Gebze Technical University
Department of Computer Engineering
CSE 665 (Introduction) to Computer Vision
HW3 Report

H.Yavuz ERZURUMLU
181041004

June 15, 2018

Abstract

In this paper, I will tell you how to count the number of vehicles, identify their types of cars (trucks, cars, buses, minivans) for the marked lane via background subtraction and deep learning techniques.

You can access the whole source which used in this report from https://github.com/freeloki/CSE665-ComputerVisionHW3

1 Introduction

Nowadays deep learning tools are very popular and powerful. We will use two different deep learning network for detecting type of vehicles on **"M6 Motorway Traffic"** video. You can access full video from here.

We will use first 10 minutes of video for generating training data. Rest of video (20 minute) is for testing.

2 Tools

Here is the list of tools which I have used in this project.

- https://en.savefrom.net/ for downloading video from YouTube.
- Kdenlive for splitting video to train and test videos.
- Kazam for screen recording.
- opencv for image processing.
- tensorflow framework for training and testing.
- GitHub for version control.
- Overleaf latex editor for create this report.

3 Data Preparation

Data is the most important part of any deep learning application. For this project, we have 10 minutes of M6 highway stream.

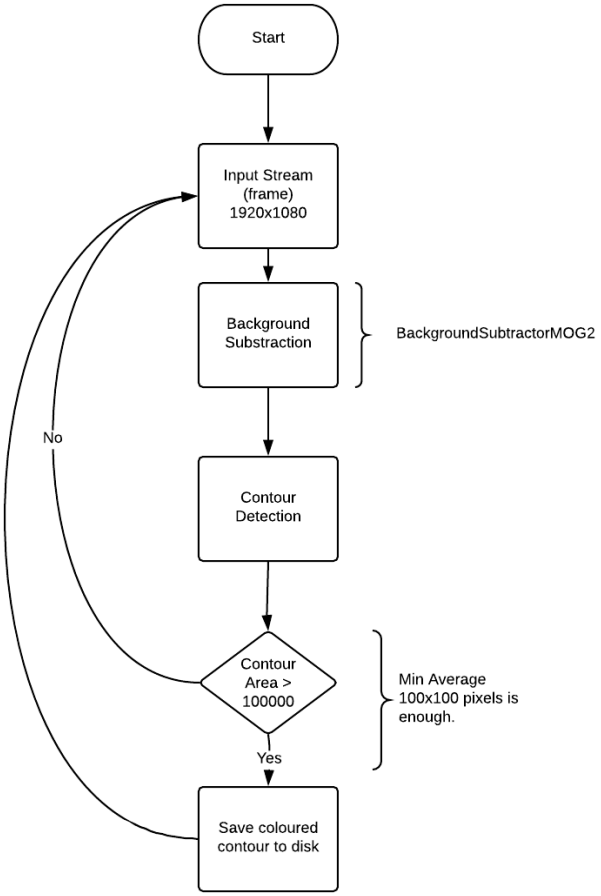


Figure 1: Data extraction from video frames.

After the process finished I've 44.984 sample data including noisy datas.
Definition of noise for this dataset:

