1. Introduction (Motivation, Problem, Related works)

In the modern times the tendency towards automation of day-to-day processes is growing fast and with the exponential development of technology is certainly influencing it. The discovery of a new branch in mathematics and informatics – Machine Learning and especially the most recent Deep Learning Algorithms created the possibility to handle and automatize more complex problems, such as driving a car. Self-driving cars are one of the main technologies, which are about to hit the markets. In this project we set ourselves the goal to reproduce an efficient self-driving model, using Deep Learning architectures. Of course, we do not have an opportunity to create and test a model based on a real car. However, we found a suitable alternative – The Open Racing Car Simulator (TORCS), an open-source game which allows to incorporate driving algorithms and simulate them.

1. Related Works

Self-Driving Cars are currently developed by Google, as well as most of the major car companies. There is also some work related to driving simulation in TORCS, the main one being that of Yan Panlau.

1. Architecture

Our goal was to create an architecture which makes decisions only based on images - the view of the car’s driver (not using other sensors provided in most cars). We decided to build different models and try as many different methods as we can in order not only to find the best one, but also to evaluate the possibilities of the field. The input of the architectures is solely the image, and the output a number in the range (-1, 1) – the interpretation of the angle of car’s movement. Next, the 3 different architectures that we implemented during this project will be described. The accomplished results of these will be stated later.

Arch. 1: Vanilla(Plain) Convolutional Neural Network

The network consists of 5 (size varied) Convolutional Layers with various kernel sizes (mostly 5 or 3) and strides (mostly 2 and 1). Between the layers the activation function ReLU as well as Batch Normalization were used. We avoided using Fully-Connected (Linear) layers in this case, so the output of the last convolutional layer is of size 1x1x1.

Arch. 2: Convolutional Neural Network based on a pre-trained Network (VGG16)

This network uses the VGG16 pre-trained network with frozen Convolutional layers as the base of the architecture and adds to them only the last Fully-Connected (Linear) layers. Their number and sizes also varied in the process of development (mostly 4 layers).

Arch. 3: Deep Deterministic Policy Network

This network is based on the work of Yan Panlau and was adapted towards our modified problem – using only the image as input (the original work used the sensors provided by the game). This architecture is based on Reinforcement Learning and consists of an Actor Network, which is a policy function (a mapping from input (image) to output (angle – action)) and a Critic Network which gives positive or negative rewards based on how good of an action was generated by the Actor. Both networks consist of a number of Linear Layers (2-3) with various layer sizes. The reward is based on the car’s velocity and its proximity to the center of the road.