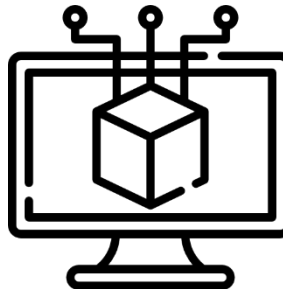
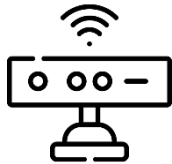


Autonomous Driving Software Engineering

Prof. Dr.-Ing. Markus Lienkamp

Phillip Karle, M. Sc.



Lecture Overview

Lecture – 90min	Practice – 45min
1 Introduction: Autonomous Driving Karle	1 Practice Karle
2 Perception I: Mapping Sauerbeck	2 Practice Sauerbeck
3 Perception II: Localization Sauerbeck	3 Practice Sauerbeck
4 Perception III: Detection Huch	4 Practice Huch
5 Prediction Karle	5 Practice Karle
6 Planning I: Global Planning Trauth	6 Practice Trauth
7 Planning II: Local Planning Ögretmen	7 Practice Ögretmen
8 Control Wischnewski	8 Practice Wischnewski
9 Safety Assessment Stahl	9 Practice Stahl
10 Teleoperated Driving Feiler	10 Practice Feiler
11 End-to-End Betz	11 Practice Betz
12 From Driver to Passenger Fank	12 Practice Karle

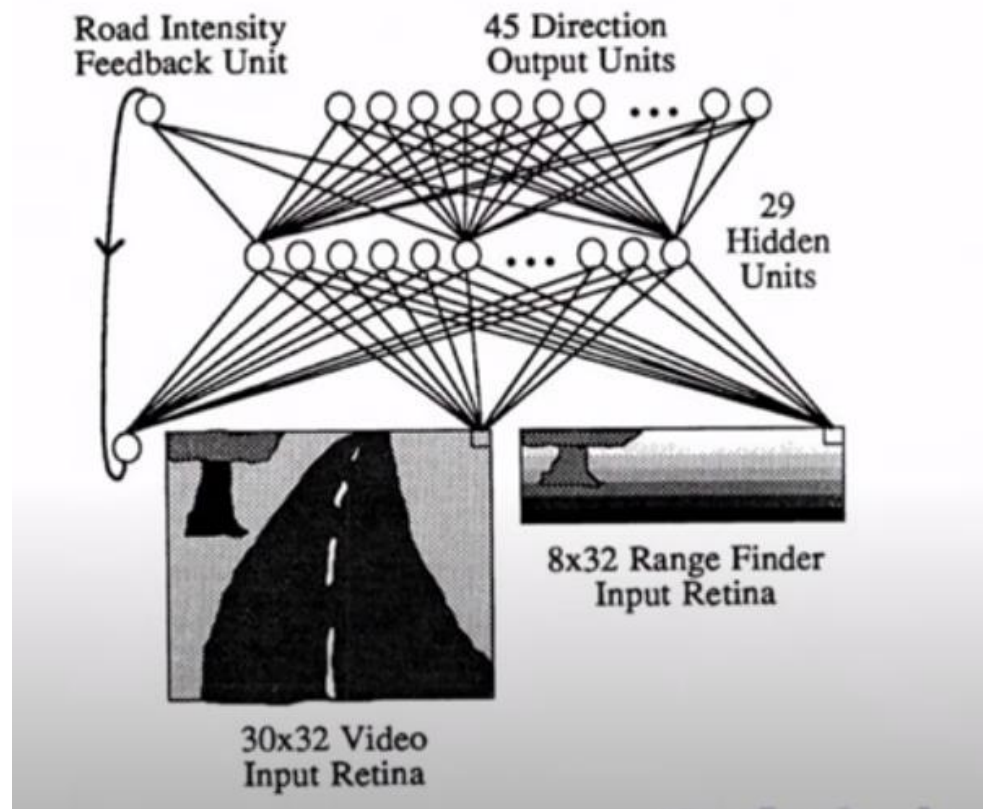


Source: https://www.youtube.com/watch?v=laolqVMd6tc&feature=emb_logo

Dean Pomerleau, „ALVINN: An Autonomous Land Vehicle in a Neural Network“, 1989

ALVINN

Replace the Driver with a Neural Network ...



Additional Slides

- 3-layer fully connected neural network designed for the task of road following
- Camera and laser range finder as input
- Direction output is a linear representation of the turn curvature along which the vehicle should travel in order to head towards the road center
- Road intensity feedback indicates whether the road is lighter or darker than the non-road in the previous image
- 40 epochs of training on 1200 simulated road snapshots
- Correctly dictates a turn curvature within two units of the correct answer approximately 90 % of the time on novel simulation data

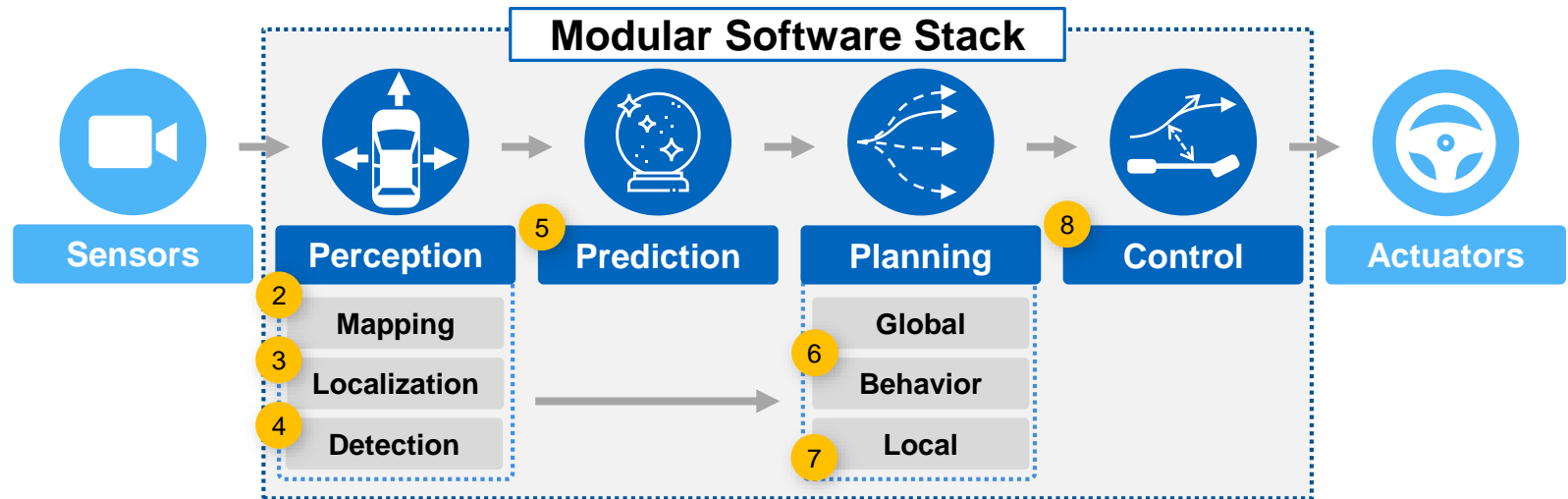
Additional Slides

Some further projects

- Pomerleau, Dean A. Alvin, **An autonomous land vehicle in a neural network.**, 1989
- Lecun, Y., Cosatto, E., Ben, J., Muller, U., & Flepp, B., **Dave: Autonomous off-road vehicle control using end-to-end learning.**, 2004
- Koutník, Jan, et al., **Evolving large-scale neural networks for vision-based TORCS.**, 2013
- Vitelli, Matt, and Aran Nayebi., **Carma: A deep reinforcement learning approach to autonomous driving.**, 2016
- Bojarski, Mariusz, et al., **End to end learning for self-driving cars.**, 2016
- Talpaert, Victor, et al., **Exploring applications of deep reinforcement learning for real-world autonomous driving systems.**, 2019

