Step 1: Import Libraries

import tensorflow as tf

from tensorflow.keras import datasets, layers, models

import matplotlib.pyplot as plt

Step 2: Load and Preprocess Data

# Load the CIFAR-10 dataset

(train\_images, train\_labels), (test\_images, test\_labels) = datasets.cifar10.load\_data()

# Normalize pixel values to be between 0 and 1

train\_images, test\_images = train\_images / 255.0, test\_images / 255.0

# Define class names

class\_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Data augmentation

data\_augmentation = tf.keras.Sequential([

layers.experimental.preprocessing.RandomFlip("horizontal", input\_shape=(32, 32, 3)),

layers.experimental.preprocessing.RandomRotation(0.1),

layers.experimental.preprocessing.RandomZoom(0.1),

])

Step 3: Build the CNN Model

model = models.Sequential([

data\_augmentation,

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.Dropout(0.5), # Dropout to handle overfitting

layers.Dense(10) # 10 output units for 10 classes

])

Step 4: Compile the Model

model.compile(optimizer='adam',

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

metrics=['accuracy'])

Step 5: Train the Model

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

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

Step 6: Evaluate the Model

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

print("\nTest accuracy:", test\_acc)

Step 7: Plot Training and Validation Accuracy and Los

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

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

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

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.ylim([0, 1])

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.ylim([0, 1.5])

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()