Projectverslag project 1

Kevin van der Meer

Applied Artificial Intelligence – cohort 1 2022/2023

1064998

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.

Het proces

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 training model 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'. Na trial, error, en veel onderling overleg met medestudenten ben ik tot het

inzicht gekomen dat bij de input het model ingeladen moet worden, en bij sonarweep en lidarweep het gros van de instructies om de stuurhoek te bepalen verwijderd diende te worden, en vervangen moest worden door de voorspelde stuurhoekwaarden uit de modellen (bij een gegeven sonar/lidar afstand). En het werkt! Voor sonar althans.

De presentatie besloten wij op te splitsen in drie delen, zodat wij allen een stuk hadden om te presenteren. Mijn stukje was een overgangsbrug tussen de implementatie van het getrainde model in de code, en ethiek. Inhoudelijk presenteerde ik onze conclusie over het project, en maakte ik een start met ethiek. Na de drie gegeven presentaties had ik uiteindelijk een voldaan gevoel, en het idee dat we de werkgevers ook daadwerkelijk iets te vertellen hadden.

Obstakels en leerpunten

Binnen de korte periode dat we met het project bezig waren viel ik onverhoopt twee dagen uit wegens ziekte. Ik kwam erachter dat ik binnen die twee dagen enorm achterliep op de rest, en moest een enorme inhaalslag maken in de avonden en in het weekend. Daarnaast heb ik behoorlijke problemen gehad met de installatie van diverse programma's en libraries op mijn laptop, Tensorflow voorop. Ook werkten de shell scipts om de world, de hardcoded client en de driving agent te openen niet, waardoor ik deze python files met een omweg diende te openen. Het grootste leerpunt kwam ik echter tegen toen ik begon aan dit verslag. Ik zag bij diverse medestudenten een goed bijgehouden research documentje, waaruit aantekeningen, grafieken etcetera gekopieerd konden worden. Zelf had ik uitsluitend aantekeningen met pen en papier bijgehouden en de onleesbare brij was eigenlijk nutteloos. Voor het volgende project ben ik voornemens om vanaf de start van het project een researchdocumentje bij te houden, zodat ik aan het einde van het project een wat uitgebreider en inhoudelijk treffender verslag kan inleveren dan ik nu kon doen.

Welke scanningsmethode is het meest geschikt en waarom?

De scanningsmethodes die we in dit project hebben vergeleken zijn Lidar en Sonar. Sonar scant de omgeving af op obstakels door middel van geluidsgolven, waar lidar gebruik maakt van lichtgolven. Kort gezegd komt het erop neer dat lidar werkt met de snelheid van het licht, waar sonar werkt op de snelheid van geluid. Lidar is dus vele malen sneller dan sonar. In onze simulatie heeft de sonar 3 scanners, en de lidar 16 scanners, die beiden een kijkhoek vooruit van 120 graden (graad voor graad) afgaan. De lidar is hierdoor veel sneller en accurater dan de sonar, en is dus per definitie het meest geschikt om een gesimuleerde auto te trainen om zelf te rijden over een circuit. In mijn model werkt sonar echter beter dan lidar (lidar heb ik zelfs niet aan de praat gekregen), en dat heeft te maken met de hogere complexiteit van de input (16 inputs in plaats van 3), maar ook met het circuit. Op het gegeven lidar circuit is er (linksonderaan) een punt waar de auto te weinig input krijgt doordat de auto zich meer dan 20 meter van de dichtstbijzijnde pion bevindt. Dit zou op te lossen moeten zijn door meer obstakels/pionnen toe te voegen aan het circuit of de simulatie zo in te stellen dat er verder vooruit gekeken wordt dan 20 meter. Als we uitgebreid de tijd

zouden nemen om het circuit en/of de instellingen van de simulatie te optimaliseren, zou de lidar scanningsmethode het uiteindelijk beter doen dan de sonar.

Welke aansturingsmethode werkt het beste en waarom?

In mijn geval werkt de TensorFlow aansturing wel, en de SciKit Learn aansturing niet. Ik heb het idee dat Tensorflow een uitgebreidere library is, waardoor ik denk dat Tensorflow uiteindelijk het beste werkt.

Hoe is veiligheid gegarandeerd in alle gevallen?

De veiligheid is allereerst gegarandeerd omdat SimPyLC een simulatieomgeving is, waarin middels een header duidelijk is aangegeven dat het software is welke niet ontworpen/ bestemd is voor het ontwerpen van ‘real world’ toepassingen. Daarnaast is de hardcoded aangedreven door instructies, maar is er te weinig input bestaat de kans dat het autootje van de baan of tegen een pion op rijdt. Ook in de driving agent zijn er geen veiligheidsgaranties en kunnen obstakels geraakt worden, afhankelijk van hoe goed het model getraind is.

Welke aansturingsmethode is het veiligst en waarom?

Bij een voldoende getraind model (waarbij er altijd voldoende input vanuit de sensoren komt tijdens het trainen) is uiteindelijk de driving agent veiliger dan de hardcoded. Voordat we op dat punt zijn zouden we echter nog vele aanpassingen moeten doen aan de circuits, en het model verder moeten finetunen.

Is de opzet van de codebase modulair? Hoe aan te tonen?

De opzet van mijn codebase is zoveel als mogelijk modulair. Wel is de code geschreven in Python, dus alleen modulair wanneer python gebruikt wordt. Als ik eraan toegekomen zou zijn om naast een Tensorflow trainingsmodel ook een goed werkend SciKit Learn model te bouwen zou ik dit met enkele aanpassingen en het importeren van andere libraries in de python file kunnen implementeren.