Modular Approach

Lecture 2-8



- Numerous sequential and parallel modules and submodules
- Requires robust solutions for all individual algorithms
- Each algorithm is developed separately, ignoring goals of the driving task (travel time, safety, comfort)
- Modular approach good for human debugging

Motivation

Driving Task

Human

Car



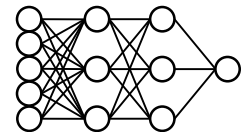
Eyes

Camera



Brain

Neural Network



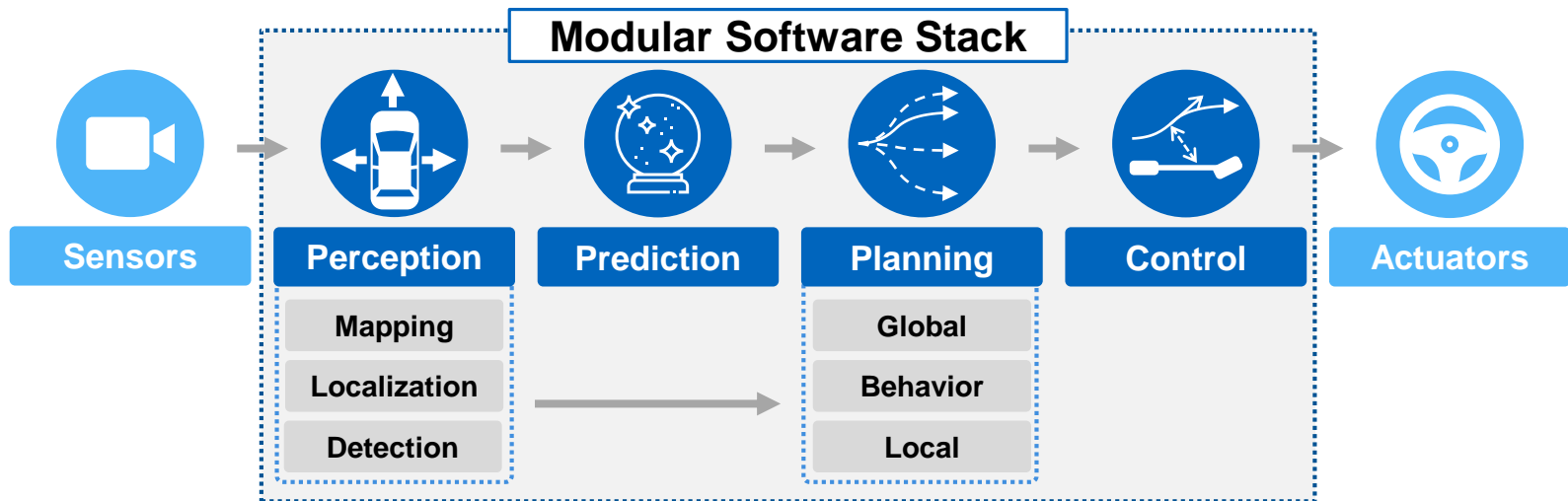
Control

Control



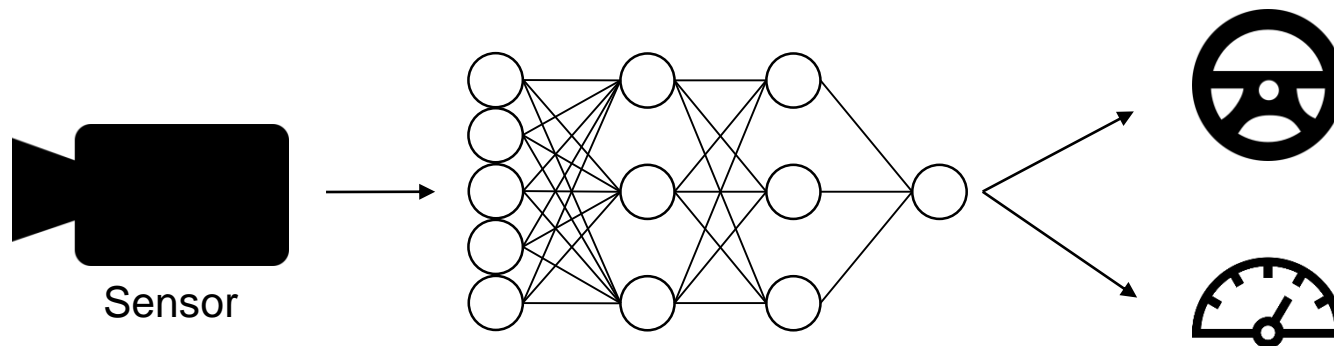
Motivation

Replacing the Modular Approach



Motivation

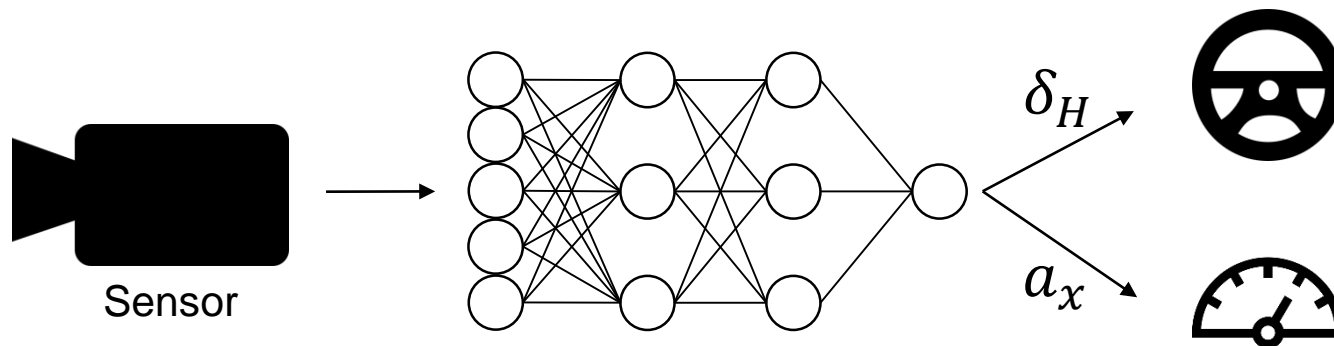
Replacing the Modular Approach



Motivation

End-to-End Learning

- Only two outputs are relevant for driving:
 - Steering δ_H
 - Acceleration a_x



Solution: Combination of the entire driving stack in a single model, which enables end-to-end learning

Objectives for Lecture 9: End-to-End

After the lecture you are able to ...

... describe the differences between the modular and end-to-end approach for autonomous driving and explain the motivation behind the end-to-end approach.

... classify the different types of imitation learning.

... illustrate the background of reinforcement learning and its application in autonomous driving.

... discuss the reason to combine modules and explain the basic functionality of the approach.

... argue the limitations of the end-to-end approach.

Remember Understand Apply Analyze Evaluate Develop



Depth of understanding

End-to-End Autonomous Driving

Prof. Dr. Markus Lienkamp

Tobias Betz, M. Sc.

Agenda

1. Introduction
2. **Methods**
3. Combination of Modules
4. Limitations
5. Summary



End-to-End Autonomous Driving

Prof. Dr. Markus Lienkamp

Tobias Betz, M. Sc.

Agenda

1. Introduction
2. Methods
 - 2.1 Imitation Learning
 - 2.2 Reinforcement Learning
3. Combination of Modules
4. Limitations
5. Summary



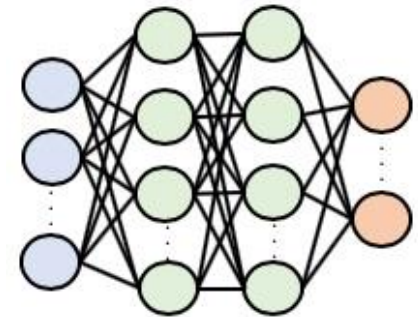
Imitation Learning

Explanation of Imitation Learning

Imitation Learning is a means of learning and developing new skills from observing these skills performed by an expert.



