Multi Class Image Classification

Mounting Drive

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
import os
os.environ['KAGGLE_CONFIG_DIR']='/content/gdrive/My Drive/Kaggle_dataset'
```

```
%cd /content/gdrive/My Drive/Kaggle_dataset
%cd multiclassimagedatasetairplanecar/
```

```
/content/gdrive/My Drive/Kaggle_dataset
/content/gdrive/My Drive/Kaggle_dataset/multiclassimagedatasetairplanecar
```

Showing images From Data Set

```
import cv2
from google.colab.patches import cv2_imshow
img=cv2.imread('/content/gdrive/MyDrive/Kaggle_dataset/multiclassimagedatasetairplanecar/Dataset/test/airplanes/airplane100.jpg')
img = cv2.resize(img , (200 , 200))
cv2_imshow(img)
```


```
import tensorflow as tf
# from keras.preprocessing import image_dataset_from_directory

data_train = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/gdrive/MyDrive/Kaggle_dataset/multiclassimagedatasetairplanecar/Dataset/train',
    labels = "inferred", label_mode = "categorical", image_size = [128 , 128],
    interpolation = 'nearest', batch_size = 64,
    shuffle = True
)

```

```
data_valid = tf.keras.preprocessing.image_dataset_from_directory(
    '/content/gdrive/MyDrive/Kaggle_dataset/multiclassimagedatasetairplanecar/Dataset/test',
    labels = "inferred", label_mode = "categorical", image_size = [128 , 128],
    interpolation = 'nearest',  batch_size = 64,
    shuffle = False
)

def convert_to_float(image , label):
    image = tf.image.convert_image_dtype(image,dtype=tf.float32)
    return image,label
```

```
Found 3000 files belonging to 3 classes.
Found 582 files belonging to 3 classes.
```

Some Important Imports

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPool2D
from keras.layers import Flatten
from keras.layers import Dense
```

Preproccessing

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator( rescale=1./255, shear_range=0.2, zoom_range=0.2,  horizontal_flip=True )

test_datagen = ImageDataGenerator( rescale=1./255 )

train_set = train_datagen.flow_from_directory(
    '/content/gdrive/MyDrive/Kaggle_dataset/multiclassimagedatasetairplanecar/Dataset/train',
    target_size = (128,128),
    batch_size = 32,
    class_mode = "categorical"


)

test_set = test_datagen.flow_from_directory(
    '/content/gdrive/MyDrive/Kaggle_dataset/multiclassimagedatasetairplanecar/Dataset/test',
    target_size = (128,128),
    batch_size = 32,
    class_mode = "categorical"


)
```

```
Found 3000 images belonging to 3 classes.
Found 582 images belonging to 3 classes.
```

Building ConveNet (CNN) Model

```
import keras.layers.experimental.preprocessing as preprocessing
import tensorflow as tf

model = Sequential()

model.add(Conv2D(filters=32 , kernel_size=5 ,input_shape=[128,128,3] ,activation="relu" ,padding="SAME"))
model.add(MaxPool2D())

model.add(Conv2D(filters=64 , kernel_size=3 ,activation="relu" ,padding="SAME"))
model.add(MaxPool2D())

model.add(Conv2D(filters=128 , kernel_size=3 ,activation="relu" ,padding="SAME"))
model.add(MaxPool2D())

model.add(Flatten())
model.add(Dense(units=6 , activation="relu"))
model.add(Dense(units=3 , activation="softmax"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      2432      
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 128)       73856     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 128)       0         
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0         
_________________________________________________________________
dense (Dense)                (None, 6)                 196614    
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 21        
=================================================================
Total params: 291,419
Trainable params: 291,419
Non-trainable params: 0
_________________________________________________________________
```

Training of Model

