Project 2: Haven (transport)

Projectverslag Daniël Schene

Vooraf

Dit verslag is relatief kort aangezien verreweg de meeste tijd van dit project is gaan zitten in het bouwen, trainen en evalueren van het model. In de code zelf staan weinig opmerkingen over de motivatie achter bepaalde keuzes; daarom zal ik deze hier nader proberen toe te lichten. In plaats van per stap te beschrijven hoe de desbetreffende stappen binnen het project zijn genomen/verlopen heb ik er deze keer voor gekozen om per onderdeel binnen de ingeleverde map kort te beschrijven wat welke map en de bestanden daarbinnen inhouden.

Git repository history

De commit historie voor dit project is hier te vinden.

Structuur en mappen

De ingeleverde projectmap heeft de volgende sub-mappen:

-Data
-Modules
-Notebooks
-simpylc

Data

Deze map bevat de door de hardcoded client gegenereerde data voor zowel de sonar als lidar variant van het voertuig.

Modules

Deze map bevat twee bestanden:

> **Models_c.py**: Hierin worden de classes voor de lidar en sonar neural-net modellen gedefinieerd.
>
> **train_model.py**: Dit is het python script waarmee de twee modellen kunnen worden getraind.
>
> In de twee bovenstaande scripts wordt de architectuur van de modellen vastgelegd. Na veel verschillende hoeveelheden hidden layers en hidden layer groottes te hebben

geprobeerd (ik ben helaas niet toegekomen aan het vastleggen van de verschillende resultaten in MLflow) ben ik uitgekomen op, afgaande op het minimaliseren van de loss, 4 hidden layers van respectievelijk 100, 75, 50 en 25 neuronen voor de lidar, en 2 hidden layers van 32 en 16 neuronen voor de sonar variant van de auto. Beide hebben dus een ‘trechtervormige’ architectuur, waarbij het aantal neuronen per laag afneemt richting de output laag.

In de werkelijkheid zou het genereren van de voorspellingen voor de stuurhoek natuurlijk zo snel mogelijk moeten gaan, in real-time. In dat opzicht is het dan een afweging tussen snelheid en precisie, waarbij je het liefst op beide vlakken niet te veel inlevert. Een ‘dichter’ netwerk zal langer doen over het voorspellen van een stuurhoek aan de hand van real-time data dan een minder dicht netwerk. Andersom is het bij een veelvoud aan inputs (16 in dit geval) natuurlijk van groot belang dat het model wel genoeg ruimte heeft om de informatie van de inputs volledig te kunnen gebruiken en om dit om te zetten in een accurate voorspelling. Voor de lidar variant, die werkt met 16 inputs, is het dan ook effectiever gebleken om een eerste hidden layer met een relatief hoog aantal neuronen te kiezen en om dit daarna stapsgewijs af te bouwen richting de output laag. Voor de sonar variant is dit, met maar 3 inputs, misschien minder van belang, maar heb ik alsnog voor twee lagen gekozen met een relatief hoog aantal neuronen (32 en 16).

Notebooks

Deze map bevat een aantal notebooks waarbinnen ik de eerste stappen van dit project heb uitgewerkt. Deze notebooks zijn erg onoverzichtelijk en dienen in dat opzicht enkel als klad-notebooks, om zo een beeld te krijgen van de totstandkoming van de uiteindelijke code.

simpylc

Deze map bevat alle benodigde scripts voor het uitvoeren van de simulatie:

De scripts **train_model.sh** en **test_agent.sh** zijn door mij gemaakte scripts om de modellen te trainen en om de simulatie die van de modellen gebruikmaakt vervolgens te testen. Het trainingsscript dient dan ook vóór het testscript te worden gedraaid.

simulations/control_client/my_agent.py is ‘mijn’ versie van de hardcoded client, aangepast om de auto te laten rijden met de voorspellingen van het getrainde model.

models is de map waarin de getrainde modellen van het **train_model.py** script worden opgeslagen als .pkl bestand, om deze vervolgens te kunnen inladen in **simulations/control_client/my_agent.py**