Expert demonstration

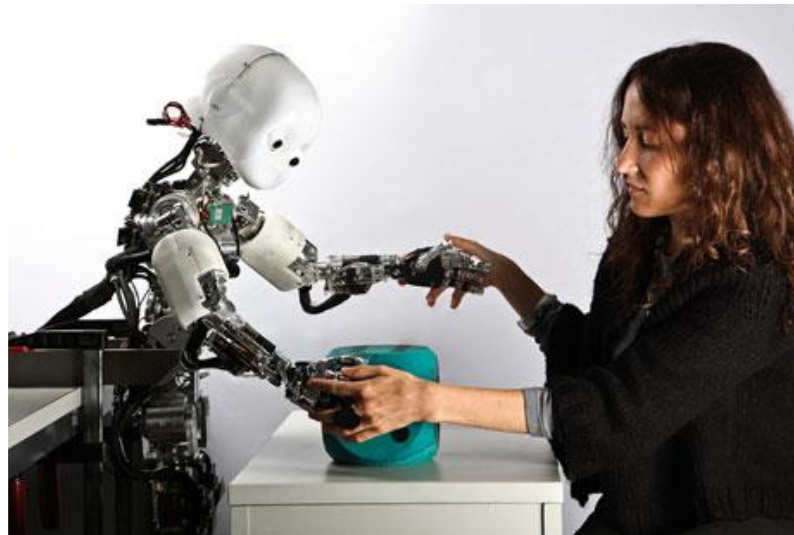


Learning

Imitation Learning

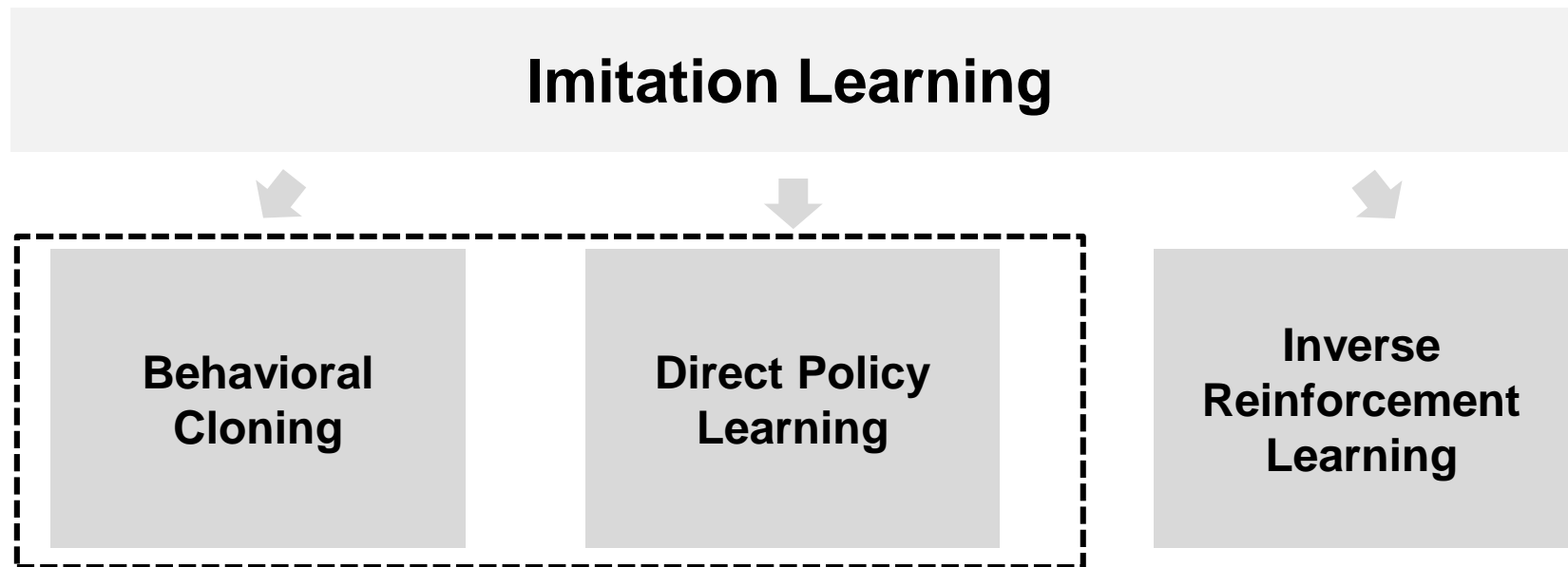
When should you use imitation learning?

- Imitation learning is useful when it is easier for the expert to demonstrate the desired behavior rather than:
 - Specifying the desired policy directly
 - Specifying a reward that would generate such behavior



Imitation Learning

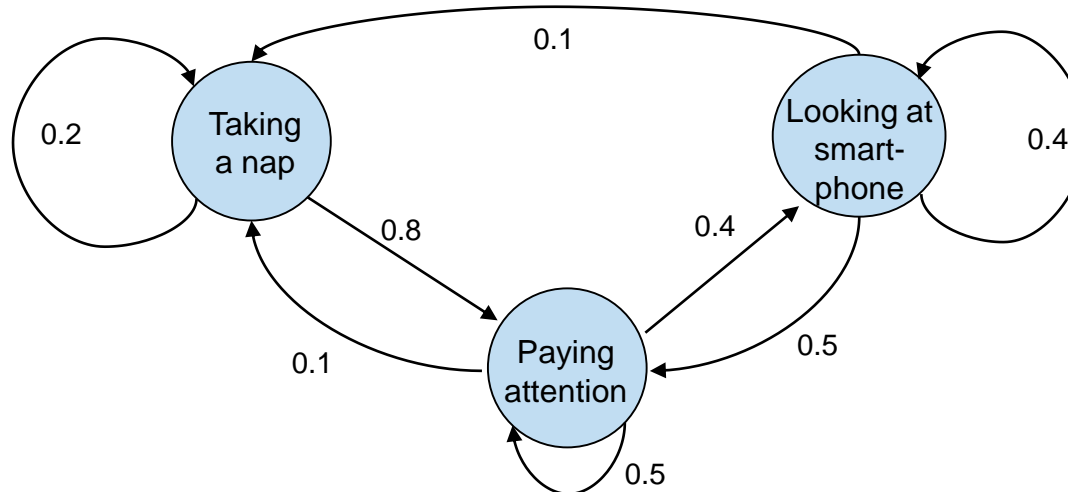
Different Methods of Imitation Learning



Imitation Learning

Markov Property

- A Markov chain represents a Markov process of state transitions, where the Markov property is assumed



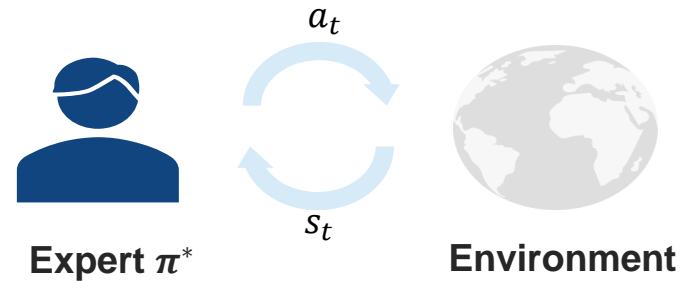
- Markov property: The state of the system at time $t + 1$ depends **only** on the state of the system at time t

$$P(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = P(s_{t+1} | s_t, a_t)$$

Imitation Learning

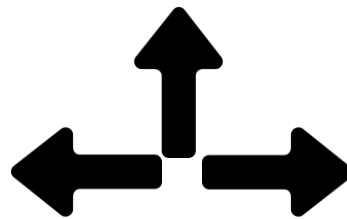
Notation

- State: $s_t \in \mathcal{S}$
- Action: $a_t \in \mathcal{A}$
- Expert policy: π^*
- Imitator policy: π_θ
 - The policy π_θ models a probability for an action a_t given a state s_t :
 $a_t \sim \pi_\theta(a_t | s_t)$
 - It maps states to (discrete) actions $\pi_\theta(s_t) \rightarrow a_t$
 - ... or distributions over actions: $\pi_\theta(s_t) \rightarrow P(a_t)$

