Projectverslag – ML model toepassen op een (zelfrijdende) autosimulatie in SimPyLC

Op donderdag 22 september vond de kickoff plaats van dit project. In de 3,5 week hiervoor hebben wij kennisgemaakt met programmeren, en het gebruik van vele begrippen en tools die wij nodig hebben om dit project te kunnen uitvoeren. Sommige zaken zijn uitgebreid voorbijgekomen (Machine Learning, loss functies, OOP, versiebeheer, etc.), andere tools hebben we slechts even aangeraakt (NumPy, Tensorflow, SciKitLearn) en zullen we tijdens het uitvoeren van ons project beter leren kennen. Voor ons project dienden wij voor zowel de ‘lidar’ besturing als de ‘sonar’ besturing een trainingsmodel te bouwen, eerst met behulp van de library Tensorflow/keras, en daarna met behulp van SciKitLearn.

Ik ben begonnen met observeren van de hardcoded simulatie: wat gebeurt er met de stuurhoek, de snelheid, etcetera. Een interessante bevinding was dat de stuurhoek uitsluitend een waarde van -22, 0, of 22 heeft. Tevens heb ik gespeeld met het aanpassen van de snelheid, en bemerkt dat boven een snelheid van 7 m/s het autootje van het circuit af vliegt. Hierna ben ik overgegaan op het verzamelen van trainingsdata uit het hardcoded model op zowel de lidar track als de sonar track. Hiervoor heb ik het hardcoded model ongeveer 5 rondjes op beide tracks laten rijden. Tevens heb ik validatie data verzameld door 1 rondje te rijden op beide tracks, waardoor ik voor validatie ongeveer 20% van de hoeveelheid trainingsdata heb verzameld. Vervolgens ben ik in Jupyter een (lidar & sonar) Tensorflow trainingsmodel gaan bouwen. De Tensorflow website was hierbij het belangrijkste hulpmiddel. Bij het draaien van de trainingsmodellen kwam ik erachter dat de ingevoerde csv files met data uitmaakten op het resultaat, en ook het wijzigen van het aantal hidden layers en het aantal nodes per layer en het aanpassen van de learning rate uitmaakten op het resultaat wat mijn model gaf. Het normaliseren van de data alvorens dit door het model te halen gaf per definitie een beter resultaat (lagere loss, hogere accuracy) dus heb ik besloten om normalisatie standaard op mijn model toe te passen. Wat betreft het aantal layers en nodes en het aantal epochs heb ik aantallen gekozen die een redelijk gunstig resultaat gaven, en besloot ik om pas later, na implementatie van het model in de Driving Agent, te bekijken of dit nog verder geoptimaliseerd kan worden.

Bij het implementeren van mijn trainingmodel in de drivingAgent file ben ik (samen met mijn medestudenten) de hardcoded uitvoerig gaan bestuderen. Daarbij na het regel voor regel doorlopen van de code tot de conclusie gekomen dat er iets moet gebeuren in de methodes ‘input’ en ‘sweep’.