ECSE-415 Introduction to Computer Vision
Final Project: Vehicle Counting

Due: Friday December 10, 2021, 11:59PM

1 Introduction

The goal of this project is to create algorithms for extremely overlapping vehicle counting in traffic congestion situations. We will mainly take two different approaches for vehicle counting (1) Counting using detection (2) Direct vehicle count regression from images. The dataset used in this project is TRaffic ANd COngestionS (TRANCOS) dataset [1]. It consists of 1244 images, with a total of 46796 vehicles. Each image is of size 640x480 pixels. TRANCOS dataset is divided into three subsets: train (623 images), valid (200 images), and test (421 images). CSV files for all three subsets are given with the project [1]. The CSV files provide the name of the images and the corresponding ground-truth vehicle count for that image. You should use training and validation data to create your final model, and then use this final model to present your predictions on test data. This predictions should be uploaded to class's Kaggle competition web page. Note that your team is entitled to two submissions per day. you can learn more about how to submit your results here. Note that you will need to create an account on the Kaggle site. Each group member should make an account, and then merge with the other members to form a single team. Submissions should only be done by a single team rather by each group member.

Students should read the TRANCOS paper to learn about different approaches to vehicle counting. An example of the images in this dataset is shown in the figure below.

For this project, you will form a team of 3 or 4 people. The objective of the project is to permit your team to explore different approaches for vehicle counting.

Projects should be submitted electronically via myCourses. All submitted material is expected to be the students' own work or appropriately cited (See Academic Integrity guidelines in the Syllabus). Software packages (e.g. scikit-learn, MATLAB toolboxes, Keras, PyTorch, etc.) which implement machine

[1]Check Appendix-A for python code snippet to read the CSV file.

learning-related algorithms (SVM, RF, k-fold cross-validation, etc.) are allowed. Appropriate citations are expected. The use of external code without citation will be treated as plagiarism. Projects received after the deadline up to 3 days late will be penalized by 10% per day. Projects received after Monday December 13 will not be graded. The project will be graded out of a total 100 points.

2 Submission Instructions

1. Submit (i) report in pdf format, (ii) all code (together in a single zip file)
2. Comment your code appropriately.
3. Make sure that the submitted code is running without error. Add a `README` file if required.
4. The report should be comprehensive but concise.
5. If external libraries were used in your code, please specify them by name and version in the `requirement.txt` file.
6. Submissions that do not follow the format will be penalized 10%.

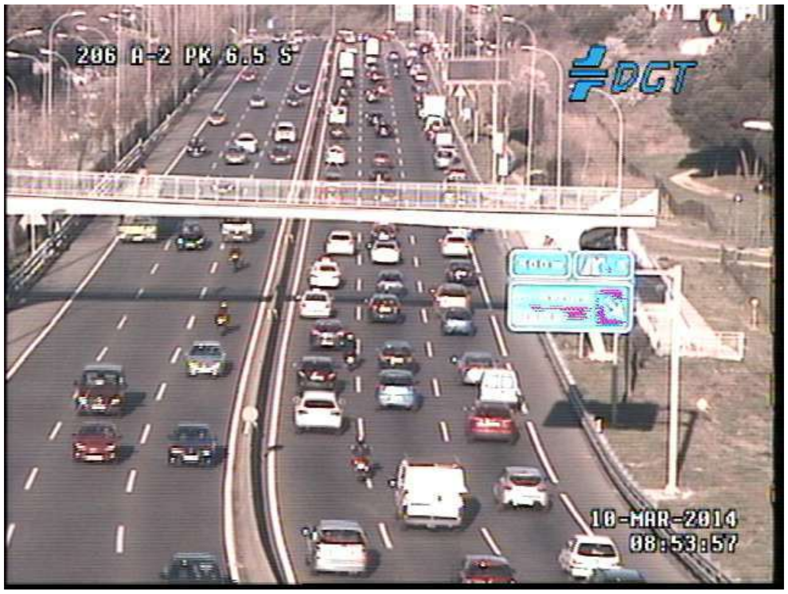

Figure 1: TRANCOS dataset example

3 Vehicle Counting

There are three parts to the project:

1. Choose an existing vehicle detection technique, such as YOLO, Faster-RCNN, where code is already available (e.g. you can find useful code on the Facebook Detectron2 site, Vehicle Detection with YOLO). Use this to obtain bounding boxes around vehicle in the traffic images. Then, write python code to eliminate duplicate detections (if any) and to count the bounding boxes.

