IMAGE CLASSFICATION

FINAL REPORT – VEHICLE CLASSIFICATION

By: V Andal Priyadharshini
SRMIST, INDIA

1 CREATION OF A NEW PROJECT IN GOOGLE COLAB

The kind of project that I chose to do was: Vehicle Classification

The rapid increase of vehicles contributes heavily to the air pollution in cities; thus, the optimization of urban transportation is of upmost importance. Vehicle classification plays a significant role in road maintenance, traffic flow modeling, and road safety management and thus is one of the most important tasks of an intelligent transport system.

This report aims to improve the accuracy of automatic vehicle classifiers for imbalanced datasets. Classification is made through utilizing a single anisotropic magnetoresistive sensor, with the models of vehicles involved being classified into hatchbacks, sedans, buses, and multi-purpose vehicles (MPVs). Using time domain and frequency domain features in combination with three common classification algorithms in pattern recognition, we develop a novel feature extraction method for vehicle classification.

Preparation:

We downloaded images from vehicle dataset to train the model for this Project.

Further notice:

The image description specifies the objects are in the image, such as an automobile or bicycle rail, while the image position provides a precise location for these objects by using restricted fields.

In order to distinguish the images, the neural convolution network had to identify various objects, such as a vehicle, a truck, and a motorcycle.

As a consequence, image classification and localization can be described as object detection.

Object Detection = Image Classification + Image Position Workflow has three pieces, the first step is to gather Training data, the second is the training of the model and the final one are the projections of new pictures.

FAST RCNN FOR IMAGE CLASSIFICATION AND LOCALIZATION

QUICK RCNN FOR IMAGE CLASSIFICATION AND LOCATION IN FAST RCNN, WE SEND THE INPUT IMAGE TO CNN, WHICH IN TURN PRODUCES MAPS OF INNOVATIVE ARTIFACTS. THE REGIONS OF THE PROPOSAL ARE DERIVED WITH THESE MAPS.

WE THEN USE THE ROI POOL LAYER TO TURN ALL PROPOSED AREAS INTO FIXED SIZES SO THAT THEY CAN BE MOVED TO A FULLY LINKED NETWORK. THE QUICK METHOD OF THE RCNN IS AS FOLLOWS

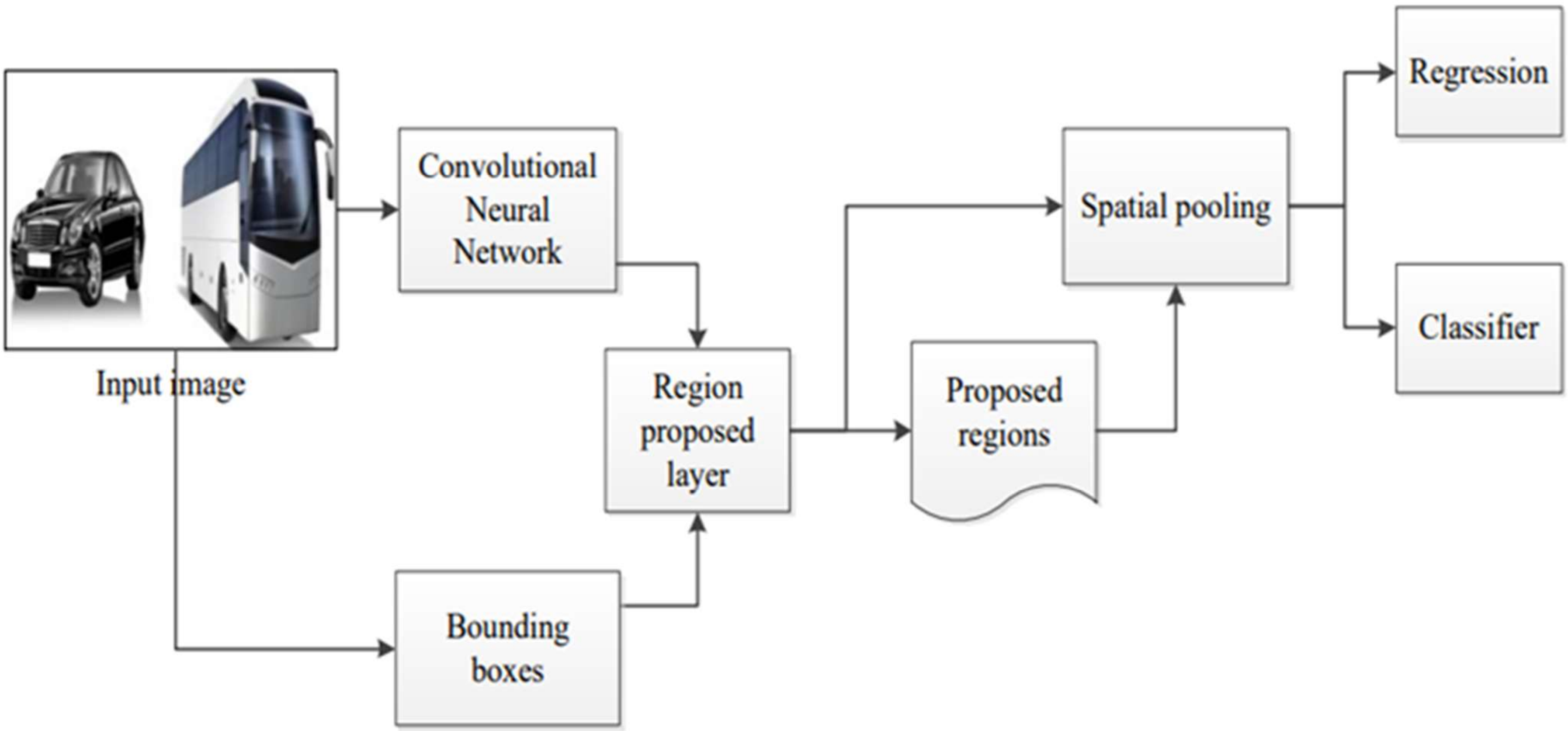
1. TO USE A CAMERA TO TAKE AN INPUT SHOT.

