Universidad Rey Juan Carlos

SUPERIOR TECHNICAL SCHOOL OF TELECOMMUNICATION ENGINEERING

Media studies and
Audiovisual Systems engineering

Bachelor's Degree Final Project

Vehicle Detection using Deep Learning

Author: David Pascual Hernández
Tutors: José María Cañas Plaza, Inmaculada Mora Jiménez

Academic year 2016/2017

Contents

List of Figures

List of Tables

Chapter 1

Framework

This chapter serves as a way to introduce the tools that I have employed during the development of this project. All of them are **open-source**. The transparency provided by the open-source platforms is a major advantage because the software can be joined together and adapted to our specific application, which is mainly written in **Python** [1].

JdeRobot

JdeRobot [2] is an open source middleware for robotics and computer vision. It has been designed to simplify the software development in this fields. It's mostly written in C++ language and it's structured like a collection of components (tools and drivers) that communicate to each other through **ICE interfaces** [3]. It is also compatible with **ROS** [4], which allows us to use ROS nodes and JdeRobot components simultaneously. This flexibility makes it very useful for our application. We're going to employ its *cameraserver* driver to capture images from different video sources.

cameraserver

According to JdeRobot documentation, this driver can serve both real cameras and video files. It interacts with other components thanks to the *Camera* interface.

[1]`https://www.python.org/`
[2]`http://jderobot.org`
[3]`https://zeroc.com/products/ice`
[4]`http://www.ros.org/`

In order to use *cameraserver*, we have to properly set its configuration file. These are the parameters that we must specify:

- A network address where the server is going to be listening.
- Parameters related with the video stream: URI, framerate, image size, format.

Keras

As stated by **Keras** documentation [1]: ”Keras is a high-level neural network library, written in Python and capable of running on top of either TensorFlow or Theano”. TensorFlow and Theano are open source libraries for numerical computation optimized for GPU and CPU. Currently, I'm running Keras on top of **Theano** [5] optimized for CPU.

Every neural network in Keras is defined as a *model.* For those neural networks which can be built as a stack of layers, Keras provides the ***Sequential model*** object. It is also possible to build more complex models with multiple outputs and shared layers using the Keras *functional API.*

The process to implement a neural network with Keras is very intuitive. We just have to declare a *Sequential model*, add the desired layers to that model and train it. Besides that, we can preprocess the input data and monitorize what is happening while the model is being trained with several built-in functions. It's also possible to build those functions on our own and integrate them into the Keras workflow.

Keras library provides a function to read **HDF5** [6] datasets that can be used as inputs to the neural networks. It also saves the *models* structure and weights into HDF5 files.

Scikit-Learn

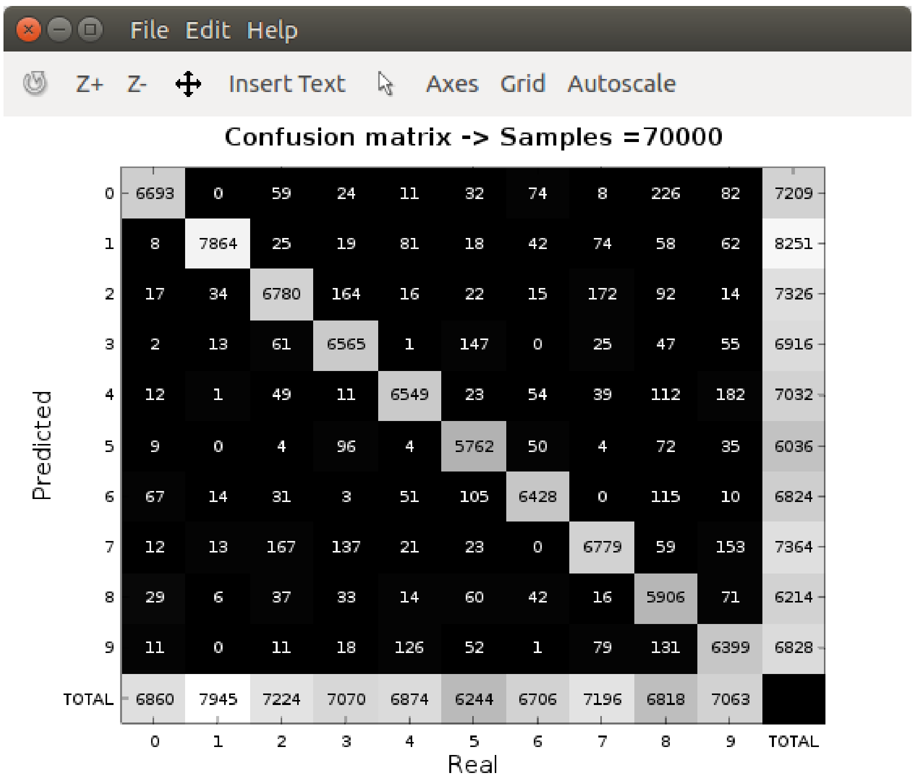
Scikit-Learn [7] is a machine learning library that includes a wide variety of algorithms for clustering, regression and classification. It can be used during the whole machine learning process: preprocessing, training, model selection and evaluation.

We're going to employ Scikit-Learn functions to evaluate the neural networks that we have developed with Keras. Using a tool that is independent from Keras allows us to

[5] `http://deeplearning.net/software/theano/index.html`

[6] `https://support.hdfgroup.org/HDF5/doc/H5.format.html`

[7] `http://scikit-learn.org/stable/index.html`


File Edit Help
Z+ Z- Insert Text Axes Grid Autoscale
Confusion matrix -> Samples =70000
Predicted
Real

	0	1	2	3	4	5	6	7	8	9	TOTAL
0	6693	0	59	24	11	32	74	8	226	82	7209
1	8	7864	25	19	81	18	42	74	58	62	8251
2	17	34	6780	164	16	22	15	172	92	14	7326
3	2	13	61	6565	1	147	0	25	47	55	6916
4	12	1	49	11	6549	23	54	39	112	182	7032
5	9	0	4	96	4	5762	50	4	72	35	6036
6	67	14	31	3	51	105	6428	0	115	10	6824
7	12	13	167	137	21	23	0	6779	59	153	7364
8	29	6	37	33	14	60	42	16	5906	71	6214
9	11	0	11	18	126	52	1	79	131	6399	6828
TOTAL	6860	7945	7224	7070	6874	6244	6706	7196	6818	7063	

Figure 1.1: Example of a confusion matrix visualization using Octave.

compare the performance of different neural network libraries (e.g. Keras and Caffe).

Octave

GNU Octave [8] is a scientific programming language compatible with Matlab. It provides powerful tools for plotting. We're going to use these tools to visualize the data collected with Scikit-Learn about the performance of our models. An example of a visualization using Octave can be seen in the figure 1.1.

[8] https://www.gnu.org/software/octave/

Bibliography

[1] F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.