Detecting Vehicle Using Webcam

```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\badda\
anaconda3\lib\site-packages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in c:\users\badda\
anaconda3\lib\site-packages (from opencv-python) (1.20.3)
Note: you may need to restart the kernel to use updated packages.
```

```
import cv2

stream = cv2.VideoCapture('traffic_footage.avi')
vehicle_cascade = cv2.CascadeClassifier('vehicle_classifier.xml')


while True:

    ret, frames = stream.read()
    gray = cv2.cvtColor(frames, cv2.COLOR_BGR2GRAY)


    cars = vehicle_cascade.detectMultiScale(gray, 1.1, 1)

    for (x,y,w,h) in cars:
        cv2.rectangle(frames,(x,y),(x+w,y+h),(0,0,255),2)

    cv2.imshow('video2', frames)

    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):    # Press q to break out
        break

# cleanup
stream.release()
cv2.waitKey(1)
cv2.destroyAllWindows()
cv2.waitKey(1)
```

```
-1
```

Vehicle detection using CNN(Deep Learning Model)

```
# Deep Learning CNN model to recognize vehicle
'''This script uses a database of images and creates CNN model on top
of it to test
   if the given image is recognized correctly or not'''

'''####### IMAGE PRE-PROCESSING for TRAINING and TESTING data
```

```
#######'''

# Specifying the folder where images are present
TrainingImagePath='C:/Users/badda/Downloads/Vehicle Images/Final
Training Images'

from keras.preprocessing.image import ImageDataGenerator


# Defining pre-processing transformations on raw images of training
data
# These hyper parameters helps to generate slightly twisted versions
# of the original image, which leads to a better model, since it
learns
# on the good and bad mix of images
train_datagen = ImageDataGenerator(
        shear_range=0.1,
        zoom_range=0.1,
        horizontal_flip=True)

# Defining pre-processing transformations on raw images of testing
data
# No transformations are done on the testing images
test_datagen = ImageDataGenerator()

# Generating the Training Data
training_set = train_datagen.flow_from_directory(
        TrainingImagePath,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')


# Generating the Testing Data
test_set = test_datagen.flow_from_directory(
        TrainingImagePath,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')

# Printing class labels for each vehicle
test_set.class_indices
```

```
Found 27997 images belonging to 16 classes.
Found 27997 images belonging to 16 classes.
```

```
{'Ambulance': 0,
 'Barge': 1,
 'Bicycle': 2,
 'Boat': 3,
```

```
 'Bus': 4,
 'Car': 5,
 'Cart': 6,
 'Helicopter': 7,
 'Limousine': 8,
 'Motorcycle': 9,
 'Segway': 10,
 'Snowmobile': 11,
 'Tank': 12,
 'Taxi': 13,
 'Truck': 14,
 'Van': 15}

'''############ Creating lookup table for all vehicles ############'''
# class_indices have the numeric tag for each vehicles
TrainClasses=training_set.class_indices

# Storing the vehicles and the numeric tag for future reference
ResultMap={}
for faceValue,faceName in
zip(TrainClasses.values(),TrainClasses.keys()):
    ResultMap[faceValue]=faceName

# Saving the vehicles map for future reference
import pickle
with open("ResultsMap.pkl", 'wb') as fileWriteStream:
    pickle.dump(ResultMap, fileWriteStream)

# The model will give answer as a numeric tag
# This mapping will help to get the corresponding vehicles name for it
print("Mapping of vehicles and its ID",ResultMap)

# The number of neurons for the output layer is equal to the number of
vehicles
OutputNeurons=len(ResultMap)
print('\n The Number of output neurons: ', OutputNeurons)

Mapping of vehicles and its ID {0: 'Ambulance', 1: 'Barge', 2:
'Bicycle', 3: 'Boat', 4: 'Bus', 5: 'Car', 6: 'Cart', 7: 'Helicopter',
8: 'Limousine', 9: 'Motorcycle', 10: 'Segway', 11: 'Snowmobile', 12:
'Tank', 13: 'Taxi', 14: 'Truck', 15: 'Van'}

 The Number of output neurons:  16

'''######################### Create CNN deep learning model
#########################'''