2. THE INPUT IMAGE WILL BE PASSED TO CONVNET, WHICH RETURNS THE AREA OF INTEREST.

3. APPLY THE LEVEL OF THE ROI POOL TO THE REMOVED REGIONS.

FINALLY, THESE AREAS ARE MOVED TO A FULLY LINKED NETWORK, WHICH CLASSIFIES THEM, AND BOUNDING BLOCKS ARE ALSO RETURNED USING THE

NETWORK OF CNN



THE FINAL PROJECT INCLUSION

consist of:

1. Importing data in .csv file from Kaggle to train the model
2. Annotating the images imported from the dataset.
3. Developing architecture for the model.
4. Training your model with the help of dataset available.
5. Capturing Images for the model.
6. Predicting images and writing a short report in *PowerPoint format about the working of the project.*

Copy_of_Vehicle_Classification.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive Connect Editing

```
from google.colab import files
uploaded = files.upload()
!ls -lha kaggle.json
!pip install -q kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle
!chmod 600 ~/.kaggle/kaggle.json
!pip install --upgrade --force-reinstall --no-deps kaggle
!kaggle competitions download -c vehicle
```

```
[ ] !unzip vehicle.zip
```

```
[ ] from matplotlib import pyplot as plt
    import numpy as np
    import os
    import time
    from keras.applications.vgg16 import VGG16
    from keras.preprocessing import image
    from keras.preprocessing.image import img_to_array,load_img
    from keras.applications.imagenet_utils import preprocess_input
    from keras.applications.imagenet_utils import decode_predictions
    from keras.layers import Dense, Activation, Flatten
    from keras.layers import merge, Input
```

Copy_of_Vehicle_Classification.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

Share

+ Code + Text Copy to Drive Connect Editing

```
[ ] from keras.layers import merge, Input
    from keras.models import Model
    from keras.utils import np_utils
    from sklearn.utils import shuffle
    from sklearn.model_selection import train_test_split
    import math
```

```
[ ] HMV = ['Bus','Truck','Van']
    LMV = ['Car','Motorcycle','Bicycle']
    TRAIN_DIR = 'train/train/'
    for i in HMV:
      print('{} : {}'.format(len(os.listdir(TRAIN_DIR+i)),i))
    for i in LMV:
      print('{} : {}'.format(len(os.listdir(TRAIN_DIR+i)),i))
```

```
hmv_data = np.zeros((1,224,224,3))
lmv_data = np.zeros((1,224,224,3))
for a in HMV:
  hmv_data = np.append(hmv_data,[img_to_array(load_img(TRAIN_DIR+a+'/'+i,target_size = (224,224))) for i in os.listdir(TRAIN_DIR+a)[:100]],axis = 0)
for a in LMV:
  lmv_data = np.append(lmv_data,[img_to_array(load_img(TRAIN_DIR+a+'/'+i,target_size = (224,224))) for i in os.listdir(TRAIN_DIR+a)[:100]],axis = 0)
hmv_data = hmv_data[1:]
lmv_data = lmv_data[1:]
```

```
[ ] print(len(hmv_data))
    print(len(lmv_data))
```

Copy_of_Vehicle_Classification.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

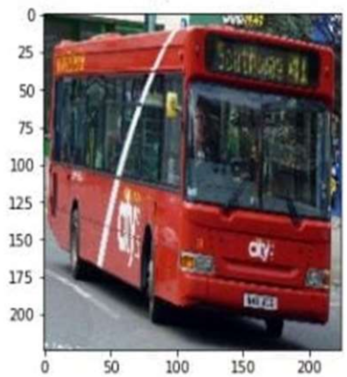
+ Code + Text Copy to Drive Connect Editing

```
print(len(hmv_data))
print(len(lmv_data))
img_data = np.append(hmv_data,lmv_data,axis = 0)
print(img_data.shape)
```

```
300
300
(600, 224, 224, 3)
```

```
plt.imshow(img_data[0].astype(int))
```

```
<matplotlib.image.AxesImage at 0x7f2a4d294358>
```



```
num_classes = 2
```

Copy_of_Vehicle_Classification.ipynb

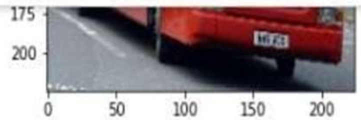
File Edit View Insert Runtime Tools Help Changes will not be saved

Share

\+ Code + Text Copy to Drive

Connect Editing

[ ]