2. Choose any new deep convolutional neural network (CNN) architecture (e.g., ResNet [2], VGG [3], etc.) with a single neuron output layer, and use an L2 loss (ridge regression) for training the CNN on the training set. You can use validation set to tune the hyperparameters of the network. Note that you can either train the network from the scratch or you can use a pre-trained network and fine-tune it on the TRANCOS dataset.

3. Implement your own vehicle detection algorithm, based on a Support Vector Machine or Random Forest classifier (such as that used in Assignment 3). For this part of the project, you need to implement **only one** of the following two approaches.

   (a) Get the positive examples by cropping out image patches containing vehicles from the TRANCOS dataset. Negative examples can be obtained by extracting randomly located patches from the images (that do not intersect vehicle patches). You can use the vehicle detector implemented in part 1 to identify the vehicle patches in each image. From these patches you should compute suitable features to feed into the SVM or the RF. Typically the raw intensity values will not work too well. You could try HoG or Haar features. You could also try Local Binary Pattern (LBP) features. These were not covered in class, but are fairly simple and fast. Train your SVM or RF using the extracted positive and negative examples. It is up to you to determine how many examples you need to do the training. Once you have the SVM-based vehicle detector working, you can use it in place of the pre-made detector in part 1, to make a new version of the complete vehicle counting system.

   (b) Train a Support Vector Regressor or Random Forest Regressor using appropriate features to directly predict the number of vehicles in an image. Typically the raw intensity values will not work too well. You could try HoG or Haar features. You could also try Local Binary Pattern (LBP) features. These were not covered in class, but are fairly simple and fast. You can refer to the original TRANCOS dataset paper for choosing appropriate features.

To evaluate the vehicle counting, we will use Mean Deviation Error (MDE) Metric. MDE is calculated as a weighted Mean Absolute Error, where weights are defined as 1/real_vehicle_count for each image. Calculate these metrics for each image in the validation dataset for all three approaches above. Calculate the same for the test dataset using the Kaggle competition.

In addition to this (for all three approaches), also display the scatter plot of true vehicle count (X-axis) vs the Predicted Vehicle Count (Y-Axis). Report the pearson correlation coefficient on the validation set. Write your observations.

At last, upload your best results on vehicle counts in test data to Kaggle. Please refer to Kaggle competition web page to understand how your submission should look like.

The final report should include the performance of the vehicle counting as expressed above. The grade for the project will be based on the following items:

- Demo - the grader will check and run the provided code to verify that it (more-or-less) works properly.
- Report - the grader will look for the required report elements: description of algorithms used (with reference to code), reasoning behind the design choices (used features), evaluation (using MSE, MAE, Pearson Correlation metrics and the scatter plot), discussion of issues arising from the design (e.g. difficulties encountered, effectiveness of different types of features, etc). Mention your Kaggle team at the starting of the report.

The project must be done in groups of 3 or 4 students. Each student must do a significant part of the project, and the role of each student should be stated in the report. All group members will receive the same grade.

Appendix-A

```python
import pandas as pd

# path to csv file
# change this to your local path to mapping.csv
csv_path = "/home/raghav/Documents/ECSE_415/ECSE_415_F_2021/Project
    /test_count.csv"

# read csv file
df = pd.read_csv(csv_path)

# convert dataframe into two separate lists

# images - list -- contains name of images. Ex. ["image-1-000001.
    jpg", "image-1-000002.jpg", "image-1-000003.jpg", "image
    -2-000001.jpg", ...]
images = list(df["images"])

# counts - list -- contains vehicle counts for a particular images.
     Ex. [40, 39, 36, 33, ...]
```

```python
# Note how each image has an associated counts.
# For Ex. "image-1-000001.jpg" has an associated count 40.
# This represents that image-1-000001.jpg has total 40 vehicles in
    the image.
# We will use these counts for counting purpose.
counts = list(df["counts"])

# just for verification purpose
print(images)
print(counts)
```

Listing 1: python code for reading a csv file

References

[1] Ricardo Guerrero-Gómez-Olmedo et al. “Extremely Overlapping Vehicle Counting”. In: *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*. 2015.

[2] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[3] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).