BHANU PRAKASH MANNAVA
Oct 22 · 3 min read · Listen

MULTICLASS CLASSIFICATION USING TENSORFLOW

I am BHANU PRAKASH MANNAVA and this project is to classify multiclass images using CNN.

To complete this project I have refered to the work done by MARYNA ANTONEVYCH on kaggle and nichnacnoch on github.

Our dataset consists of images like-airplanes,schonner and motorbikes-Dataset link-https://www.kaggle.com/datasets/maricinnamon/caltech101-airplanes-motorbikes-schooners.

Step:1- Importing all the required libraries.

```
import numpy as np
from matplotlib import pyplot as plt
import tensorflow as tf
import os
import cv2
import imghdr
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense,
Flatten, Dropout
Import PIL
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping
```

Step-3 Data Pre-processing

step-3a- store all the images in the data directory and printing the sample images in six coloums

```
data = tf.keras.utils.image_dataset_from_directory(data_dir)
batch = data_iterator.next()

#printing the sample images
fig, ax = plt.subplots(ncols=6 ,figsize=(20,20))
for idx, img in enumerate(batch[0][:6]):
ax[idx].imshow(img.astype(int))
ax[idx].title.set_text(batch[1][idx])
```

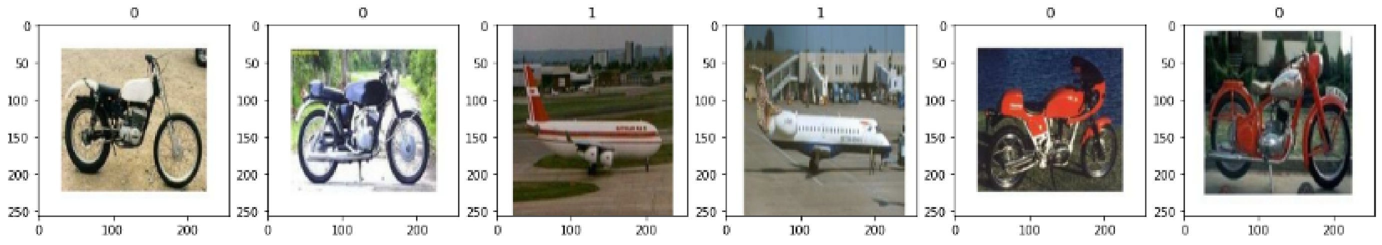


Fig-1:Sample images from our data

step-3b-Scaling

In our dataset we have three different attributes with difference in their numbers .

To scale all the number and to decrease the difference we scale the data.

```
data = data.map(lambda x,y: (x/255, y))
```

STEP-4- Split train and test data

```
#splitting train,test,dev.
train_size = int(len(data)*.6)
val_size = int(len(data)*.2)
test_size = int(len(data)*.2)
```

STEP-5- TENSORFLOW

Here we train the set of images from the dataset by adding the layes such as convolutional,activation function(relu,sigmoid),maxpooling

Dense layer is also added and 3 is the no.of attributes(motorbikes,airplanes,schooners) in the dense layer