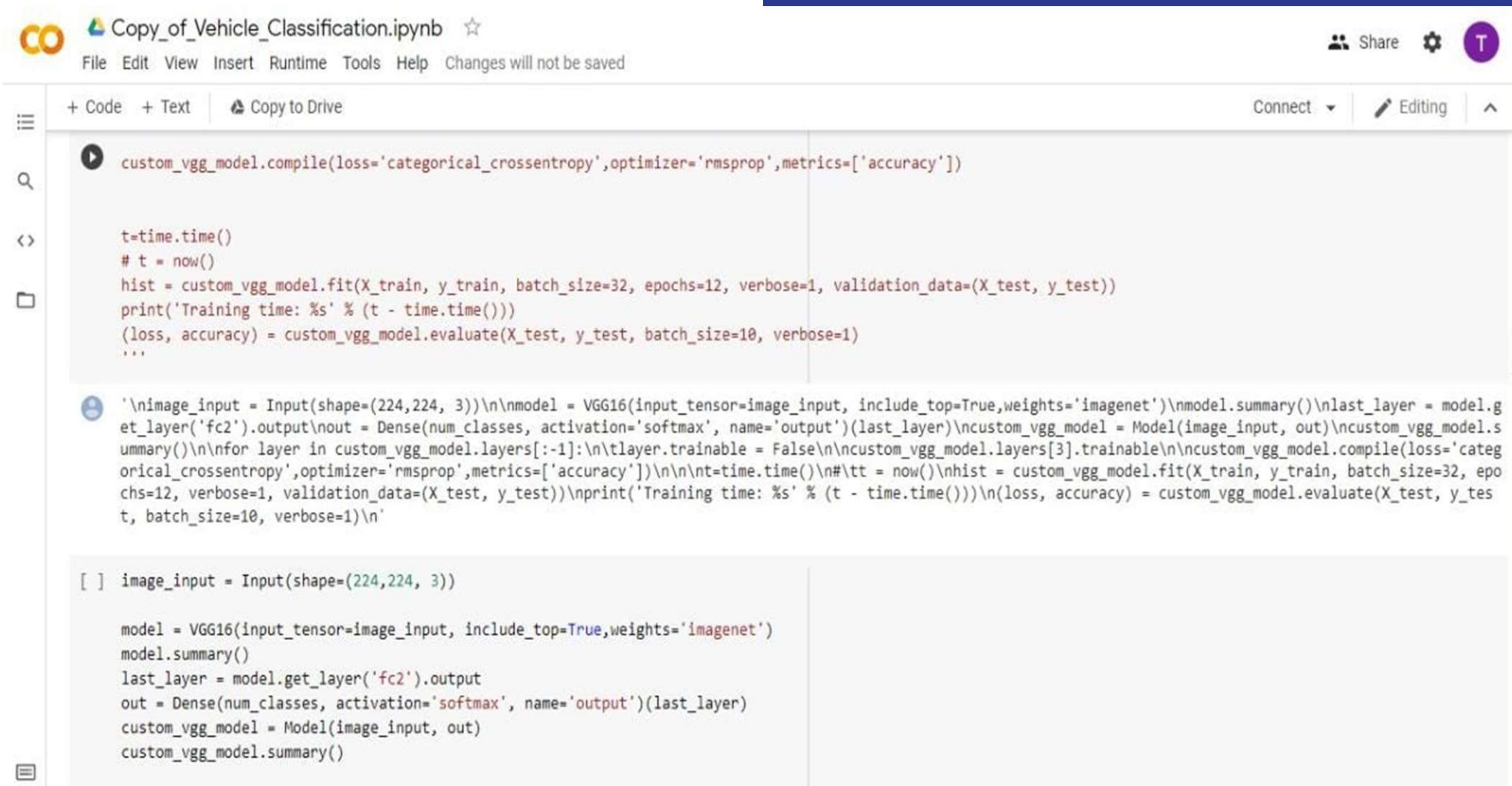

```
num_classes = 2
num_of_samples = img_data.shape[0]
labels = np.ones((num_of_samples,),dtype='int64')

labels[0:300]=0
labels[300:]=1
names = ['HMV','LMV']

# convert class labels to on-hot encoding
Y = np_utils.to_categorical(labels, num_classes)

#Shuffle the dataset
#x,y = shuffle(img_data,Y, random_state=2)
# Split the dataset
#x = img_data
#y = Y
X_train, X_test, y_train, y_test = train_test_split(img_data, Y, test_size=0.2, random_state=2)
'''
image_input = Input(shape=(224,224, 3))

model = VGG16(input_tensor=image_input, include_top=True,weights='imagenet')
model.summary()
```


