Project Proposal for Artificial Intelligence & Machine Learning

The goal of this project is to learn new methods for machine learning and use them in praxis.
This project is also available on GitHub: https://github.com/AlexanderWidmann/ML_SelfDrivingCar
This project is done as a single person group by: Alexander Lukas Widmann

Project description

The goal of the project is to create a self-driving car. Since creating an environment is out of scope for this project the open source environment Carla is used to simulate the car in action.
Link to official homepage: http://carla.org/
During this project multiple different models should be implemented and tested. Since the parameters can have a pretty pig impact on the results of the model it is also important to observe the models with different parameters to get their best results.

Motivation

The reason why i chose this specific project is relatively simple. I am interested in the future and possibilities of completely self-sufficient cars and how that could impact our daily lives. With the Udacity car simulator it is possible for me to actually test some algorithms and methods for myself.

Goal

Create an agent that can drive a vehicle in the town autonomously while avoiding collisions and later on also following basic traffic law.
Later iterations should include more fleshed out models that may also be subject to more "rules".
Some example for such rules:

- Staying on the right side of the street
- Managing speed accordingly To keep the project small acceleration and breaking will not be considered for the first implementation. This will also be kept in mind for the data generation. If the models work and the project progresses as planned I will also implement the speed part of the self-driving car. This will also create more interesting reinforcement learning task since the car should go as fast as possible to get the maximum rewards.
  In a similar manner the car should stay inside of the right side of the road and at least avoid crossing double middle lines and should minimize solid line crosses overall. Since this also adds another complexity this should also be added to an already functioning model.

Bonus goals

Carla provides a way to test a fully fleshed out agent that returns a pretty solid evaluation metric for models, as described in their leaderboard:
https://leaderboard.carla.org/
The ultimate goal for the project is to accomplish a model that is at able to finish the task set for that evaluation. The Task:
Following a route with minimal traffic infractions. Those include:

- Running red light/ stoplight
- collisions
- being of track

Data generation

The simulator already provides a tool to generate data for the models to use. To keep the processing time small every produced dataset should be between 1 and 5 minutes of footage.

Technology

List of technologies I plan to use in the project:

- Carla car simulator
- Jupyter Notebook
- Github
- Tensorflow
- Sklearn
- Keras
- Python libraries to enable the ML-Workflow (communication with car simulator, etc..)

Approach

The approach is split into different steps that make the project easy to expand and still keep each step simple and replicable.

Step 1: Proof of concept

This step is not directly part of the project but is rather a step to ensure the wanted outcome can be achieved. In this step I want to show, that it is in fact possible to create a machine learning algorithm that creates a self-driven car for the Carla simulator. To accomplish this step, I will follow a simple tutorial to implement a prototype and use that as a basis for further development.
I decided to use the following tutorial: https://pythonprogramming.net/introduction-self-driving-autonomous-cars-carla-python/

Step 2: First imitation model creation and tests

Using the prototype created with Step 1 I will create the training data and implement a model using imitation learning. For this the most likely algorithm to be used is the Sequential algorithm from Keras. After implementing and training the models the goal is to observe how the agent handles the street. Afterwards it is planned to adjust parameters, add new parameters, and try to tweak the algorithm to observe the effects and find a good implementation of the Sequential Model. This model is a relatively simple model that just maps images of the training data to the given outputs and then correlates similar inputs accordingly.
While this is model will probably struggle in completely new environments it is still worthwhile to observe.
The most interesting part for this model will be how the agent handles the autopiloted vehicles provided with carla, that simulate real traffic.
Important to note: This step is not the focus for the project but is included to generate a baseline agent for the reinforcement learning agent to compare to.

