CARLA Paper

PROBLEM: Navigation in densely populated urban environments -> training in physical environment not easily possible
-> Solution: Simulation environments

- CARLA: open simulator
- provides different signals: GPS coordinates, speed, acceleration, data on collisions
- different weather conditions available: clear day, clear sunset, daytime rain, and daytime after rain
- three approaches are tested: modular pipeline, imitation learning, reinforcement learning
- API is implemented in Python
- Server-client architecture with sockets
- -> Find wrapper
- Client sends commands to server
- Receives sensor data
- Commands control vehicle -> steering acceleration braking
- Meta-commands: Control behavior of server (e.g. resetting simulation, changing environment properties …)

Sensors:

- Three sensoring modalities:
  - normal vision
  - ground-truth depth
  - ground-truth segmentation (road, lane-marking, traffic sign, sidewalk, fence, pole, wall, building, vegetation, vehicle, pedestrian, and other)
  - Traffic lights (rules), speed limits, collisions are already available

Reinforcement learning approach:

- A3C-algorithm
  - asynchronous -> enables running multiple simulation threads in parallel
  - Training on goal-directed navigation (in each training episode)
  - Terminated when goal is reached, collision, time budget exhausted
  - Reward: weighted sum of:
    - Positive: speed, distance traveled towards the goal
    - Negative: collision damage, overlap with sidewalk, opposite lane overlap
    - -> Karam hat es schon implementiert
  - Network was trained on 10 parallel actor threads -> 10 M simulation steps
  - Different difficulty steps: Straight, one turn, navigation, navigation with dynamic obstacles (do not have to start with navigation immediately)
  - Pay attention to weather conditions (maybe not with segmentation?)
  - Reinforcement learning for navigation does not perform well

- Why does RL not perform well?:
  - is known to be brittle (empfindlich)
  - extensive hyperparameter search becomes infeasible
    - trial and error approach
  - more difficult task than previous tasks that RL was used for
  - has been trained without dropouts (unimportant neurons are eliminated -> more stable training)

Technical details

- Client-server system
- Server: Renders CARLA world
- Client: Interface to interact with the simulation (controlling vehicle or simulation props)

- Commands:
  - Steering: -1,1
  - Throttle: 0,1
  - Brake: 0,1
  - Hand-brake: 0 or 1
  - Reverse gear: 0 or 1
- Meta-commands:
  - No of vehicles: int
  - No of pedestrians: int
  - Weather ID: index int for different weather conditions
  - Seed Vehicles/Pedestrians
  - Set of Cameras with e.g. segmentation
- Sensor readings:
  - Player speed/position
  - Collision
  - Lane/Sidewalk Intersection
  - …
  - -> Create reward function from these

Adjustments Gym to CARLA

- Resize to 84x84 pixels + vector of measurements („concatenate")
- Additional input: high-level commands (provided by topological planner) one-hot encoded (active neuron is on, all other are 0)
- Processed by two separate inputs: convolutional module + fully-connected measurements
  - -> concatenated and processed jointly