Copy_of_Vehicle_Classification.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text Copy to Drive
Connect
Editing
custom_vgg_model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])
t=time.time()
# t = now()
hist = custom_vgg_model.fit(X_train, y_train, batch_size=32, epochs=12, verbose=1, validation_data=(X_test, y_test))
print('Training time: %s' % (t - time.time()))
(loss, accuracy) = custom_vgg_model.evaluate(X_test, y_test, batch_size=10, verbose=1)
'''
'\nimage_input = Input(shape=(224,224, 3))\n\nmodel = VGG16(input_tensor=image_input, include_top=True,weights='imagenet')\nmodel.summary()\nlast_layer = model.get_layer('fc2').output\nout = Dense(num_classes, activation='softmax', name='output')(last_layer)\ncustom_vgg_model = Model(image_input, out)\ncustom_vgg_model.summary()\n\nfor layer in custom_vgg_model.layers[:-1]:\n\tlayer.trainable = False\n\ncustom_vgg_model.layers[3].trainable\n\ncustom_vgg_model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])\n\n\nt=time.time()\n#\tt = now()\nhist = custom_vgg_model.fit(X_train, y_train, batch_size=32, epochs=12, verbose=1, validation_data=(X_test, y_test))\nprint('Training time: %s' % (t - time.time()))\n(loss, accuracy) = custom_vgg_model.evaluate(X_test, y_test, batch_size=10, verbose=1)\n'
[ ] image_input = Input(shape=(224,224, 3))
model = VGG16(input_tensor=image_input, include_top=True,weights='imagenet')
model.summary()
last_layer = model.get_layer('fc2').output
out = Dense(num_classes, activation='softmax', name='output')(last_layer)
custom_vgg_model = Model(image_input, out)
custom_vgg_model.summary()

Copy_of_Vehicle_Classification.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