from tensorflow.keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPool2D
from keras.layers import Flatten
```

```
from keras.layers import Dense

'''Initializing the Convolutional Neural Network'''
classifier= Sequential()

''' STEP--1 Convolution
# Adding the first layer of CNN
# we are using the format (64,64,3) because we are using TensorFlow
backend
# It means 3 matrix of size (64X64) pixels representing Red, Green and
Blue components of pixels
'''
classifier.add(Convolution2D(32, kernel_size=(5, 5), strides=(1, 1),
input_shape=(64,64,3), activation='relu'))

'''# STEP--2 MAX Pooling'''
classifier.add(MaxPool2D(pool_size=(2,2)))

'''############## ADDITIONAL LAYER of CONVOLUTION for better accuracy
#################'''
classifier.add(Convolution2D(64, kernel_size=(5, 5), strides=(1, 1),
activation='relu'))

classifier.add(MaxPool2D(pool_size=(2,2)))

'''# STEP--3 FLattening'''
classifier.add(Flatten())

'''# STEP--4 Fully Connected Neural Network'''
classifier.add(Dense(64, activation='relu'))

classifier.add(Dense(OutputNeurons, activation='softmax'))

'''# Compiling the CNN'''
#classifier.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
classifier.compile(loss='categorical_crossentropy', optimizer =
'adam', metrics=["accuracy"])

############################################################
import time
# Measuring the time taken by the model to train
StartTime=time.time()

# Starting the model training
classifier.fit_generator(
                    training_set,
                    steps_per_epoch=7,
                    epochs=10,
```

```
                    validation_data=test_set,
                    validation_steps=10)

EndTime=time.time()
print("###### Total Time Taken: ", round((EndTime-StartTime)/60),
'Minutes ######')
```

```
C:\Users\badda\AppData\Local\Temp/ipykernel_100884/3785646586.py:44:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.
  classifier.fit_generator(

Epoch 1/10
7/7 [==============================] - 14s 2s/step - loss: 28.1250 -
accuracy: 0.2054 - val_loss: 2.7369 - val_accuracy: 0.1969
Epoch 2/10
7/7 [==============================] - 13s 2s/step - loss: 2.4877 -
accuracy: 0.2500 - val_loss: 2.4000 - val_accuracy: 0.2969
Epoch 3/10
7/7 [==============================] - 12s 2s/step - loss: 2.4862 -
accuracy: 0.2946 - val_loss: 2.4479 - val_accuracy: 0.2969
Epoch 4/10
7/7 [==============================] - 13s 2s/step - loss: 2.5161 -
accuracy: 0.2455 - val_loss: 2.4918 - val_accuracy: 0.2594
Epoch 5/10
7/7 [==============================] - 13s 2s/step - loss: 2.5067 -
accuracy: 0.2991 - val_loss: 2.4396 - val_accuracy: 0.2937
Epoch 6/10
7/7 [==============================] - 12s 2s/step - loss: 2.3581 -
accuracy: 0.3393 - val_loss: 2.4279 - val_accuracy: 0.3031
Epoch 7/10
7/7 [==============================] - 13s 2s/step - loss: 2.3428 -
accuracy: 0.2723 - val_loss: 2.4063 - val_accuracy: 0.3219
Epoch 8/10
7/7 [==============================] - 12s 2s/step - loss: 2.4549 -
accuracy: 0.3527 - val_loss: 2.8199 - val_accuracy: 0.3125
Epoch 9/10
7/7 [==============================] - 13s 2s/step - loss: 2.5807 -
accuracy: 0.2768 - val_loss: 2.5001 - val_accuracy: 0.3125
Epoch 10/10
7/7 [==============================] - 13s 2s/step - loss: 2.5525 -
accuracy: 0.3125 - val_loss: 2.4654 - val_accuracy: 0.3438
###### Total Time Taken:  2 Minutes ######
```

```
'''########### Making single predictions ###########'''
import numpy as np
from tensorflow.keras.preprocessing import image

ImagePath='C:/Users/badda/Downloads/Vehicle Images/Final Testing
Images/Boat/test.jpg'
```

```
test_image=image.load_img(ImagePath,target_size=(64, 64))
test_image=image.img_to_array(test_image)

test_image=np.expand_dims(test_image,axis=0)

result=classifier.predict(test_image,verbose=0)
#print(training_set.class_indices)

print('####'*10)
print('Prediction is: ',ResultMap[np.argmax(result)])

########################################
Prediction is:  Boat
```