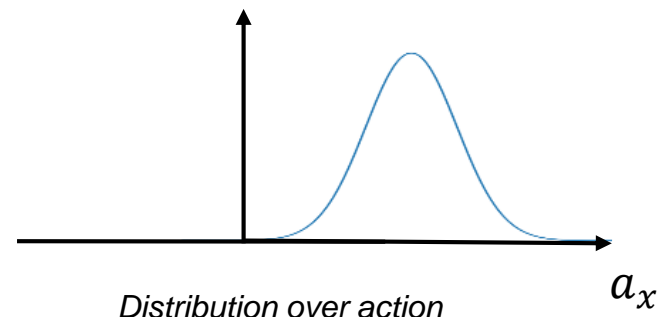


[0.03 0.45]

Continuous action



Discrete action



Imitation Learning

Notation

- Rollout: sequentially execute $\pi(s_0)$ on an initial state
 - Produce trajectory $\tau = \{(s_0, a_0), (s_1, a_1), \dots, (s_T, a_T)\}$
- State Dynamics: $P(s_{t+1} \mid s_t, a_t)$
 - Typically not known to policy
 - Essentially the simulator/environment
- $P(s \mid \pi)$: distribution of states induced by a policy



Imitation Learning

Example 1: Real World Autonomous Driving

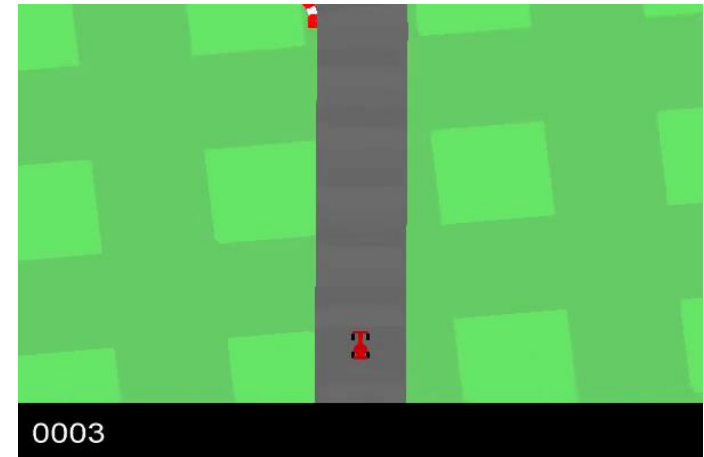
- s = RGB camera image
 - $a = [a_x, \delta_H]^\top$
 - Training set: $\mathcal{D} = \{\tau_1, \tau_2, \dots, \tau_L\}$,
where $\tau_l = \{(s_0, a_0), (s_1, a_1), \dots, (s_T, a_T)\}$
 - Trajectories τ_l sampled with expert policy π^*
- ➔ Learn optimal imitator policy π_θ



Imitation Learning

Example 2: Virtual Car Racing

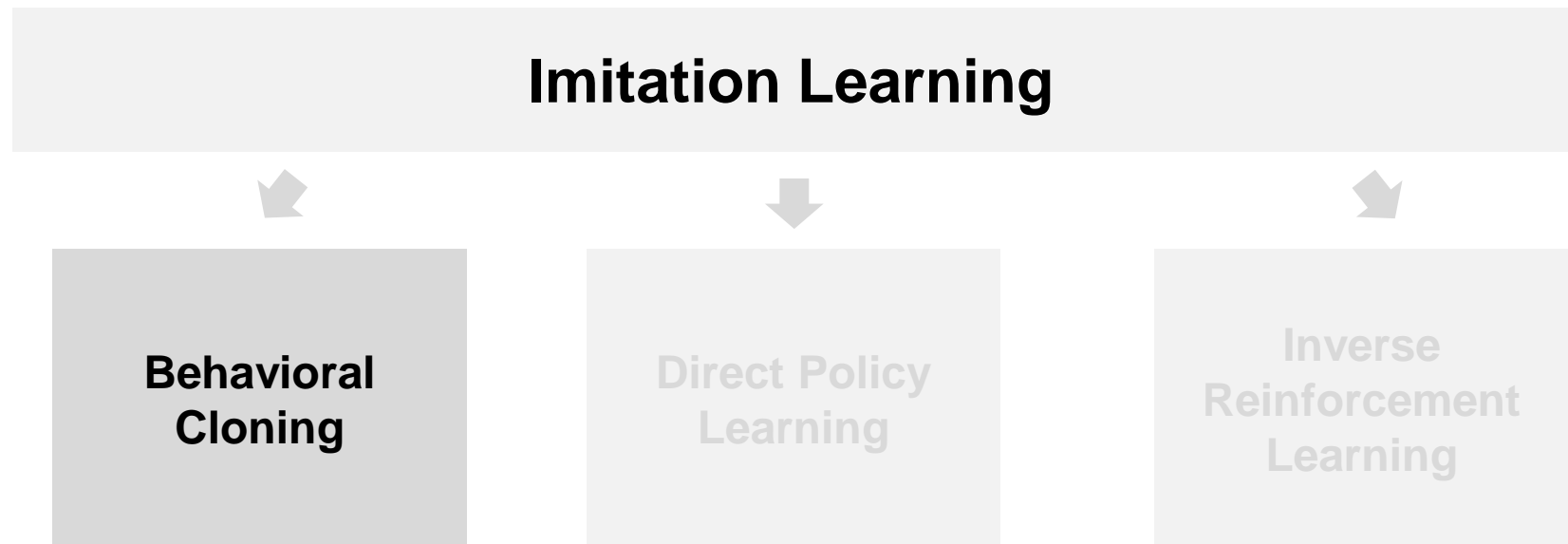
- s = game screen (3x96x96 pixels)
 - $a = [\text{steer}, \text{gas}, \text{brake}]^\top$
 - Training set: $\mathcal{D} = \{\tau_1, \tau_2, \dots, \tau_L\}$,
where $\tau_l = \{(s_0, a_0), (s_1, a_1), \dots, (s_T, a_T)\}$
 - Trajectories τ_l sampled with expert policy π^*
- ➔ Learn optimal imitator policy π_θ



» Example will be shown in the practice session.