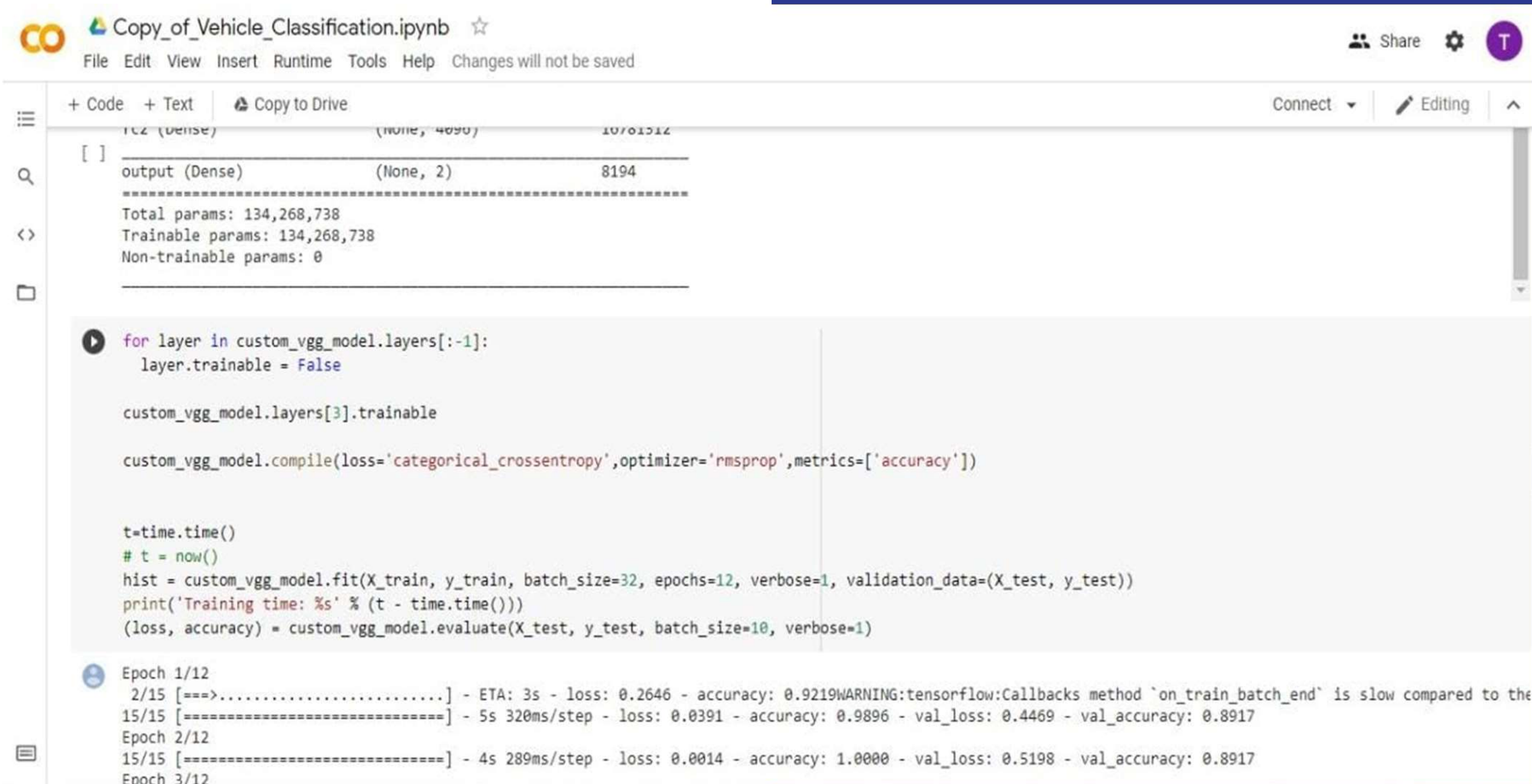
Share

+ Code + Text Copy to Drive

Connect Editing

```
model.summary()
last_layer = model.get_layer('fc2').output
out = Dense(num_classes, activation='softmax', name='output')(last_layer)
custom_vgg_model = Model(image_input, out)
custom_vgg_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
```




```
output (Dense)               (None, 2)                 8194
=================================================================
Total params: 134,268,738
Trainable params: 134,268,738
Non-trainable params: 0
_________________________________________________________________
```

```
for layer in custom_vgg_model.layers[:-1]:
  layer.trainable = False

custom_vgg_model.layers[3].trainable

custom_vgg_model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])


t=time.time()
# t = now()
hist = custom_vgg_model.fit(X_train, y_train, batch_size=32, epochs=12, verbose=1, validation_data=(X_test, y_test))
print('Training time: %s' % (t - time.time()))
(loss, accuracy) = custom_vgg_model.evaluate(X_test, y_test, batch_size=10, verbose=1)
```

```
Epoch 1/12
 2/15 [===>..........................] - ETA: 3s - loss: 0.2646 - accuracy: 0.9219WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the
15/15 [==============================] - 5s 320ms/step - loss: 0.0391 - accuracy: 0.9896 - val_loss: 0.4469 - val_accuracy: 0.8917
Epoch 2/12
15/15 [==============================] - 4s 289ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.5198 - val_accuracy: 0.8917
Epoch 3/12
```

QUICK TEST RESULTS

```
Epoch 1/12
 2/15 [===>..........................] - ETA: 3s - loss: 0.2646 - accuracy: 0.9219WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the
15/15 [==============================] - 5s 320ms/step - loss: 0.0391 - accuracy: 0.9896 - val_loss: 0.4469 - val_accuracy: 0.8917
Epoch 2/12
15/15 [==============================] - 4s 289ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.5198 - val_accuracy: 0.8917
Epoch 3/12
15/15 [==============================] - 4s 290ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.4633 - val_accuracy: 0.9000
Epoch 4/12
15/15 [==============================] - 4s 291ms/step - loss: 8.1215e-04 - accuracy: 1.0000 - val_loss: 0.5404 - val_accuracy: 0.8833
Epoch 5/12
15/15 [==============================] - 4s 290ms/step - loss: 5.6910e-04 - accuracy: 1.0000 - val_loss: 0.5221 - val_accuracy: 0.8750
Epoch 6/12
15/15 [==============================] - 4s 291ms/step - loss: 0.0034 - accuracy: 0.9979 - val_loss: 0.4631 - val_accuracy: 0.9000
Epoch 7/12
15/15 [==============================] - 4s 291ms/step - loss: 3.4545e-04 - accuracy: 1.0000 - val_loss: 0.5274 - val_accuracy: 0.8917
Epoch 8/12
15/15 [==============================] - 4s 292ms/step - loss: 3.4029e-04 - accuracy: 1.0000 - val_loss: 1.0202 - val_accuracy: 0.8583
Epoch 9/12
15/15 [==============================] - 4s 293ms/step - loss: 0.0052 - accuracy: 1.0000 - val_loss: 0.5609 - val_accuracy: 0.9000
Epoch 10/12
15/15 [==============================] - 4s 293ms/step - loss: 1.6886e-04 - accuracy: 1.0000 - val_loss: 0.5314 - val_accuracy: 0.9000
Epoch 11/12
15/15 [==============================] - 4s 293ms/step - loss: 1.7099e-04 - accuracy: 1.0000 - val_loss: 0.5044 - val_accuracy: 0.8917
Epoch 12/12
15/15 [==============================] - 4s 294ms/step - loss: 1.0655e-04 - accuracy: 1.0000 - val_loss: 0.5969 - val_accuracy: 0.9000
Training time: -54.0170681476593
 2/12 [====>.........................] - ETA: 1s - loss: 0.1463 - accuracy: 0.9500WARNING:tensorflow:Callbacks method `on_test_batch_end` is slow compared to the
12/12 [==============================] - 1s 116ms/step - loss: 0.5969 - accuracy: 0.9000
```

This shows the accuracy of the model which is certainly 0.9

Copy_of_Vehicle_Classification.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

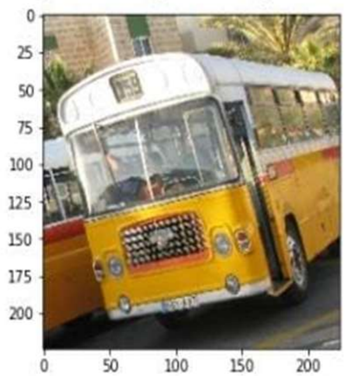
Share

+ Code + Text Copy to Drive Connect Editing

```
test_img = img_to_array(load_img('/content/test/testset/000130.jpg',target_size = (224,224)))
test_img_feed = test_img.reshape(1,test_img.shape[0],test_img.shape[1],test_img.shape[2])
preds = custom_vgg_model.predict(test_img_feed)
#print(preds[0])

if preds[0][0]>preds[0][1]:
  print('Heavy Motor Vehicle')
else:
  print('Light Motor Vehicle')
plt.imshow(test_img.astype(int))
```

```
Heavy Motor Vehicle
<matplotlib.image.AxesImage at 0x7f2a4b03c5f8>
```



