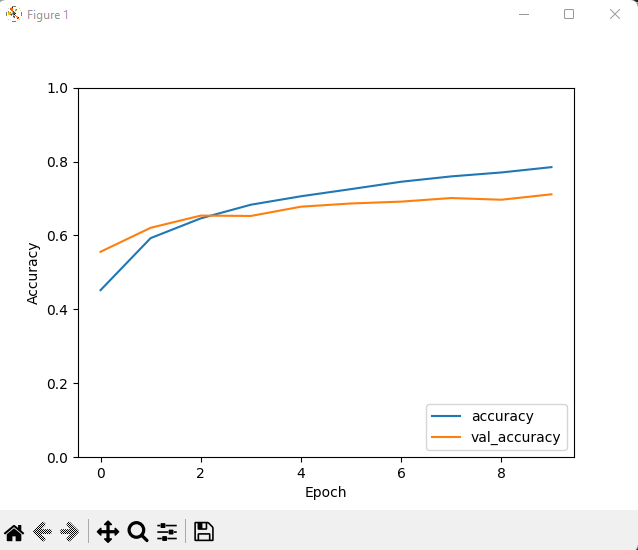
* It evaluates the trained model on the test data and prints the test accuracy.

1. **Plot Training History:**

* It plots the training and validation accuracy over epochs to visualize the training progress.

**OUTPUT:**

****

****