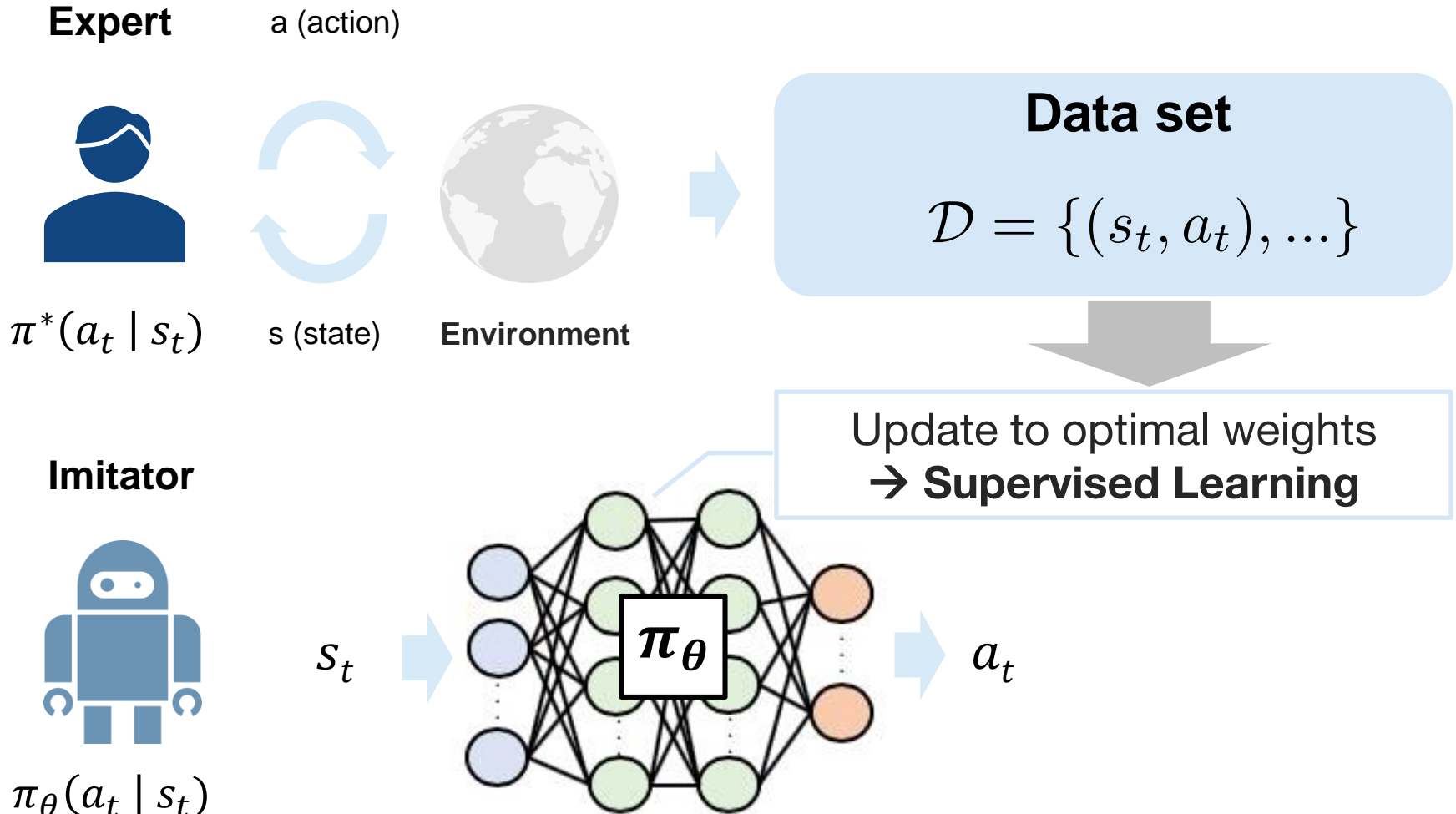
Imitation Learning

Different Methods of Imitation Learning



Imitation Learning

Behavioral Cloning



Imitation Learning

Behavioral Cloning

Behavioral Cloning = Reduction to supervised learning

- Define $P^* = P(s \mid \pi^*) \rightarrow$ Distribution of states visited by expert




- Learning goal:

$$\operatorname{argmin}_{\theta} E_{(s, a^*) \sim P^*} L(a^*, \pi_{\theta}(s))$$

Imitation Learning

NVIDIA DAVE 2



NVIDIA Automotive Team begins testing
an autonomous vehicle using Deep Learning.

Imitation Learning

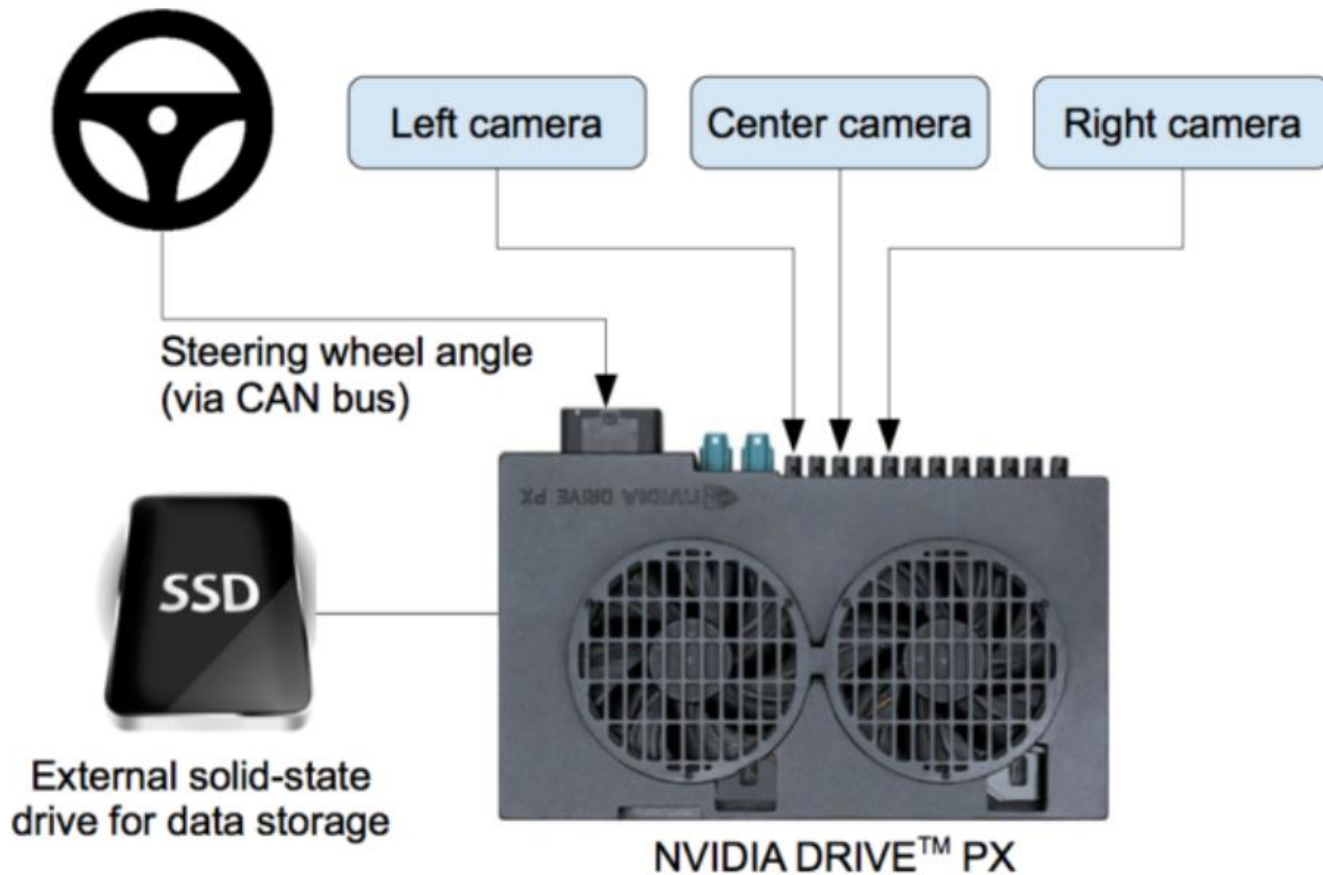
Behavioral Cloning in DAVE-2

- Trained CNN to map raw pixels from a single front-facing camera directly to steering commands
- With minimum training data the system learns to drive in traffic on local roads and operates in areas with unclear visual guidance such as in parking lots and on unpaved roads
- Data augmentation to enrich the data set
- The system learns detecting useful road features with only the steering angle as the training signal