Copy_of_Vehicle_Classification.ipynb

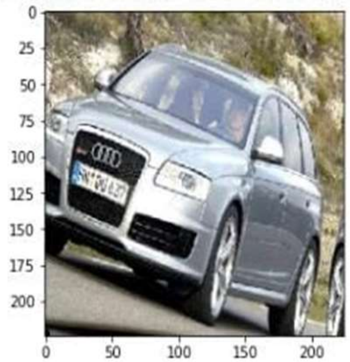
File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

Share Connect Editing

```
  print('Light Motor Vehicle')
plt.imshow(test_img.astype(int))
```

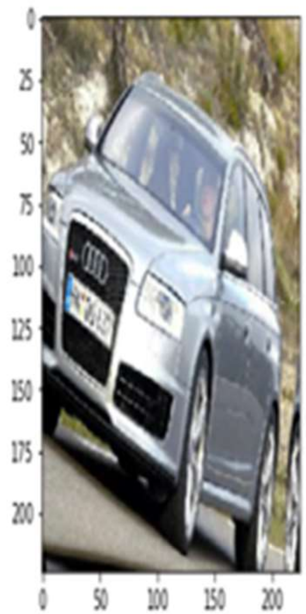
```
Light Motor Vehicle
<matplotlib.image.AxesImage at 0x7f2a4b0d1f28>
```




```
test_img = img_to_array(load_img('/content/test/testset/000130.jpg',target_size = (224,224)))
test_img_feed = test_img.reshape(1,test_img.shape[0],test_img.shape[1],test_img.shape[2])
preds = custom_vgg_model.predict(test_img_feed)
#print(preds[0])

if preds[0][0]>preds[0][1]:
```

```
Light Motor Vehicle
<matplotlib.image.AxesImage at 0x7f2a4b0d1f28>
```



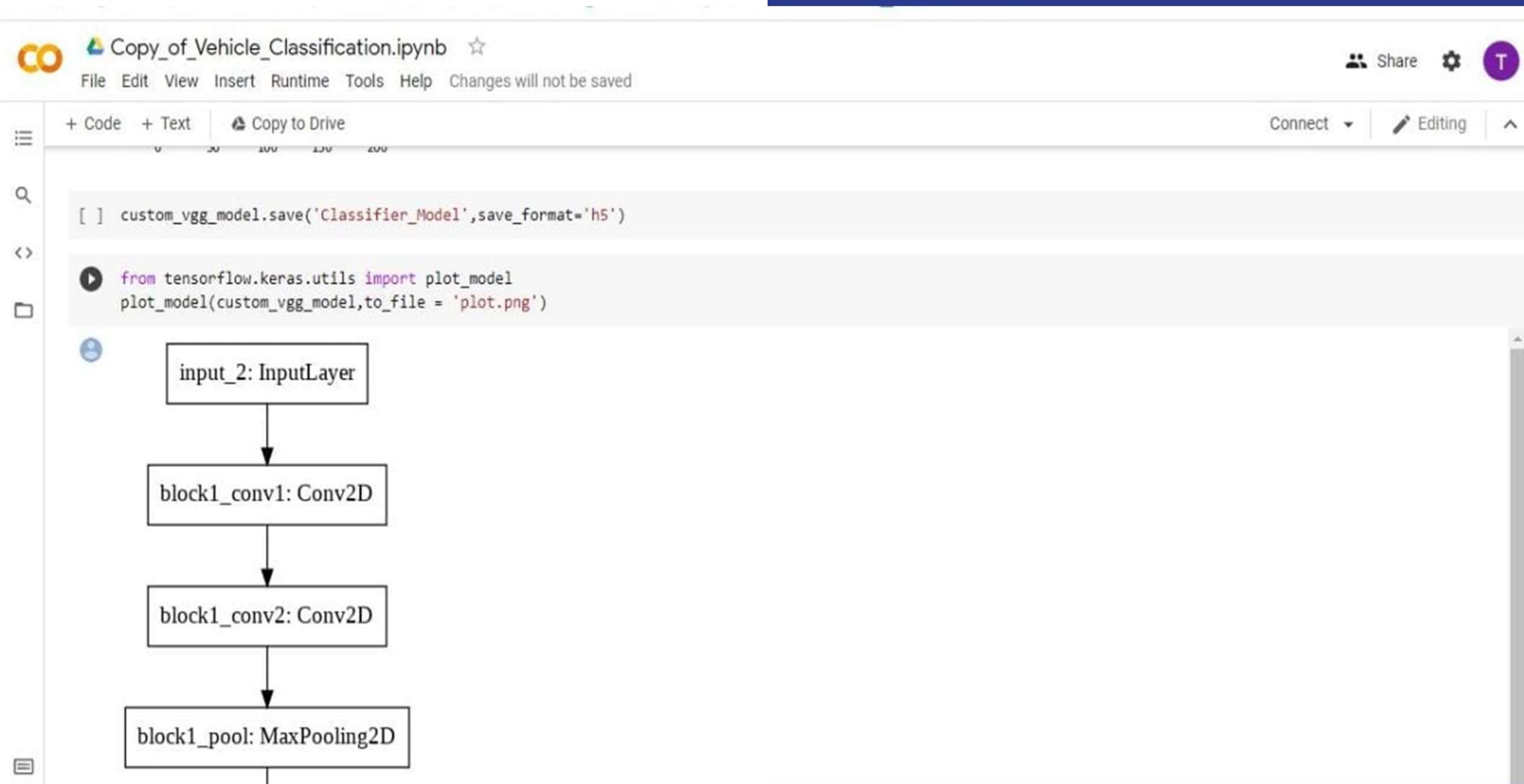
```
Heavy Motor Vehicle
<matplotlib.image.AxesImage at 0x7f2a4b03c5f8>
```

