**CODE**

**# Import necessary libraries**

**import tensorflow as tf**

**from tensorflow.keras import datasets, layers, models**

**import matplotlib.pyplot as plt**

**import numpy as np**

**# Load and preprocess 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 for CIFAR-10 dataset**

**class\_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',**

**'dog', 'frog', 'horse', 'ship', 'truck']**

**# Build the CNN model**

**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')**

**])**

**# Compile the model**

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

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

**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}")**

**# Visualize training results**

**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()**

**# Save the model**

**model.save('image\_recognition\_model.h5')**

**# Make predictions on test images**

**predictions = model.predict(test\_images)**

**# Function to plot image and prediction results**

**def plot\_image(i, predictions\_array, true\_label, img):**

**true\_label, img = true\_label[i][0], img[i]**

**plt.grid(False)**

**plt.xticks([])**

**plt.yticks([])**

**plt.imshow(img, cmap=plt.cm.binary)**

**predicted\_label = np.argmax(predictions\_array)**

**if predicted\_label == true\_label:**

**color = 'blue'**

**else:**

**color = 'red'**

**plt.xlabel(f"{class\_names[predicted\_label]} {100\*np.max(predictions\_array):2.0f}% ({class\_names[true\_label]})",**

**color=color)**

**def plot\_value\_array(i, predictions\_array, true\_label):**

**true\_label = true\_label[i][0]**

**plt.grid(False)**

**plt.xticks(range(10))**

**plt.yticks([])**

**thisplot = plt.bar(range(10), predictions\_array, color="#777777")**

**plt.ylim([0, 1])**

**predicted\_label = np.argmax(predictions\_array)**

**thisplot[predicted\_label].set\_color('red')**

**thisplot[true\_label].set\_color('blue')**

**# Display a few test images with predictions**

**num\_rows = 5**

**num\_cols = 3**

**num\_images = num\_rows \* num\_cols**

**plt.figure(figsize=(2\*2\*num\_cols, 2\*num\_rows))**

**for i in range(num\_images):**

**plt.subplot(num\_rows, 2\*num\_cols, 2\*i+1)**

**plot\_image(i, predictions[i], test\_labels, test\_images)**

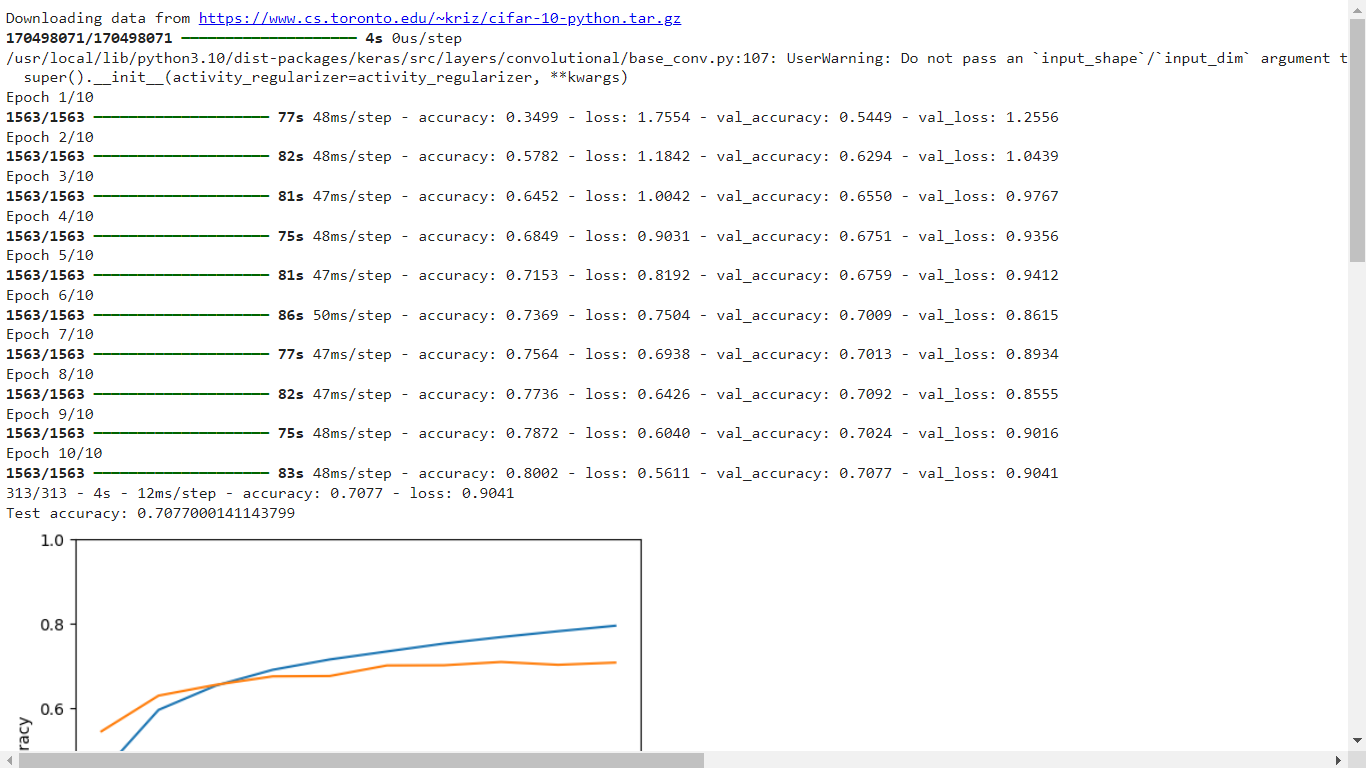
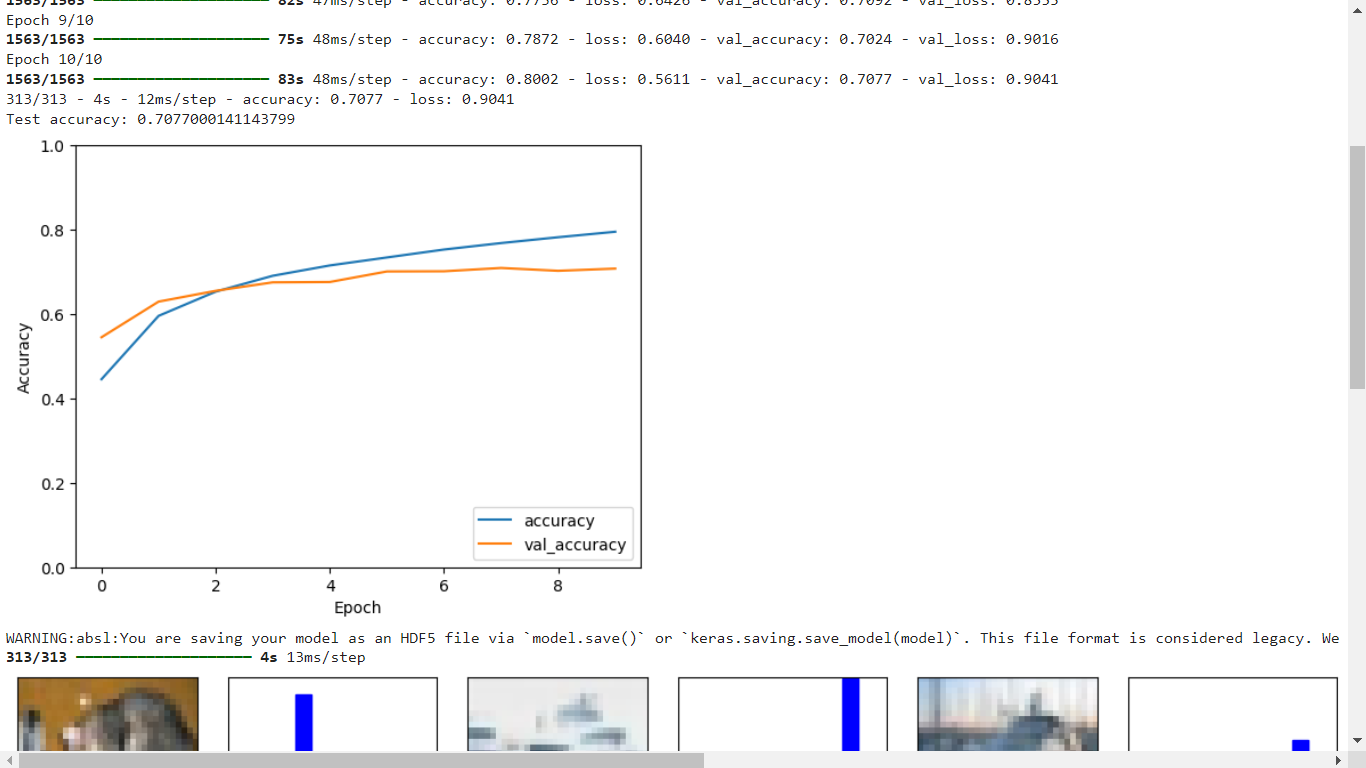
**plt.subplot(num\_rows, 2\*num\_cols, 2\*i+2)**

**plot\_value\_array(i, predictions[i], test\_labels)**

**plt.tight\_layout()**

**plt.show()**

**OUTPUT**

****

****