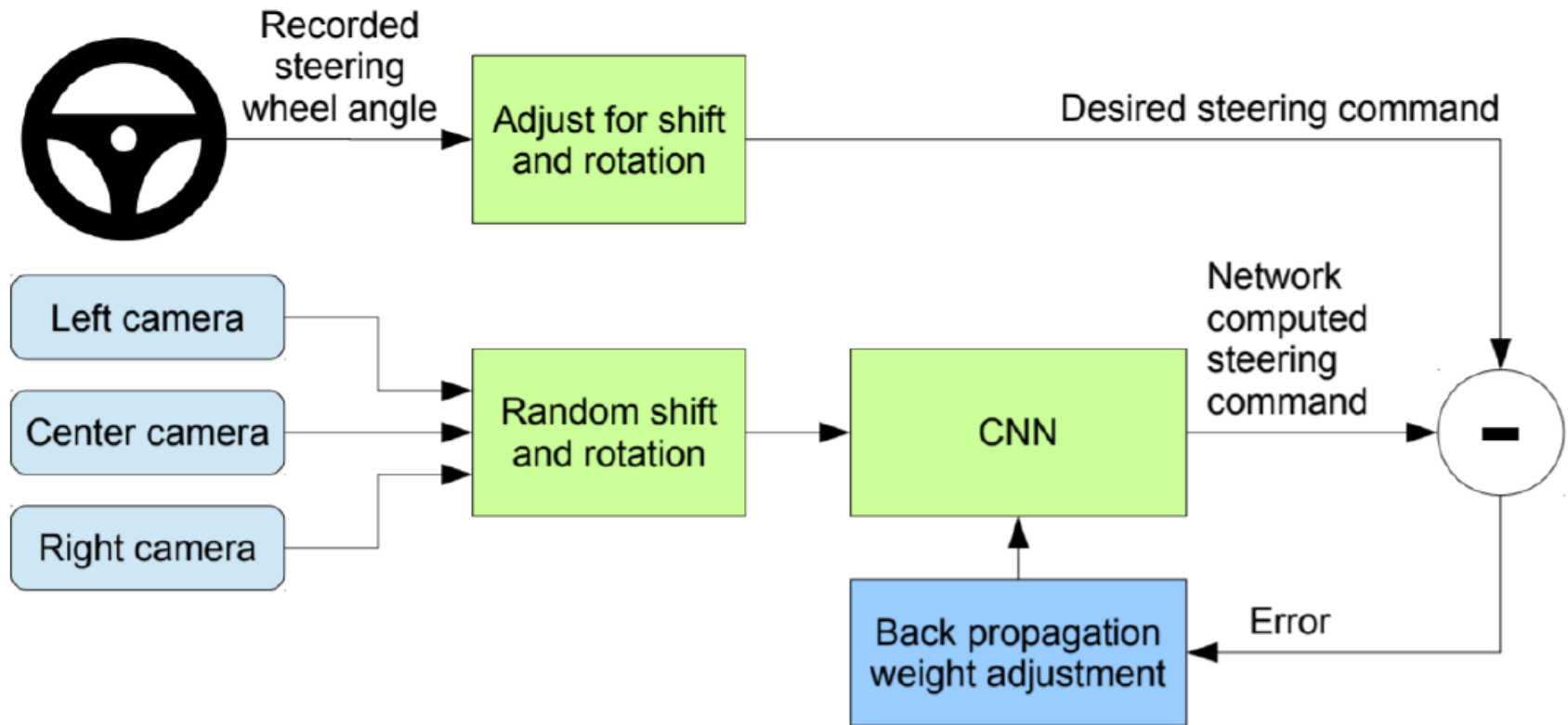
Imitation Learning

DAVE-2: Data Collection System



Imitation Learning

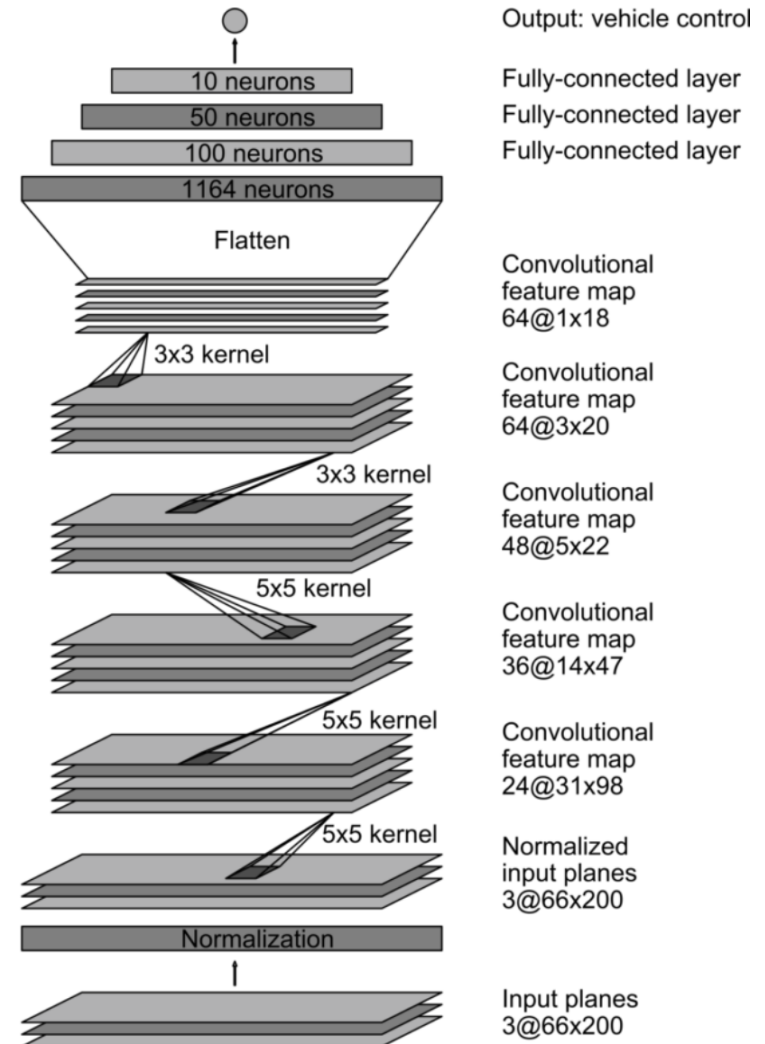
DAVE-2: Training the Neural Network



Imitation Learning

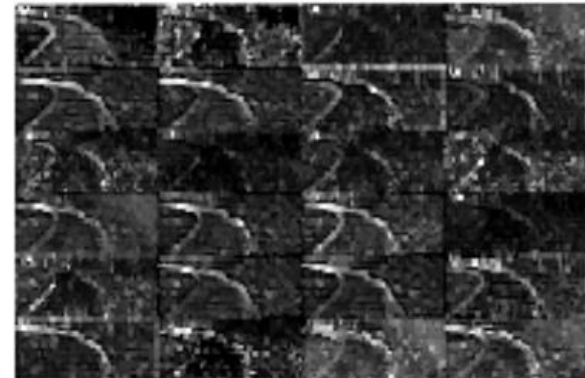
DAVE-2: CNN Architecture

- The network has about 27 million connections and 250 thousand parameters
- Input is a cropped RGB image (sky contains useless information)
- Output is the inverse turning radius ($\frac{1}{r}$, where r is turning radius in meter)



Imitation Learning

DAVE-2: Visualization of Internal CNN State



- ➡ With only the human steering angle as training signal the network trained to detect the outlines of roads