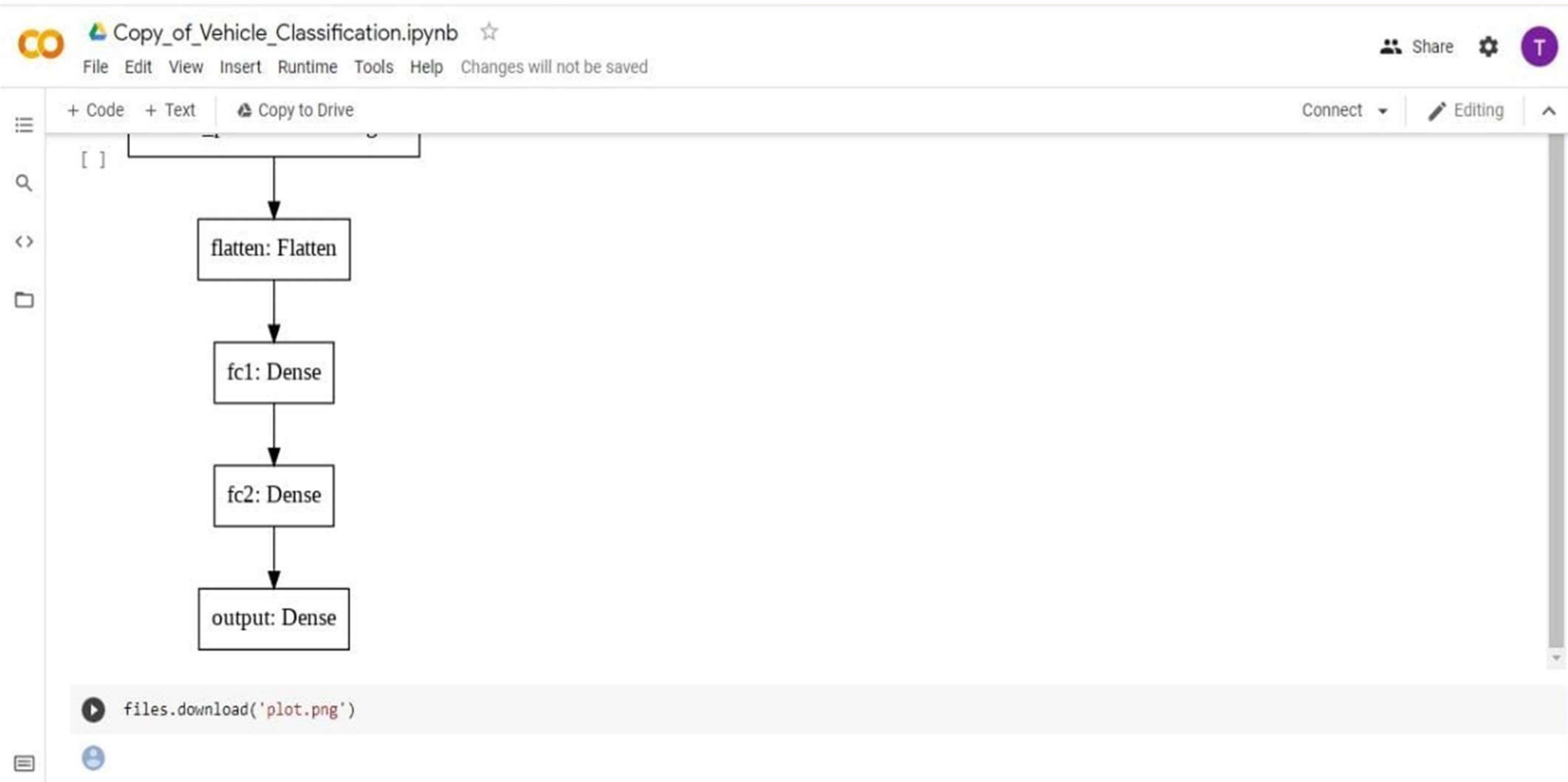
```
[ ] test_img = img_to_array(load_img('/content/test/testset/000130.jpg',target_size = (224,224)))
    test_img_feed = test_img.reshape(1,test_img.shape[0],test_img.shape[1],test_img.shape[2])
    preds = custom_vgg_model.predict(test_img_feed)
    #print(preds[0])
```

```
[ ] custom_vgg_model.save('Classifier_Model',save_format='h5')
```

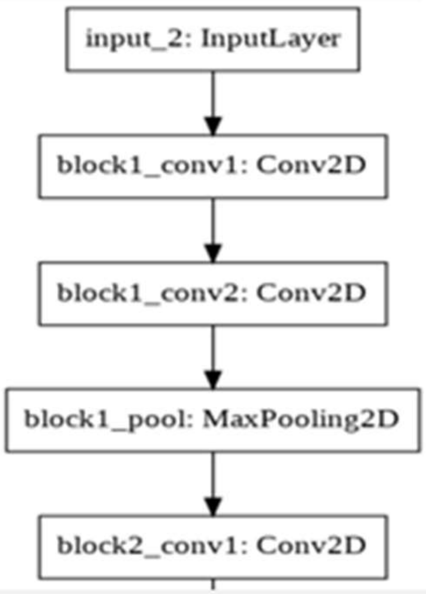
Screenshot of the Training Images column.

