analyze python image classification task

[paste task ]

the folder structure is - python code file - data |-test |-train both test and train contain jpg folders cat and dog with images of the respective give task wise code with code split into tasks with function based approaches for simplicity and and folder structure into consideration

[you will get version 1]

modify code to include # Plot the training and validation accuracy and loss plt.plot(history.history['accuracy'], label='Training Accuracy') plt.plot(history.history['val\_accuracy'], label='Validation Accuracy') plt.legend() plt.title('Training and Validation Accuracy') plt.xlabel('Epoch') plt.ylabel('Accuracy') plt.show() and plt.plot(history.history['loss'], label='Training Loss') plt.plot(history.history['val\_loss'], label='Validation Loss') plt.legend() plt.title('Training and Validation Loss') plt.xlabel('Epoch') plt.ylabel('Loss') plt.show() and from sklearn.metrics import classification\_report # Use the predict() method to get the predicted labels for the test data # Use the model to make predictions on the test data y\_pred\_prob = model.predict(test\_dataset) # Convert the predicted probabilities to class indices y\_pred = np.argmax(y\_pred\_prob, axis=1) # Print the classification report print(classification\_report(y\_test, y\_pred)) and # Save the trained model to a file model.save('vitMixup.h5') from sklearn.metrics import confusion\_matrix # Use the model to make predictions on the test data y\_pred = np.argmax(model.predict(test\_dataset), axis=1) # Create the confusion matrix conf\_mat = confusion\_matrix(y\_test, y\_pred) # Print the confusion matrix print(conf\_mat) and import seaborn as sns import matplotlib.pyplot as plt # Use the model to make predictions on the test data y\_pred = np.argmax(model.predict(test\_dataset), axis=1) # Create the confusion matrix conf\_mat = confusion\_matrix(y\_test, y\_pred) # Create a list of class labels class\_names = ['cat,dog '] # Create a heatmap of the confusion matrix sns.heatmap(conf\_mat, annot=True, fmt='d', cmap='Blues', xticklabels=class\_names, yticklabels=class\_names) # Add labels and title plt.xlabel('Predicted Class') plt.ylabel('True Class') plt.title('Confusion Matrix') # Display the plot plt.show()

[you will get version 2]

--------------------------------------------------------------------------- KeyError Traceback (most recent call last) Cell In[13], line 14 11 plot\_confusion\_matrix(model, test\_generator) 13 if \_\_name\_\_ == '\_\_main\_\_': ---> 14 main() Cell In[13], line 7, in main() 5 history = train\_model(model, train\_generator) 6 evaluate\_model(model, test\_generator) ----> 7 plot\_accuracy(history) 8 plot\_loss(history) 9 print\_classification\_report(model, test\_generator) Cell In[7], line 4, in plot\_accuracy(history) 2 def plot\_accuracy(history): 3 plt.plot(history.history['accuracy'], label='Training Accuracy') ----> 4 plt.plot(history.history['val\_accuracy'], label='Validation Accuracy') 5 plt.legend() 6 plt.title('Training and Validation Accuracy') KeyError: 'val\_accuracy' this requires avalidation split of 0.8 train and 0.2 in validation make corrections in code accordingly

[you will get version 3]

generate me a readme as code with 3 word title 200 character description Brief about dataset Brief about processing techniques Brief about model architecture brief about training and valuation and space for screenshots as per task name

[you will get your readme for github]