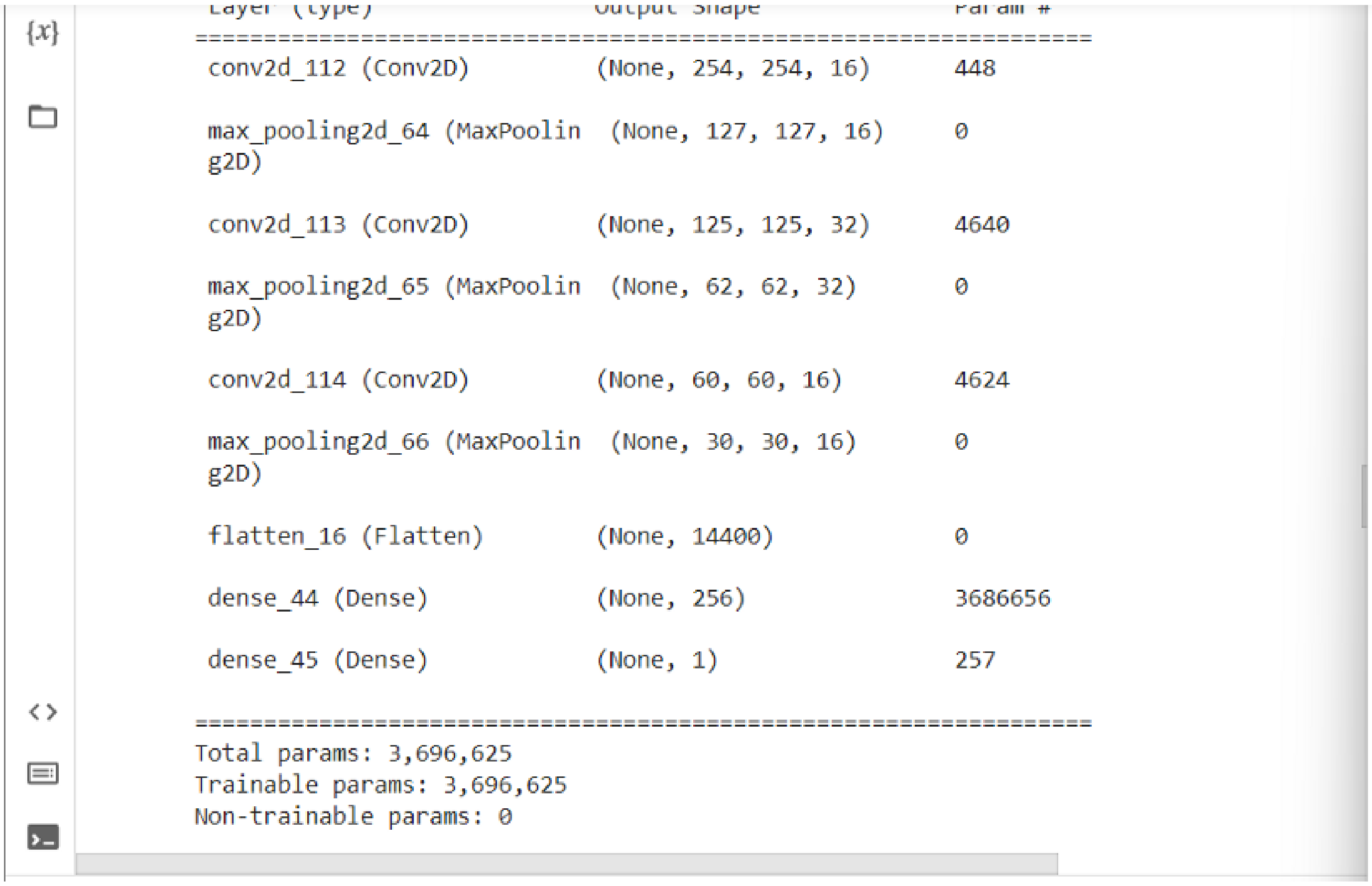
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense,
Flatten, Dropout

model=Sequential()
model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=
(256,256,3)))
model.add(MaxPooling2D())
model.add(Conv2D(32, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(16, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(3, activation='sigmoid'))
```

Summary of sequential model

```
model.summary
```

Get unlimited access Open in app



```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_112 (Conv2D)         (None, 254, 254, 16)      448

 max_pooling2d_64 (MaxPoolin  (None, 127, 127, 16)     0
 g2D)

 conv2d_113 (Conv2D)         (None, 125, 125, 32)      4640

 max_pooling2d_65 (MaxPoolin  (None, 62, 62, 32)       0
 g2D)

 conv2d_114 (Conv2D)         (None, 60, 60, 16)        4624

 max_pooling2d_66 (MaxPoolin  (None, 30, 30, 16)       0
 g2D)