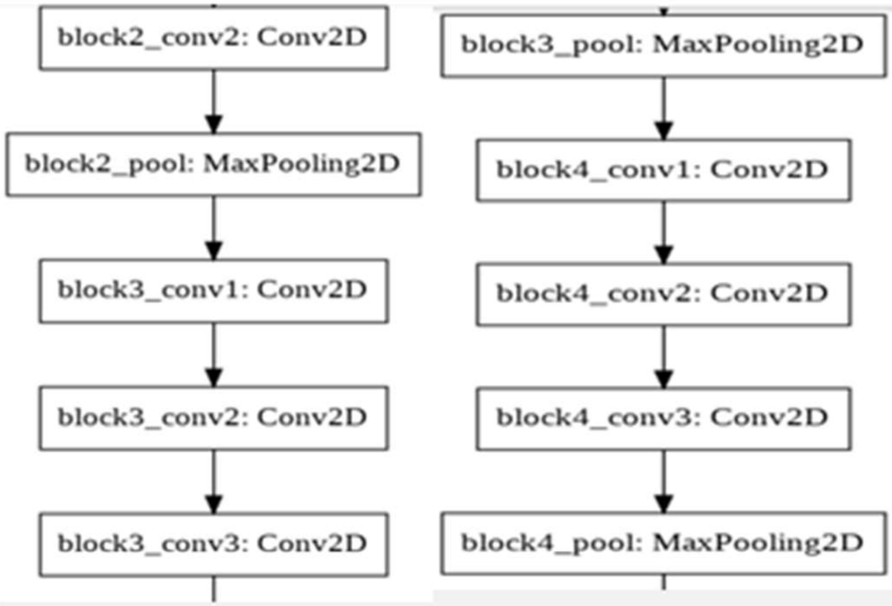
Virtual Exchange Program
@Asia University

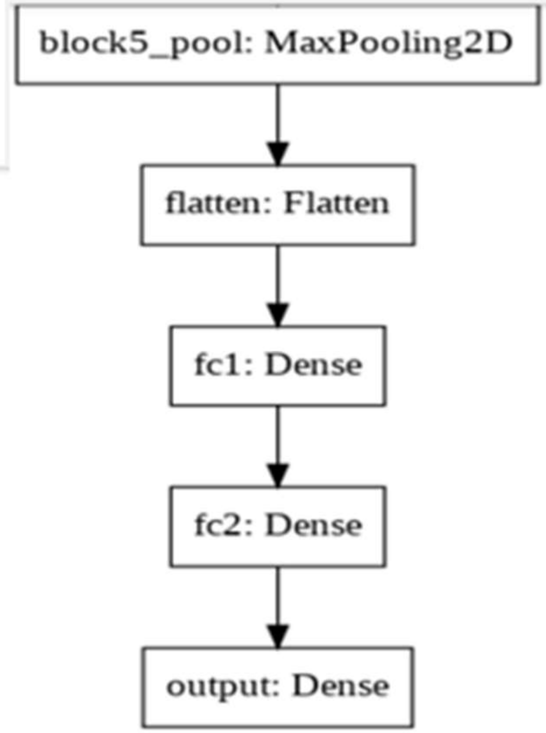

Copy_of_Vehicle_Classification.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
T
+ Code + Text
Copy to Drive
Connect
Editing
[ ] custom_vgg_model.save('Classifier_Model',save_format='h5')
from tensorflow.keras.utils import plot_model
plot_model(custom_vgg_model,to_file = 'plot.png')
input_2: InputLayer
block1_conv1: Conv2D
block1_conv2: Conv2D
block1_pool: MaxPooling2D


Copy_of_Vehicle_Classification.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Share
+ Code + Text
Copy to Drive
Connect
Editing
flatten: Flatten
fc1: Dense
fc2: Dense
output: Dense
files.download('plot.png')

EVALUATION







The proposed approach senses particles by constructing a convolutionary neural network from the foundation. The first level extracts the edges from the raw image, the second level extracts the shapes from the edge details, and so on. The First Level and Second Level Convolutionary Layers function map is shown in Figure . Samples obtained for training and ground truth bounding boxes are seen in Figure for the vehicle. The system presented is checked for another picture that is not in the database and the estimation of a bounding car is seen.

OBSERVATION

We have created a fully innovative neural convolutionary network that is simple but accurate and effective. In the object recognition framework, the convolutionary features collected from our system are stronger than the state-of-the-art image classification network. Our system achieves precision by swapping versatility features with a quicker RCNN, both during preparation and during evaluation. But our model did not recognize the noise when the picture was being recorded. In the future, noise will be considered as a pre-processing stage. The proposed model worked well without noise, making reliable forecasts of some of the test images.


Thank You