```
!kaggle datasets download -d nodoubttome/skin-cancer9-classesisic
```

```
Dataset URL: https://www.kaggle.com/datasets/nodoubttome/skin-cancer9-classesisic
License(s): other
Downloading skin-cancer9-classesisic.zip to /content
100% 785M/786M [00:38<00:00, 22.9MB/s]
100% 786M/786M [00:38<00:00, 21.3MB/s]
```

```
import zipfile
zip_ref = zipfile.ZipFile('/content/skin-cancer9-classesisic.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

Start coding or generate with AI.

```
import tensorflow as tf
import numpy as np
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
import cv2
import matplotlib.pyplot as plt

# Load the dataset
data_dir = "/content/Skin cancer ISIC The International Skin Imaging Collaboration/Tra

# Define image size
img_height, img_width = 128, 128

# Data Preprocessing
def load_data(data_dir):
    images = []
    labels = []
    classes = os.listdir(data_dir)

    for label in classes:
        class_dir = os.path.join(data_dir, label)
        if os.path.isdir(class_dir):
            for img in os.listdir(class_dir):
                img_path = os.path.join(class_dir, img)
                image = cv2.imread(img_path)
                image = cv2.resize(image, (img_height, img_width))
                images.append(image)
                labels.append(classes.index(label))

    images = np.array(images)
    labels = np.array(labels)

    return images, labels

images, labels = load_data(data_dir)

# Normalize images
images = images / 255.0

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, ran

# Create ImageDataGenerator for data augmentation
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
)

datagen.fit(X_train)
```

```
print(X_test.shape)
```

```
(448, 128, 128, 3)
```

eError NameError AttributeError •••

Gemini

Hypothesis

The code is trying to access a column (index 4) that doesn't exist in the `history` DataFrame, which only has 4 columns (indices 0 to 3). This is happening because the code is assuming that the `history` object contains validation metrics (e.g., 'val_loss', 'val_accuracy'), but it seems like the model was not trained with a validation set, or the validation metrics were not properly recorded during training.

Suggested changes

```
# Assuming history is the History object returne
import pandas as pd

# Check if history is already a DataFrame
if not isinstance(history, pd.DataFrame):
    # Convert the history object to a Pandas Data
    history = pd.DataFrame(history.history)

plt.figure(figsize=(15,10))

# Plot only the available metrics
plt.subplot(2,2,1)
plt.plot(history.iloc[:,0],'b-o',label='Train_Lo
# Check if the DataFrame has a column at index 4
if len(history.columns) > 4:
    plt.plot(history.iloc[:,4],'r-o',label='Vali
plt.title('Train_Loss',fontsize=20)
plt.legend()

plt.subplot(2,2,2)
plt.plot(history.iloc[:,1],'b-o',label='Train_Ac
# Check if the DataFrame has a column at index 5
if len(history.columns) > 5:
    plt.plot(history.iloc[:,5],'r-o',label='Vali
plt.title('Train_Accuracy',fontsize=20)
plt.legend()
```

Use code with caution

Sources

1. stackoverflow.com/questions/73968053/how-to-plot-loss-terms-and-accuracy-in-keras
2. stackoverflow.com/questions/47633789/keras-history-not-accessible-for-loss-or-accuracy
3. machinelearningmastery.com/plotting-the-training-and-validation-loss-curves-for-the-transformer-model/
4. github.com/tensorflow/tensorflow/issues/60087
5. datascience.stackexchange.com/questions/4595 plotting-loss-and-mse

Rate this answer

Enter a prompt here

0 / 400

Responses may display inaccurate or offensive information that doesn't represent Google's views. Learn more