Group 5

Bernardo Barcellos de Castro Cunha	**9293380**
Eduardo Santos Carlos de Souza	**9293481**
William Quelho Ferreira	**9293421**

Pre-Trained CNN applied to driving-related object recognition

Main Objective:

Given an image taken facing the direction in which a car is moving, the system shall detect road signs, cars, pedestrians and other relevant objects for a driver or autonomous system using a pre-trained convolutional neural network (CNN). The system would ideally work in real-time and could possibly be able to detect the position of the objects within the image.

Input Images:

The images of the our dataset will be collected by two different ways. We will research for datasets that other people have already built and manually select from them images that can be useful for our project. Also we will use an automated crawler of our own to look for relevant images on Google Images. Some of the datasets that we will use include:

- Stanford Dogs (http://vision.stanford.edu/aditya86/ImageNetDogs/)
- GRAZ_02 (http://www-old.emt.tugraz.at/~pinz/data/GRAZ_02/)
- PascalVOC (http://host.robots.ox.ac.uk/pascal/VOC/databases.html)
- STL-10 (https://cs.stanford.edu/~acoates/stl10/)

All images used in our datasets will be available at http://bit.ly/2spOvbj.

The images will be rescaled to 64x64, and some variations will be applied for the training images, with transformations such as translation, rotation and/or horizontal mirroring.

Figure 1: Example of a dataset image collected from Google Images

Figure 2: Image after processing

The dataset’s images will belong to one of the following classes:

1. Bicycle
2. Car
3. Motorcycles
4. Pedestrian
5. Traffic Light
6. Dog
7. No parking sign
8. Stop Sign
9. Toll
10. Truck

Detailed description:

The idea of our project was to adapt a pre-trained CNN to recognize objects of classes specific for use in a driving scenario, such as in an autonomous vehicle.

We used the VGG16 model trained on the ImageNet dataset as our pre-trained CNN. We downloaded the trained CNN without the input and top layers, then froze the parameters for the convolutional layers, and re-generated the top layers and input layers, and retrained them on our specific dataset.

After acquiring the images from the aforementioned datasets or Google Images, we manually filtered out the unwanted ones, and divided them into different directories, one for each class. After that, we randomly selected 20% of the images in each class and moved them into a different directory, meant to test the CNN after training was complete over the remaining 80%. For the remaining 80% we applied the size reduction and random transformations mentioned above on 10 copies of each image. We did this in order to increase the variety of our training data and reduce the training time.

With both the modeled CNN and the dataset in hands, we did the following steps six times, two times training for 30 epochs without fine tuning, once training for 100 epochs without fine tuning, two times training for 30 epochs with fine tuning and once training for 100 epochs with fine tuning:

1. Generate a new CNN based on the model we used
2. Train the CNN on the training dataset generated as described above, with batch sizes of 128 images
3. Test the accuracy of the CNN on the test dataset*

Results:

We evaluated accuracy by taking a testing set, predicting the class for each image, assuming that the identified class for the image is the position of the maximum value of the output of the CNN, and comparing the predicted class with the correct one. The results are as follows:

Description	Epoch training time	Training set accuracy	Test set accuracy
30 epochs, no fine tuning (1st iteration)	17s	0.9520	0.7419
30 epochs, no fine tuning (2nd iteration)	17s	0.9499	0.7218
100 epochs, no fine tuning	17s	0.9825	0.7535
30 epochs, fine tuned (1st iteration)	51s	0.9967	0.8249
30 epochs, fine tuned (2nd iteration)	51s	0.9963	0.8479
100 epochs, fine tuned	51s	0.9977	0.8456

Source code and images:

The source code for the project can be found at https://github.com/wqferr/PPdI. The demo program is the “camera.py” script.

The datasets used for training and testing can be found at http://bit.ly/2spOvbj.