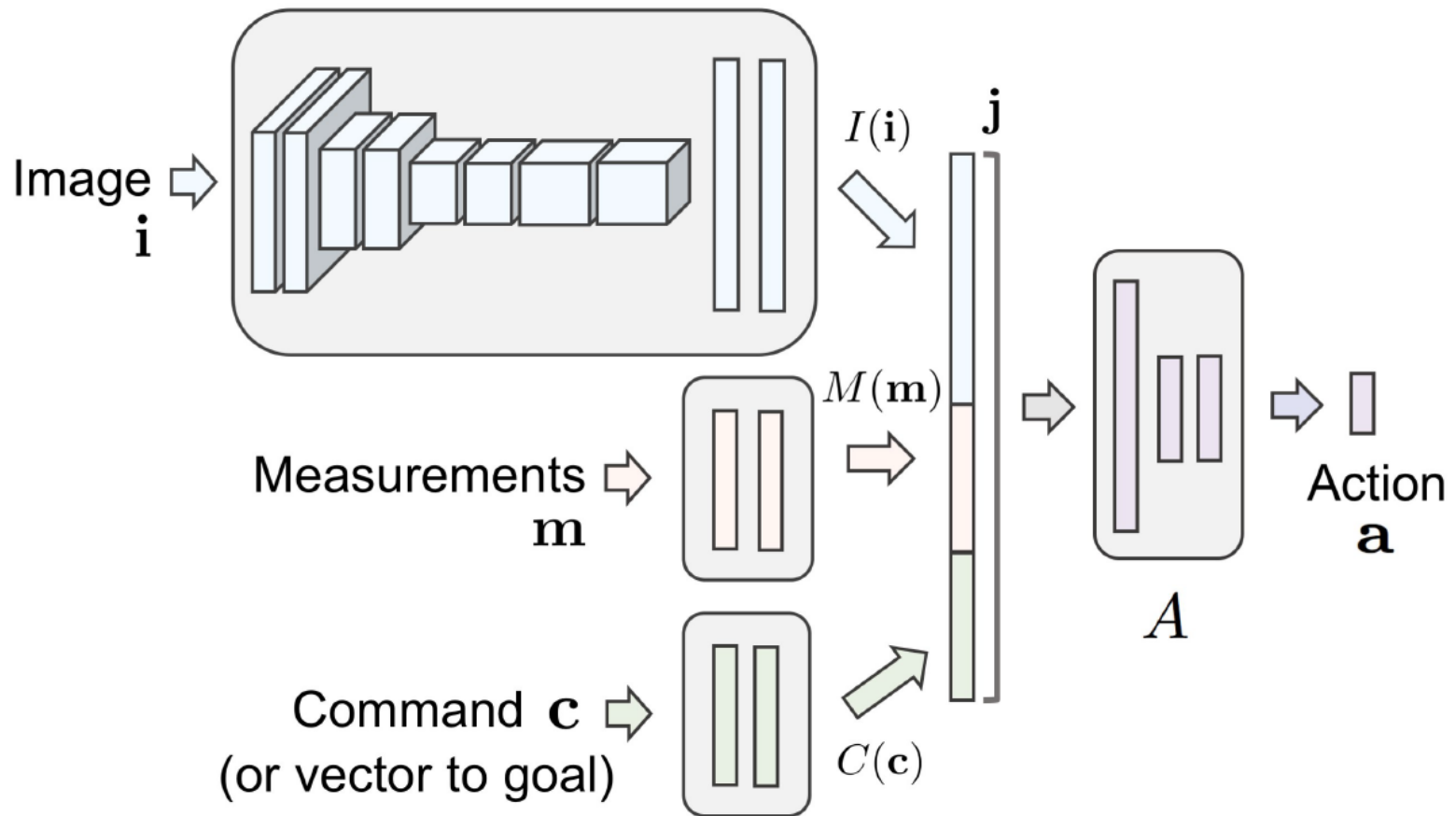
Imitation Learning

What happens at an intersection?



