import tensorflow as tf  
from tensorflow.keras import layers, models, datasets  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
import matplotlib.pyplot as plt  
import numpy as np  
  
# Load the CIFAR-10 dataset  
(x\_train, y\_train), (x\_test, y\_test) = datasets.cifar10.load\_data()  
  
# Resize images to a standard size and normalize pixel values  
x\_train\_resized = tf.image.resize(x\_train, (32, 32)) / 255.0  
x\_test\_resized = tf.image.resize(x\_test, (32, 32)) / 255.0  
  
# Create an ImageDataGenerator with data augmentation  
datagen = ImageDataGenerator(  
 rotation\_range=15,  
 width\_shift\_range=0.1,  
 height\_shift\_range=0.1,  
 horizontal\_flip=True,  
 zoom\_range=0.1  
)  
  
# Reshape the images  
x\_train\_resized = tf.reshape(x\_train\_resized, (-1, 32, 32, 3))  
  
  
# Generate augmented images  
augmented\_images = datagen.flow(x\_train\_resized, y\_train, batch\_size=1)  
  
# # Display some augmented images  
# num\_images\_to\_display = 5  
#  
# plt.figure(figsize=(10, 2))  
# for i in range(num\_images\_to\_display):  
# augmented\_image, \_ = augmented\_images.next()  
# plt.subplot(1, num\_images\_to\_display, i + 1)  
# plt.imshow(augmented\_image[0])  
# plt.title(f'Class: {y\_train[i][0]}')  
# plt.axis('off')  
#  
# plt.show()  
#  
  
validation\_split = 0.2  
split\_index = int((1 - validation\_split) \* len(x\_train))  
  
x\_train, x\_val = x\_train[:split\_index], x\_train[split\_index:]  
y\_train, y\_val = y\_train[:split\_index], y\_train[split\_index:]  
  
# Define the CNN architecture  
model = models.Sequential()  
  
# Convolutional and Pooling Layers  
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'))  
  
# Flatten layer to transition from convolutional to dense layers  
model.add(layers.Flatten())  
  
# Fully Connected Layers  
model.add(layers.Dense(64, activation='relu'))  
model.add(layers.Dense(10, activation='softmax')) # Output layer with 10 classes for CIFAR-10  
  
# Compile the model  
model.compile(optimizer='adam',  
 loss='sparse\_categorical\_crossentropy',  
 metrics=['accuracy'])  
  
# Display the model summary  
model.summary()  
  
# Train the model with validation data and monitor progress  
history = model.fit(x\_train, y\_train, epochs=10, validation\_data=(x\_val, y\_val), batch\_size=64)  
  
# Evaluate the model on the test set  
test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=2)  
print("\nTest Accuracy:", test\_acc)