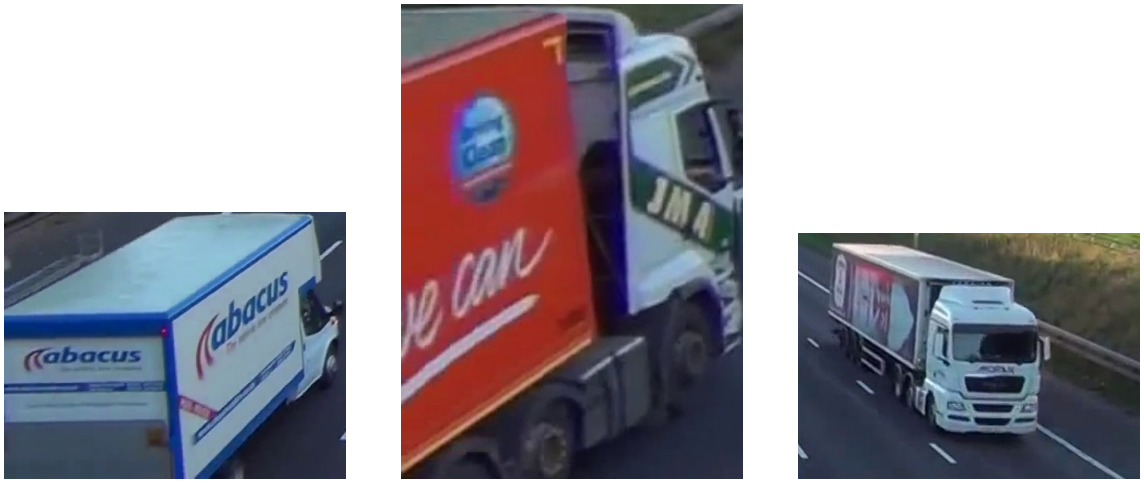
- Multiple car in same frame.
- No car in frame.
- Any frame is not able to classify by human eyes.

I didn't use all of these data. Because I don't want to mess with 44k images :). I have grouped 1715 data into four category.

- m6automobile (1018)
- m6minivan (270)
- m6truck (238)
- m6bus (149)

(a) Sample valid automobile data

(b) Sample valid minivan data



(c) Sample valid truck data

(d) Sample valid bus data

Figure 2: Some grouped images

4 Training

4.1 Models

I've used two CNN models for training.

- GoogLeNet Inception v3
- MobileNet v2

4.1.1 GoogLeNeT - Inception v3

Inception-v3 is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012.

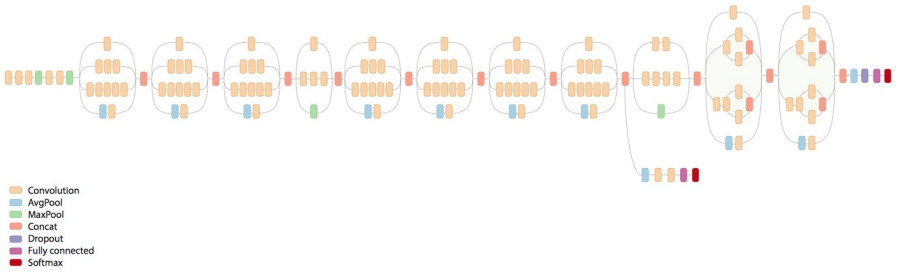


Figure 3: GoogLeNet Inception v3 Model.

Detailed info about this model is here.
Training parameters:

- Training Data %80
- Validation Data %10
- Test Data %10
- Learning Rate 0.01
- Training Step 4000

These values are default values for Tensorflow's training script.

4.1.2 MobileNet v2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 4: Summary of MobileNet v2.

Training parameters:

- Training Data %80
- Validation Data %10
- Test Data %10
- Learning Rate 0.01
- Training Step 4000

5 Test

For testing, I've drawn a handmade rectangle in test video. Every 30 frame checked for available contours. Then send this test image to convolutional network. After parsing response. The guessed response will adding into car counter.

You can watch real time tests from

- Right Road Incoming Traffic Inception v3
- Left Road Outcoming Traffic MobileNet v2
- Left Road Outcoming Traffic Inception v3

6 Results

Here is result of some training variations. I've used the default ones on real time video.

Model	**DataSize**	**LearnRate**	**TrainStep**	**TrainRate**	**ValidRate**	**Test Rate**
Inceptionv3(Default)	1715	0 01	4000	%100	%100	%100
Inceptionv3	1715	0 01	500	%98	%98	%97 9
Inceptionv3	1715	0 001	500	%97 3	%98	%96 2
Inceptionv3	1715	0 01	1000	%100	%98	%98 9
MobileNetv2(Default)	1715	0 01	4000	%100	%100	%100
MobileNetv2	1715	0 01	500	%100	%100	%99 4
MobileNetv2	1715	0 001	100	%94	%99	%96?
MobileNetv2	1715	0 001	4000	%100	%100	%100

Table 1: Training variations

7 Conclusion

I used two different CNN networks to train my custom car data. Both of the networks are have very successful results.