 flatten_16 (Flatten)        (None, 14400)             0

 dense_44 (Dense)            (None, 256)               3686656

 dense_45 (Dense)            (None, 1)                 257

=================================================================
Total params: 3,696,625
Trainable params: 3,696,625
Non-trainable params: 0
```

FIG-1: Summary

STEP:6 — Train the Model

Callbacks function helps to complete the next function after completing the previous function.

EPOCHS- Hyperparameter where an algorithm learns n number of times and works with the training data.Here no .of epochs we take is 5

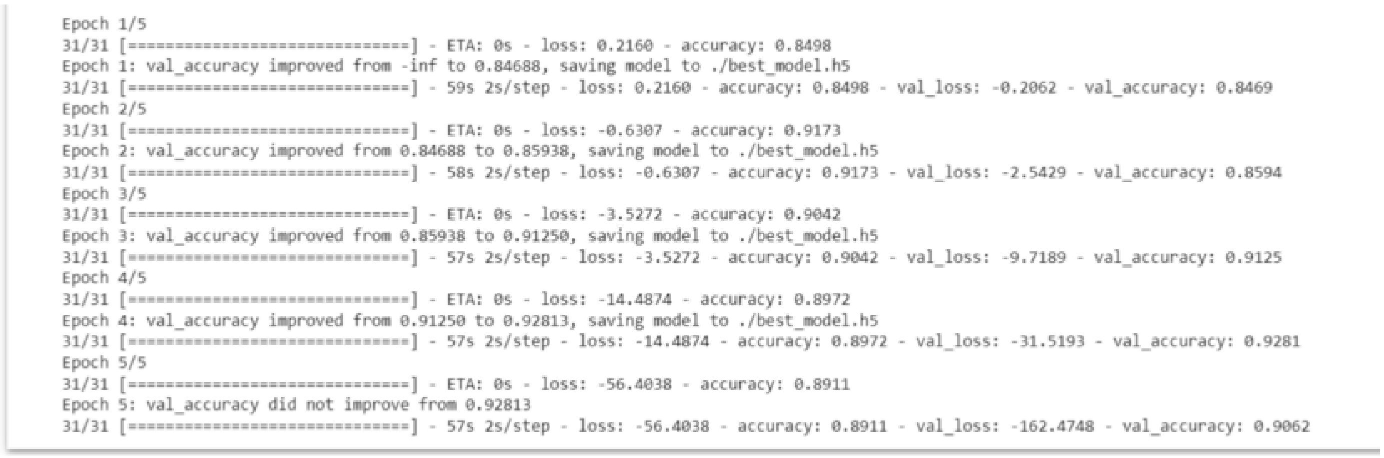
Checkpoints are just the snapshots of our work done so far.

```
from tensorflow.keras.models import Sequential

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

callbacks = EarlyStopping(monitor='val_loss', patience=3, verbose=1,
mode='auto')
```

