Phase 2 – Planning Phase

Training Data

1. Acquisition of Training Data:
   - Adaption of the source code of the OpenAI Car Racing Game to save each frame and the values for steering (-1;1), acceleration (0;1) and braking (0;1).
   - Each frame will be labelled with the corresponding input values in a .csv file, which will be later used for the supervised learning process.
   - The training data will be acquired by manually driving the car around different tracks, which are randomly generated.
   - Since we are driving manually, we will try to slow the game to generate the optimal training data for more efficient training of the neural network.
2. Training Data Handling
   - Automatized script for deleting images to thin out trivial and not relevant frames.
   - Another script will be developed, which counts the amount of different steering/acceleration/brake input signals. The information will be used to generate statistics, which enable the decision-making on deleting certain categories to avoid overfitting.
   - E.g. a category would be that 90% of all recorded images show a straight line and have the input [0;1;0] (only accelerating).
   - Generating two separate datasets. One for the training, and one for the validation of the network.

Architecture/ Model structure

1. Input / Output Layers:
   - Input: PyTorch tensor which is extracted out of the pre-processed image
   - Use image of size 84x96 combined with extracted speed and steering angle of status bar.
   - Output: Output steering vector
     - x1 = steering angle (-1 to 1)
     - x2 = acceleration (0 to 1)
     - x3 = braking (0 to 1)
2. CNN Architecture
   - We choose to implement the DenseNet because it has a very good performance, looking at Benchmarks referring to the data sets ImageNet and Cifar-10.
   - DenseNet is well documented and delivers us enough information resources which facilitates our implementation.
   - The original DenseNet's using ImageNet data sets has a few million parameters. We expect to be able to reduce the number of parameters to match our parameter limit of 300.000.
     - Our research has shown that the used image size for the neural network has been 224x224, since our images are a lot smaller with a size of 96x96 or even smaller considering image pre-processing the number of parameters will decrease and some layers in the network must be adjusted to the different image input size.
     - ImageNet/ Cifar-10 are having 1000 output neurons, we only have 3 so the amount of the parameters will decrease.

- We choose to implement two additional input neurons connected with the FC Layer of the CNN to pass the extracted values of speed and steering to the network.

3. Optimizer
   - SGD and ADAM are the most common optimizer referring to our research.
   - We want to start with ADAM because it delivers quick results and is quite usable.
   - Later we want to focus on the SGD because we suggest a better performance for ResNet and DenseNet architectures from our research. Since the implementation of the optimizer is quite trivial, we want to use both and we generally want to evaluate which optimizer works best for our purpose.

Image processing

- The purpose of image pre-processing is to highlight important features of the pictures and extract relevant data. Moreover, we want to Reduce data without loss of information and read information from graphs in the status bar.
- Input: frames stored in numpy format file
- Output:
  - save processed images in .png format.
  - save processed images in the 2D list of bools in .csv file
  - save all obtained data (speed, abs sensors 1-4, gyroscope, steering, keyboard input) as dictionary into JSON format.
- Process:
  - Count number of horizontal pixels for steering and gyroscope on specific coordinates.
  - Count number of vertical pixels for abs (for each wheel) and current speed on specific coordinates.
  - Cut status bar containing graphs (11 pixels).
  - Apply filter that decides if it is road (1) or grass (0) – binary coding.
  - Produce a 2D list of bools (84x96) - processed frame.

Detailed Time Plan

Due	Task	Subtask
07.05	Training Data	Record Training Data
		Training Data Handling
	Pre-processing	Extract important data from PNG, Resize (No status bar)
	Architecture & Model Structure	Implementation of DenseNet
		Adaption of the structure
21.05	Training of Neural Network	Training Function (1 Day Training Time)
		Data loading PyTorch
28.05	Optimisation	Hyperparameters
	Optimisation	Increase Stability
04.06	INTERNAL DEADLINE	