Learning rate and training step are having a huge role for training. Sometimes test success ratio can be bigger than training success ratio. I couldn't figure it out why.

On real time test sometimes the same car can be count as twice. That's because we are not tracking it. we are just sampling the contour area for every 30 fps. If we track the object then we can solve the duplicate counting problem. We used min 10k pixel contours if signed are is smaller then this the algorithm cannot detect properly.

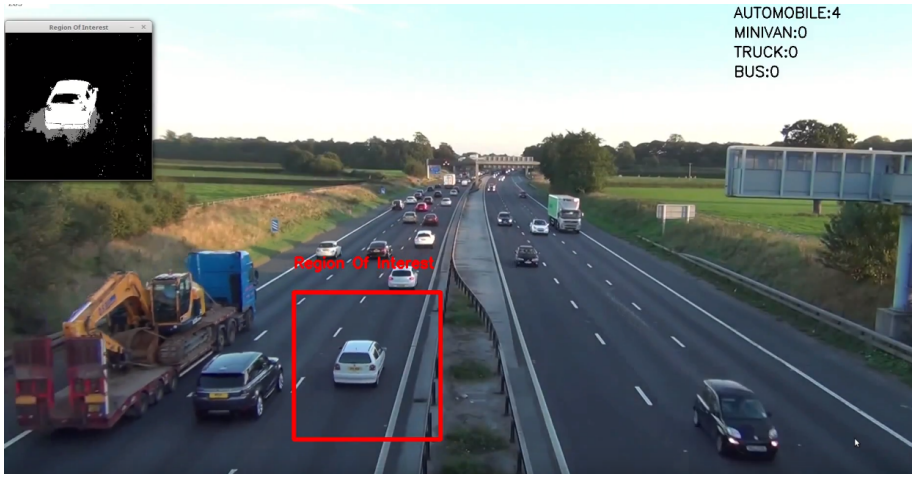


Figure 5: Sample Test 1

Figure 6: Sample Test 2

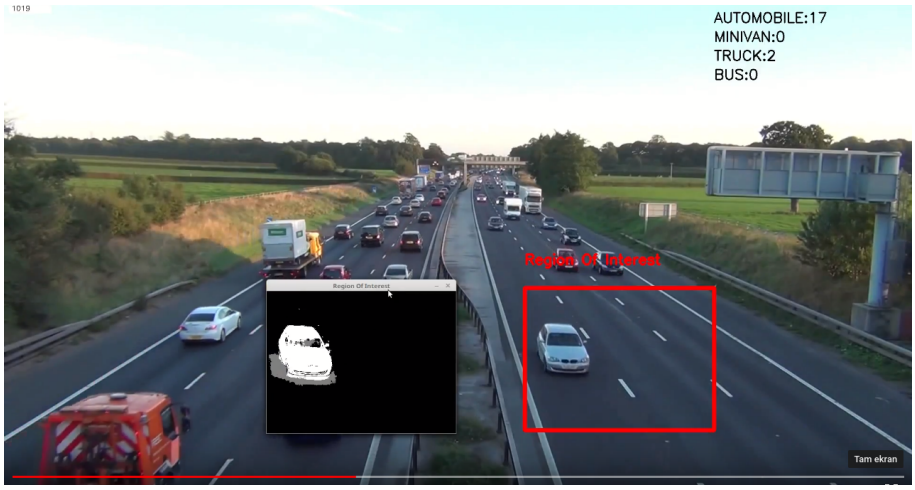


Figure 7: Sample Test 3