```
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```

```
    metrics=['accuracy'])
)

history = model.fit(
    train_set,
    validation_data = test_set,
    epochs = 30
)
```

```
94/94 [==============================] - 56s 590ms/step - loss: 0.1258 - accuracy: 0.9634 - val_loss: 0.3679 - val_accuracy: 0.9244
Epoch 2/30
94/94 [==============================] - 54s 578ms/step - loss: 0.1047 - accuracy: 0.9706 - val_loss: 0.3935 - val_accuracy: 0.8883
Epoch 3/30
94/94 [==============================] - 54s 578ms/step - loss: 0.0836 - accuracy: 0.9807 - val_loss: 0.4152 - val_accuracy: 0.9296
Epoch 4/30
94/94 [==============================] - 54s 570ms/step - loss: 0.1132 - accuracy: 0.9722 - val_loss: 0.4320 - val_accuracy: 0.8832
Epoch 5/30
94/94 [==============================] - 54s 574ms/step - loss: 0.0862 - accuracy: 0.9744 - val_loss: 0.4515 - val_accuracy: 0.8900
Epoch 6/30
94/94 [==============================] - 55s 588ms/step - loss: 0.1091 - accuracy: 0.9717 - val_loss: 0.3305 - val_accuracy: 0.9381
Epoch 7/30
94/94 [==============================] - 54s 577ms/step - loss: 0.0863 - accuracy: 0.9752 - val_loss: 0.4952 - val_accuracy: 0.8625
Epoch 8/30
94/94 [==============================] - 54s 576ms/step - loss: 0.0886 - accuracy: 0.9726 - val_loss: 0.3458 - val_accuracy: 0.9158
Epoch 9/30
94/94 [==============================] - 54s 578ms/step - loss: 0.0835 - accuracy: 0.9779 - val_loss: 0.3076 - val_accuracy: 0.9210
Epoch 10/30
94/94 [==============================] - 54s 573ms/step - loss: 0.0870 - accuracy: 0.9720 - val_loss: 0.3935 - val_accuracy: 0.9227
Epoch 11/30
94/94 [==============================] - 54s 578ms/step - loss: 0.0565 - accuracy: 0.9878 - val_loss: 0.6518 - val_accuracy: 0.8282
Epoch 12/30
94/94 [==============================] - 55s 583ms/step - loss: 0.0846 - accuracy: 0.9740 - val_loss: 0.3253 - val_accuracy: 0.9278
Epoch 13/30
94/94 [==============================] - 54s 572ms/step - loss: 0.0581 - accuracy: 0.9846 - val_loss: 0.4839 - val_accuracy: 0.8935
Epoch 14/30
94/94 [==============================] - 54s 572ms/step - loss: 0.0535 - accuracy: 0.9859 - val_loss: 0.3411 - val_accuracy: 0.9244
Epoch 15/30
94/94 [==============================] - 54s 572ms/step - loss: 0.0491 - accuracy: 0.9885 - val_loss: 0.4592 - val_accuracy: 0.8814
Epoch 16/30
94/94 [==============================] - 56s 596ms/step - loss: 0.0553 - accuracy: 0.9858 - val_loss: 0.3537 - val_accuracy: 0.9175
Epoch 17/30
94/94 [==============================] - 55s 583ms/step - loss: 0.0598 - accuracy: 0.9860 - val_loss: 0.3330 - val_accuracy: 0.9021
Epoch 18/30
94/94 [==============================] - 55s 588ms/step - loss: 0.0674 - accuracy: 0.9809 - val_loss: 0.3624 - val_accuracy: 0.9141
Epoch 19/30
94/94 [==============================] - 56s 599ms/step - loss: 0.0470 - accuracy: 0.9857 - val_loss: 0.4277 - val_accuracy: 0.9175
Epoch 20/30
94/94 [==============================] - 55s 585ms/step - loss: 0.0491 - accuracy: 0.9895 - val_loss: 0.4030 - val_accuracy: 0.9192
Epoch 21/30
94/94 [==============================] - 54s 579ms/step - loss: 0.0546 - accuracy: 0.9812 - val_loss: 0.3741 - val_accuracy: 0.9261
Epoch 22/30
94/94 [==============================] - 54s 571ms/step - loss: 0.0397 - accuracy: 0.9900 - val_loss: 0.4085 - val_accuracy: 0.9278
Epoch 23/30
94/94 [==============================] - 55s 581ms/step - loss: 0.0404 - accuracy: 0.9898 - val_loss: 0.2734 - val_accuracy: 0.9175
Epoch 24/30
94/94 [==============================] - 55s 586ms/step - loss: 0.0736 - accuracy: 0.9705 - val_loss: 0.4147 - val_accuracy: 0.9175
Epoch 25/30
94/94 [==============================] - 55s 589ms/step - loss: 0.0427 - accuracy: 0.9872 - val_loss: 0.3111 - val_accuracy: 0.9192
Epoch 26/30
94/94 [==============================] - 55s 581ms/step - loss: 0.0671 - accuracy: 0.9813 - val_loss: 0.3871 - val_accuracy: 0.9210
Epoch 27/30
```

```
94/94 [==============================] - 55s 584ms/step - loss: 0.0655 - accuracy: 0.9776 - val_loss: 0.2549 - val_accuracy: 0.9296
Epoch 28/30
94/94 [==============================] - 55s 582ms/step - loss: 0.0413 - accuracy: 0.9874 - val_loss: 0.4567 - val_accuracy: 0.9072
Epoch 29/30
94/94 [==============================] - 54s 579ms/step - loss: 0.0363 - accuracy: 0.9904 - val_loss: 0.4716 - val_accuracy: 0.8986
Epoch 30/30
94/94 [==============================] - 55s 581ms/step - loss: 0.0444 - accuracy: 0.9849 - val_loss: 0.3486 - val_accuracy: 0.9244
```