Step 3: Reinforcement learning

In this step I will create a model that uses reinforcement learning to make the car drive autonomously in the town.
This model should be able to start learning without the need for any input data.
An important part of this is how the reward is defined. At the start I will go with a very simple process of rewarding a small amount for distance travelled and subtracting a big amount for collisions. Since Carla considers driving over the sidewalk a collision this will also keep the car on the street. After creating a successful model, it is again important to test if a different reward function can result in a better trained model.
This step will most likely be based on the Markov Decision Process. This is subject to change if a better process is found.

Step 4: Experiment

In this step there is already a trained working model for Step 2 and 3 available. Now the goal is to experiment with the models and tools available to create a hopefully better agent. Some ideas that will be considered in this step:

- Base the reinforcement learning model on the imitation model
  The idea here is, that letting the model train itself solely by itself may result in odd dynamics to maximise the reward. By also using the imitation model the reinforcement model should be forced to act more naturally and not try to exploit reward maximisation as much.

Step 5: Evaluation

This step is not separate from the other steps since every model will be evaluated in some form to decide on how to handle further development. As described in the Goals part the best case scenario is, that the final model for each step is sophisticated enough to be evaluated using the leaderboard evaluation provided by Carla.
If during the project the requirements for this can **not** I will try to find another evaluation tool.
Next to those performance measures the project will include at least the Tensorboard measures for the models.

Literature

Plan for the literature

I plan to use the literature to find new models that can be used to solve the challenges for each step and learn how the parameters affect the outcome for a model. The literature can also show new ways on how to evaluate models and they may be used for this project.

Literature list

Building powerful image classification models using very little data, F. Chollet
https://deeplearning.lipingyang.org/wp-content/uploads/2016/12/Building-powerful-image-classification-models-using-very-little-data.pdf
Self-Driving Car Steering Angle Prediction Based on Image Recognition, S. Du, H. Guo
https://arxiv.org/pdf/1912.05440.pdf
Controlling Steering Angle for Cooperative Self-driving Vehicles utilizing CNN and LSTM-based Deep Networks, R. Valiente, M. Zaman
https://arxiv.org/pdf/1904.04375.pdf
End to end learning for self-driving cars, M. Bojarski, D. Del Testa, et. al.
https://arxiv.org/pdf/1604.07316.pdf
Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars, A. I. Maqueda, A. Loquerio, et. al.
https://openaccess.thecvf.com/content_cvpr_2018/papers/Maqueda_Event-Based_Vision_Meets_CVPR_2018_paper.pdf
CIRL: Controllable Imitative Reinforcement Learning for Vision-based Self-driving, X. Liang, T. Wang
https://openaccess.thecvf.com/content_ECCV_2018/papers/Xiaodan_Liang_CIRL_Controllable_Imitative_ECCV_2018_paper.pdf

Risk management

Simulator based issues

In my first draft for the proposal I planned to use the Udacity car simulator (https://github.com/udacity/self-driving-car-sim). After reviewing that simulator it turned out that the Udacity car simulator only supports imitative learning and does not have an easy way to get negative rewards for a reinforcement learning model. As in the documentation there is no mention of a flag thrown or something similar if the car leaves the track. It would be possible to create such a flag, but that would create its own small project: detect when a car left the road. Since I want to focus on the machine learning part of the self-driving car I searched for a new simulator until I found the Carla simulator.
I also considered the Deepdrive simulator, but Carla had the better documentation: https://github.com/deepdrive/deepdrive
If reinforcement learning turns out to be out of scope or during the project I am not able not create any working models for the Carla simulator the first backup plan is to scale down the project to only an imitation model implemented with the Udacity car simulator.

Project failure/ car agent too complex

If for unknown reason I cannot get a working prototype for most of the steps in a reasonable time or some additional reason is revealed why the project will be unable to be concluded by the deadline (Ex.: Training time is too high) I will go on the following backup project:
Using OpenAi Gym (https://gym.openai.com/) implement a reinforcement learning model for an Atari game. This will again be split into 2 different categories: RAM-based or Image-based. This implementation would start with the RAM-based variant. The goal for the model here would be to score the highest possible amount of points.