**Aim:** Write a program to build ConvNet in TensorFlow for a classification problem

**Description:**

A Convolutional Neural Network (ConvNet or CNN) is a deep learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and differentiate one from the other. This program demonstrates how to build and train a ConvNet using TensorFlow and Keras on the CIFAR-10 dataset.

**Objective:** To design and implement a CNN that classifies images into one of the ten classes in the CIFAR-10 dataset using TensorFlow.

**Steps Overview:**

 Import required libraries

 Load and preprocess the CIFAR-10 dataset

 Define the CNN architecture using Keras

 Compile the model

 Train the model

 Evaluate the model on test data

 Display results

**Implementation:**

*# Step 1: Import libraries*

*import tensorflow as tf*

*from tensorflow.keras import layers, models*

*from tensorflow.keras.datasets import cifar10*

*import matplotlib.pyplot as plt*

*import numpy as np*

*# Step 2: Load and preprocess the dataset*

*(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()*

*x\_train, x\_test = x\_train / 255.0, x\_test / 255.0  # Normalize*

*y\_train = y\_train.flatten()*

*y\_test = y\_test.flatten()*

*# Step 3: 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, activation='softmax')  # 10 classes])*

*# Step 4: Compile the model*

*model.compile(optimizer='adam',*

*loss='sparse\_categorical\_crossentropy',*

*metrics=['accuracy'])*

*# Step 5: Train the model*

*history = model.fit(x\_train, y\_train, epochs=10,*

*validation\_data=(x\_test, y\_test),*

*batch\_size=64)*

*# Step 6: Evaluate the model*

*test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=2)*

*print(f"\nTest accuracy: {test\_acc:.2f}")*

*# Step 7: Plot training and validation accuracy*

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

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

*plt.xlabel('Epoch')*

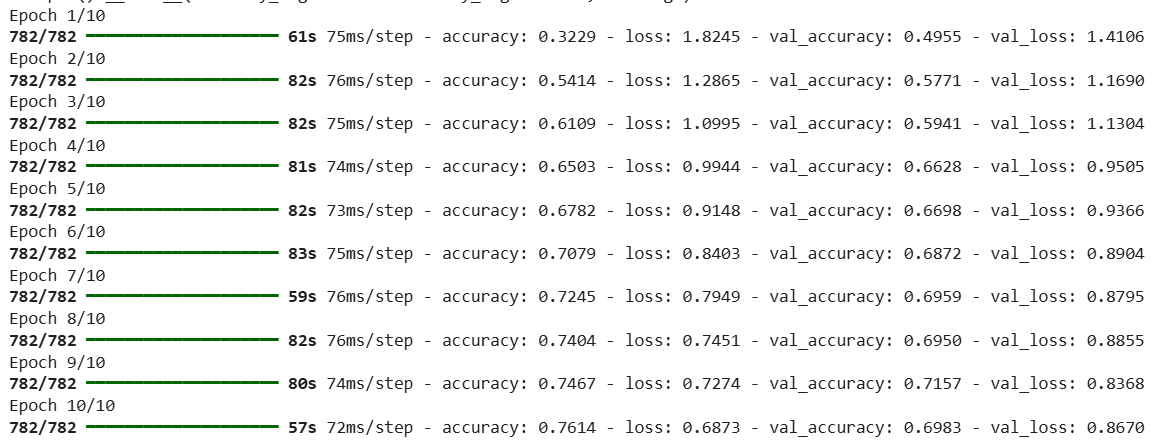
*plt.ylabel('Accuracy')*

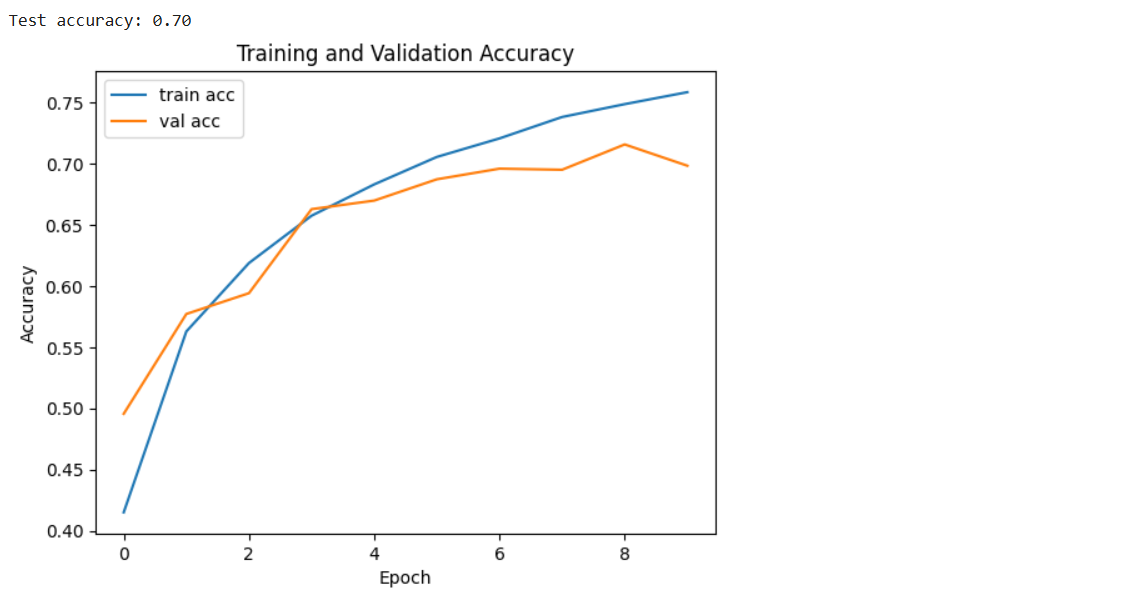
*plt.legend()*

*plt.title('Training and Validation Accuracy')*

*plt.show()*

**Output:**

****

****

**Conclusion:**

The implemented CNN model successfully classifies images from the CIFAR-10 dataset with reasonable accuracy. The architecture can be further optimized using data augmentation, dropout, or hyperparameter tuning for better performance.