```
hist = model.fit(train, epochs=5,verbose=1, validation_data=val,
callbacks=[model_callback])
```

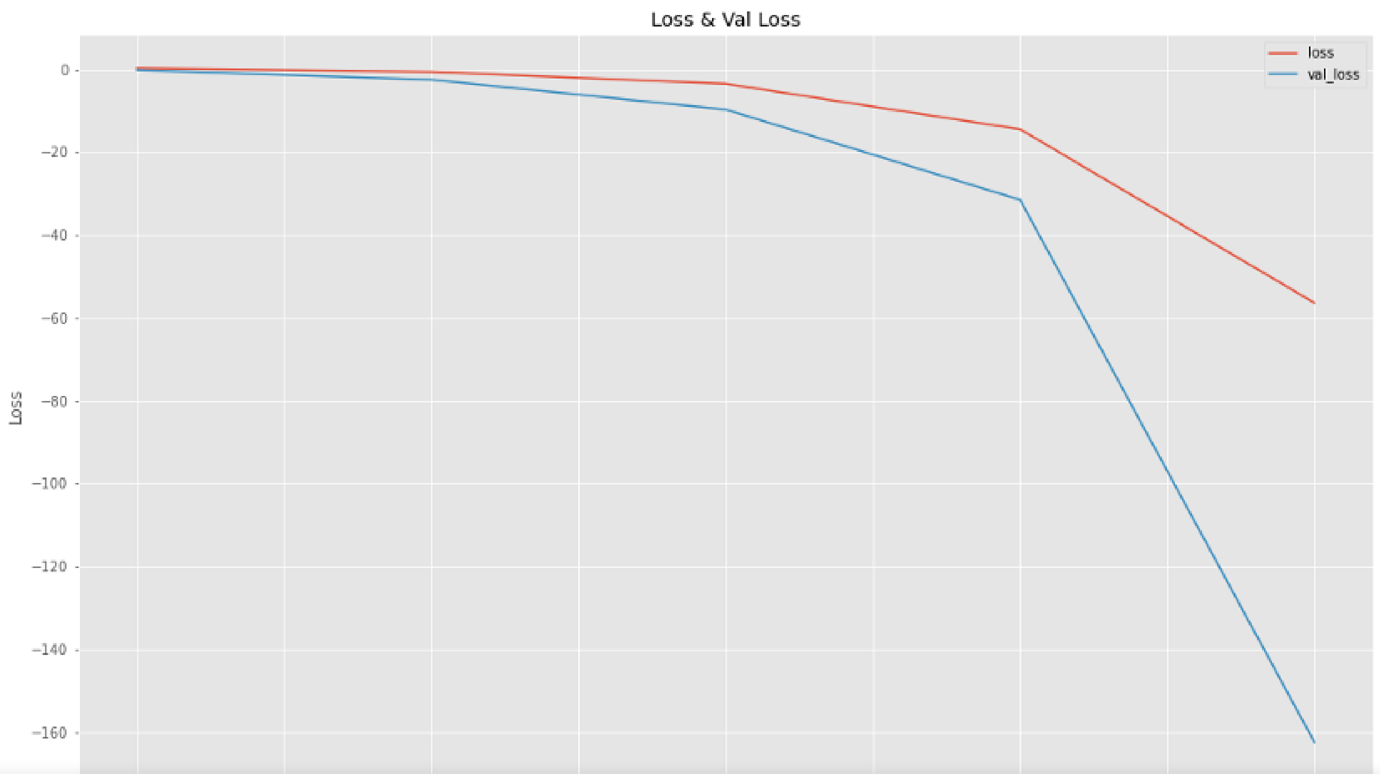