Saving Model

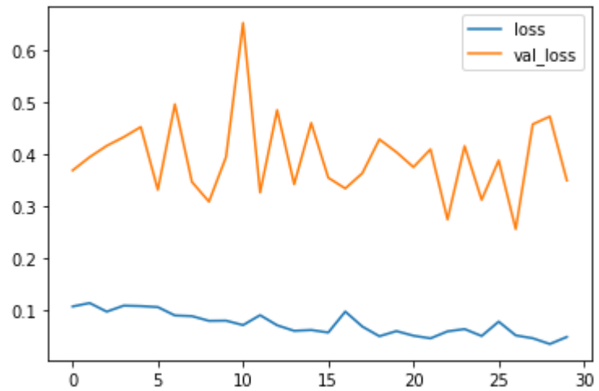
```
model.save('MyModelClassifier.h5')
```

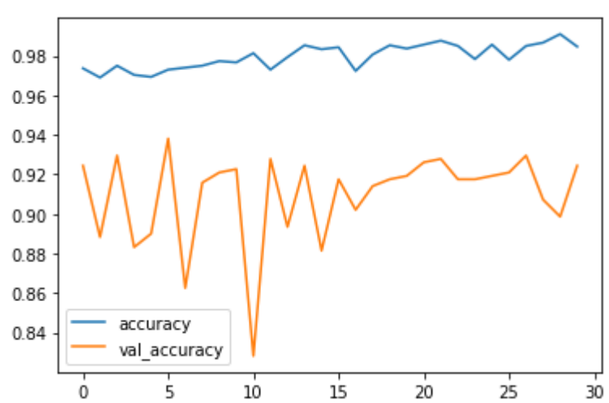
Model Evaluation

```
import pandas as pd
history_frame = pd.DataFrame(history.history)
history_frame.loc[:,['loss','val_loss']].plot()
history_frame.loc[:,['accuracy','val_accuracy']].plot()

```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fec1036b810>
```

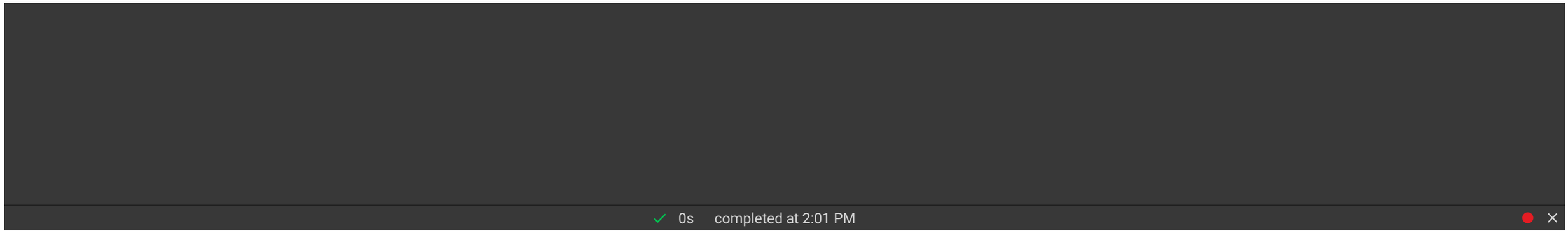




Testing of Model

```
from tensorflow.keras.preprocessing import image
import tensorflow as tf
import cv2
import numpy as np
import matplotlib.pyplot as plt
testPath = '/content/gdrive/MyDrive/Kaggle_dataset/multiclassimagedatasetairplanecar/Dataset/test/cars/cars127.jpg'
img = image.load_img(testPath, target_size=(128,128))
plt.imshow(img)
plt.axis('off')
plt.plot()
img = np.reshape(img,[1,128,128,3])
img_f = tf.cast(img,tf.float32)
cl = model.predict(img_f)
print(cl)
cl = (cl>0.5).astype("int32")
if cl[0][0]>0.5:
    print("AIRPLANE")
elif cl[0][1]>0.5:
    print("CAR")
else:
    print("SHIP")
print(cl)
```

```
[[0. 1. 0.]]
CAR
[[0 1 0]]
```


0s completed at 2:01 PM