```
Epoch 1/5
31/31 [==============================] - ETA: 0s - loss: 0.2160 - accuracy: 0.8498
Epoch 1: val_accuracy improved from -inf to 0.84688, saving model to ./best_model.h5
31/31 [==============================] - 59s 2s/step - loss: 0.2160 - accuracy: 0.8498 - val_loss: -0.2062 - val_accuracy: 0.8469
Epoch 2/5
31/31 [==============================] - ETA: 0s - loss: -0.6307 - accuracy: 0.9173
Epoch 2: val_accuracy improved from 0.84688 to 0.85938, saving model to ./best_model.h5
31/31 [==============================] - 58s 2s/step - loss: -0.6307 - accuracy: 0.9173 - val_loss: -2.5429 - val_accuracy: 0.8594
Epoch 3/5
31/31 [==============================] - ETA: 0s - loss: -3.5272 - accuracy: 0.9042
Epoch 3: val_accuracy improved from 0.85938 to 0.91250, saving model to ./best_model.h5
31/31 [==============================] - 57s 2s/step - loss: -3.5272 - accuracy: 0.9042 - val_loss: -9.7189 - val_accuracy: 0.9125
Epoch 4/5
31/31 [==============================] - ETA: 0s - loss: -14.4874 - accuracy: 0.8972
Epoch 4: val_accuracy improved from 0.91250 to 0.92813, saving model to ./best_model.h5
31/31 [==============================] - 57s 2s/step - loss: -14.4874 - accuracy: 0.8972 - val_loss: -31.5193 - val_accuracy: 0.9281
Epoch 5/5
31/31 [==============================] - ETA: 0s - loss: -56.4038 - accuracy: 0.8911
Epoch 5: val_accuracy did not improve from 0.92813
31/31 [==============================] - 57s 2s/step - loss: -56.4038 - accuracy: 0.8911 - val_loss: -162.4748 - val_accuracy: 0.9062
```

FIG-2: LOSS VALUE AND ACCURACY VALUE

Step:7- Plotting graphs

This is the plot of loss for both training and validation set



— GRAPH OF ACCURACIES.

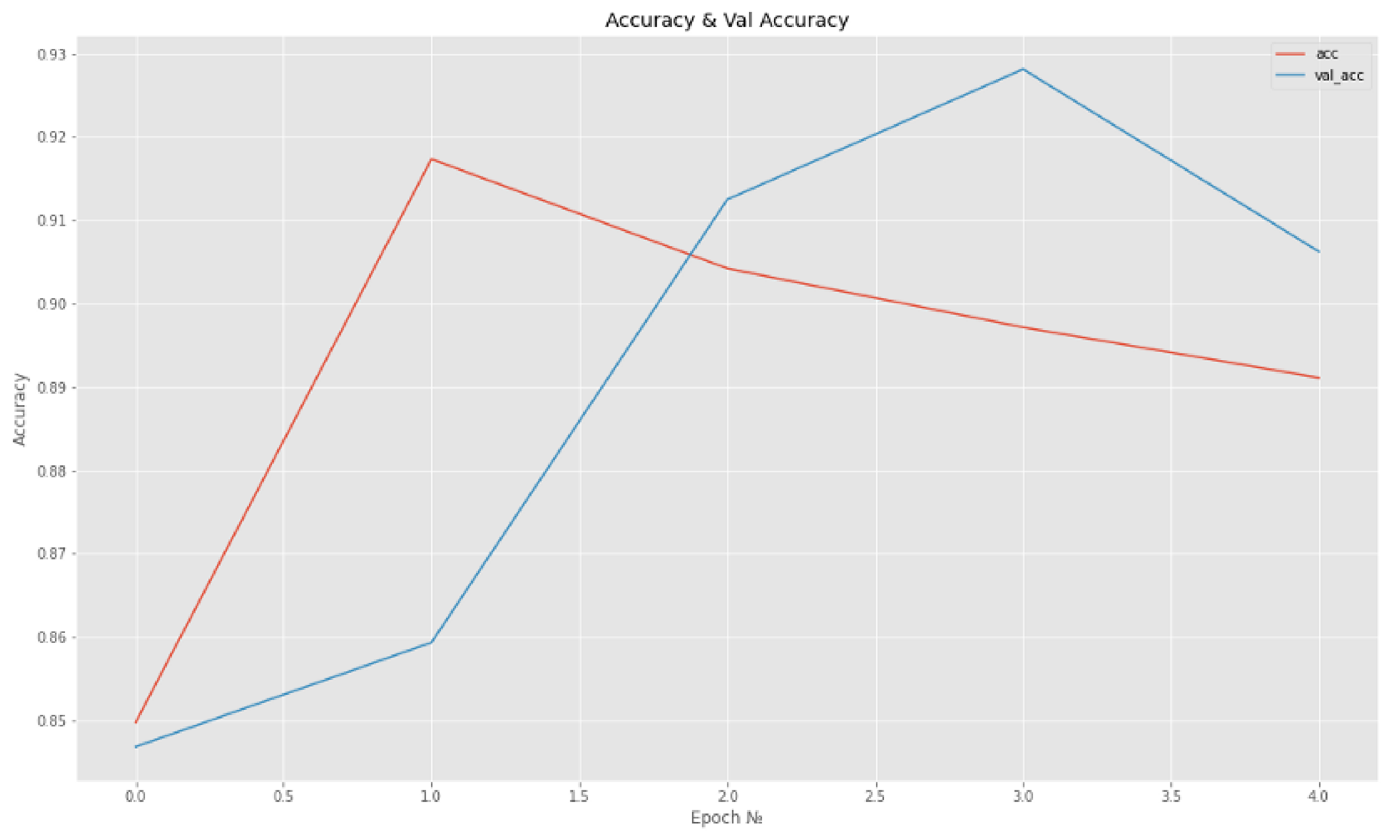


FIG-4:Graph between accuracy and value accuracy

MY CONTRIBUTION

In this project I Plotted the graph of accuracies and loss between training and validation set.

I also improved the accuracy from 70(from my reffered model)to the accuracy around 90 by applying scaling.

I also worked with multiple classes where reffered model consists of 2 classes.

REFERENCES

1-Kaggle-**https://www.kaggle.com/datasets/maricinnamon/caltech101-airplanes-motorbikes-schooners**

pynb