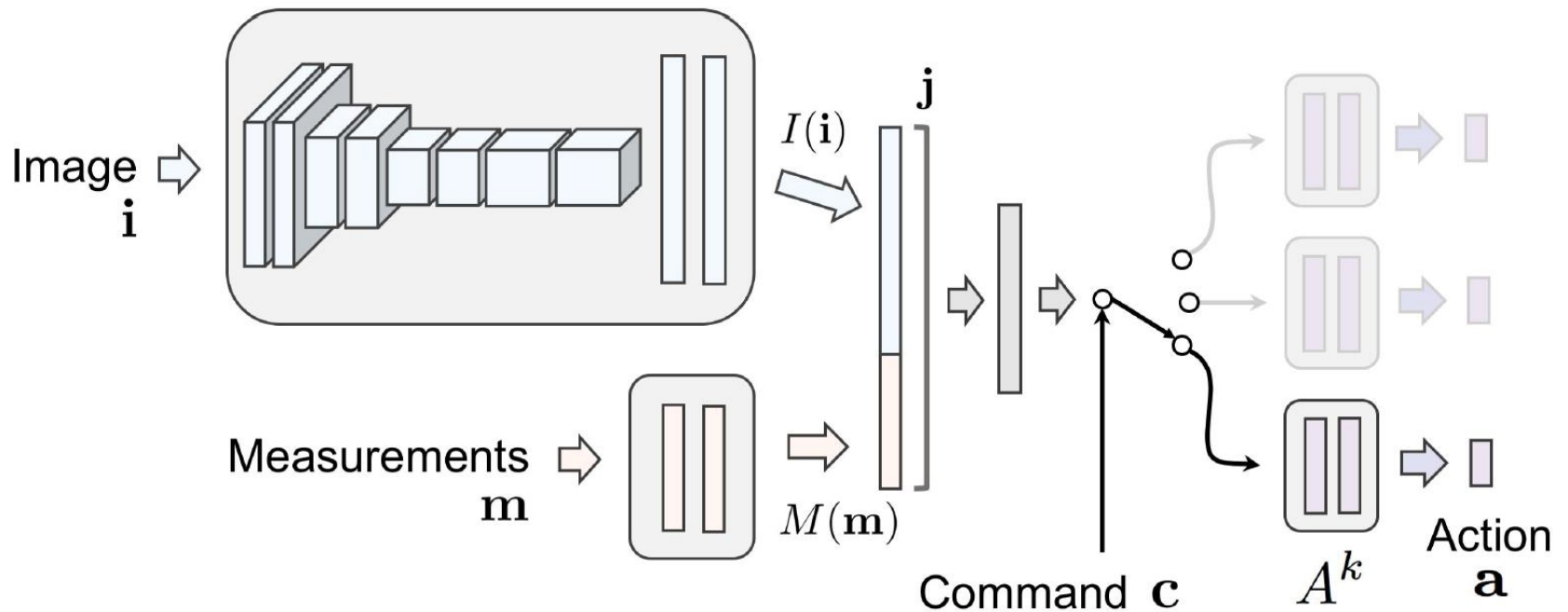
Imitation Learning

Conditional Imitation Learning – Command Input



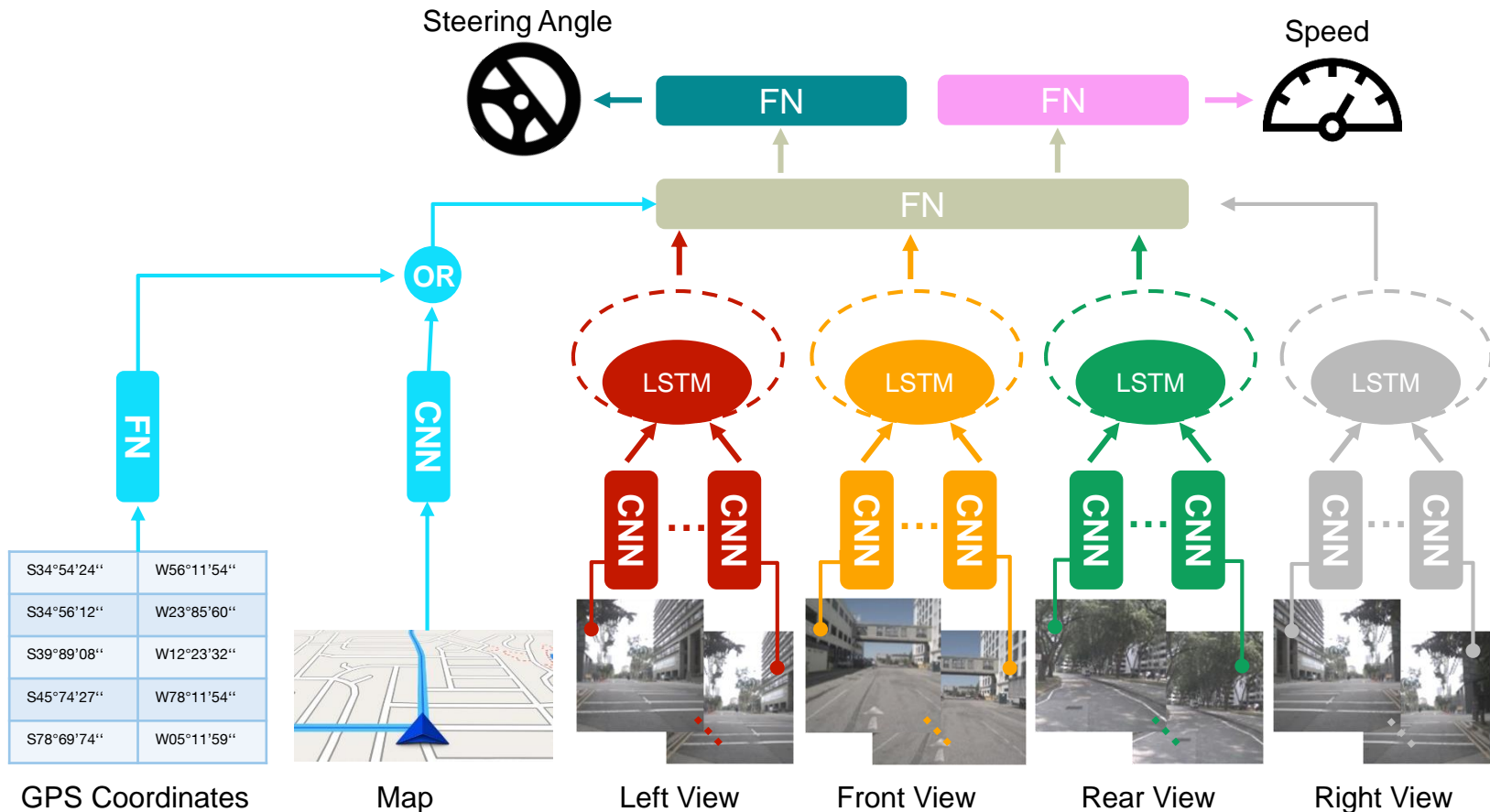
Imitation Learning

Conditional Imitation Learning – Branched Output



Imitation Learning

Conditional Imitation Learning – Route Planner



Imitation Learning

Conditional Imitation Learning

- **Until now:**
 - We can predict steering angle and acceleration
 - We can integrate the route information

Do we now have the perfect solution for an autonomous driving software?



Imitation Learning

Questioning the Markov Property



Imitation Learning

Questioning the Markov Property



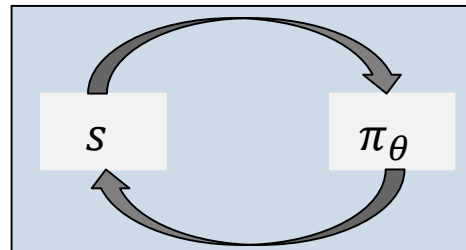
Imitation Learning

Questioning the Markov Property

- Supervised learning assumes the Markov Property and ignores temporal structure.

$$P(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = P(s_{t+1} | s_t, a_t)$$

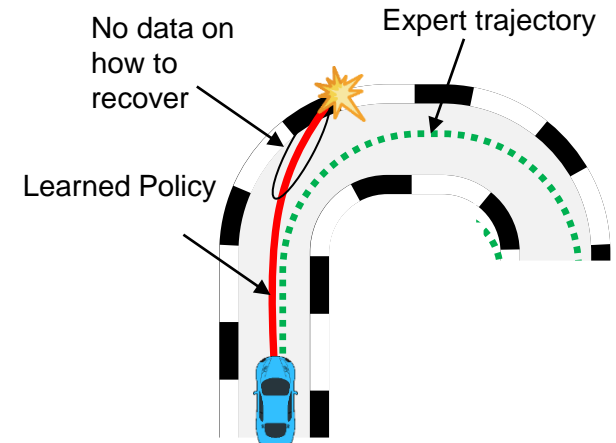
- But, the state s_{t+1} depends on all previously taken actions a of the imitator



Imitation Learning

Behavioral Cloning – Compounding Errors

- Decisions are purposeful, in supervised learning we don't have a goal or planning problem
- Small errors compound over time
- State distribution under learned π_θ differs from those generated by π^*

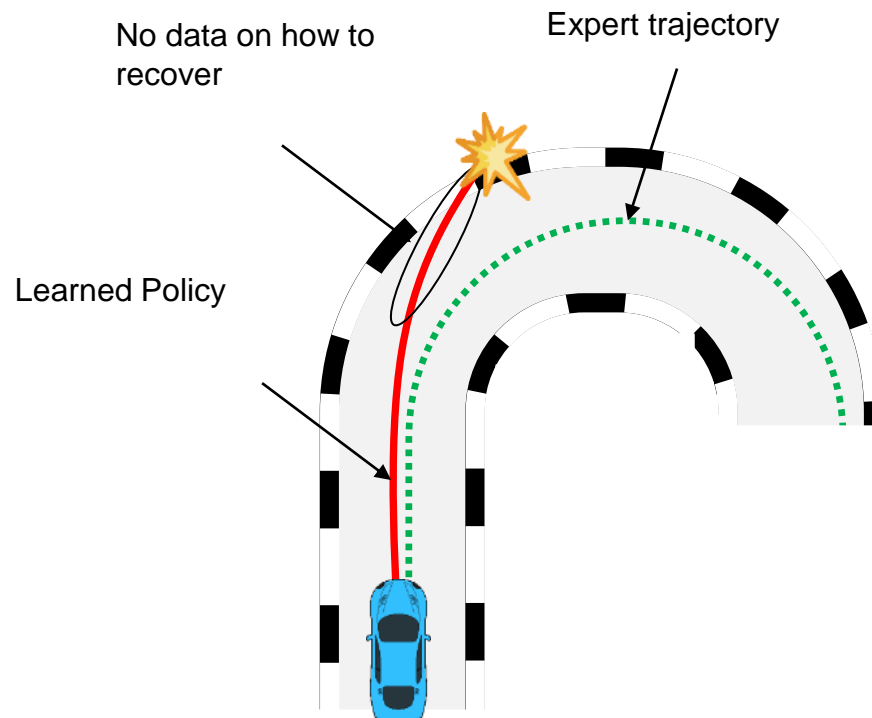


$$P(s \mid \pi^*(s)) \neq P(s \mid \pi_\theta(s))$$

Supervised learning succeeds when training and test data distribution match.

Imitation Learning

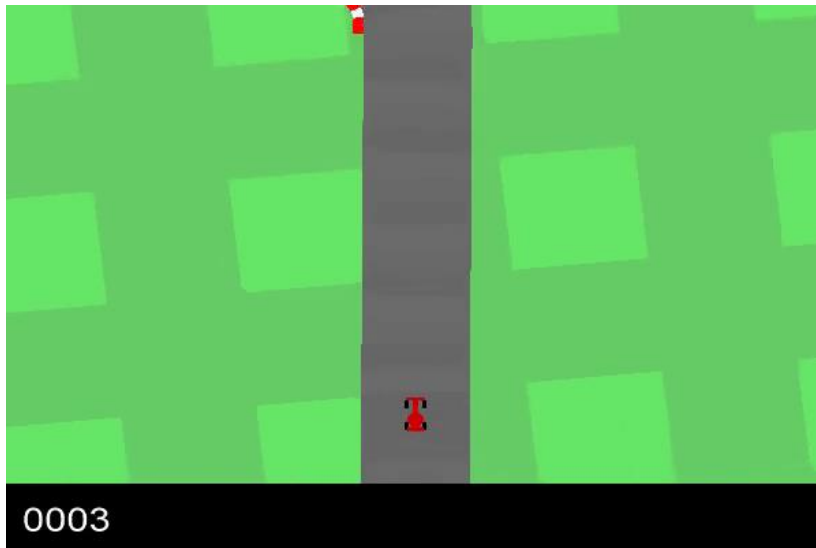
Limitations of Behavioral Cloning



Imitation Learning

Limitations of Behavioral Cloning

Expert Trajectory



Imitator Policy



Imitation Learning

Discussion of Behavioral Cloning



ADVANTAGES

- Simple
- Efficient

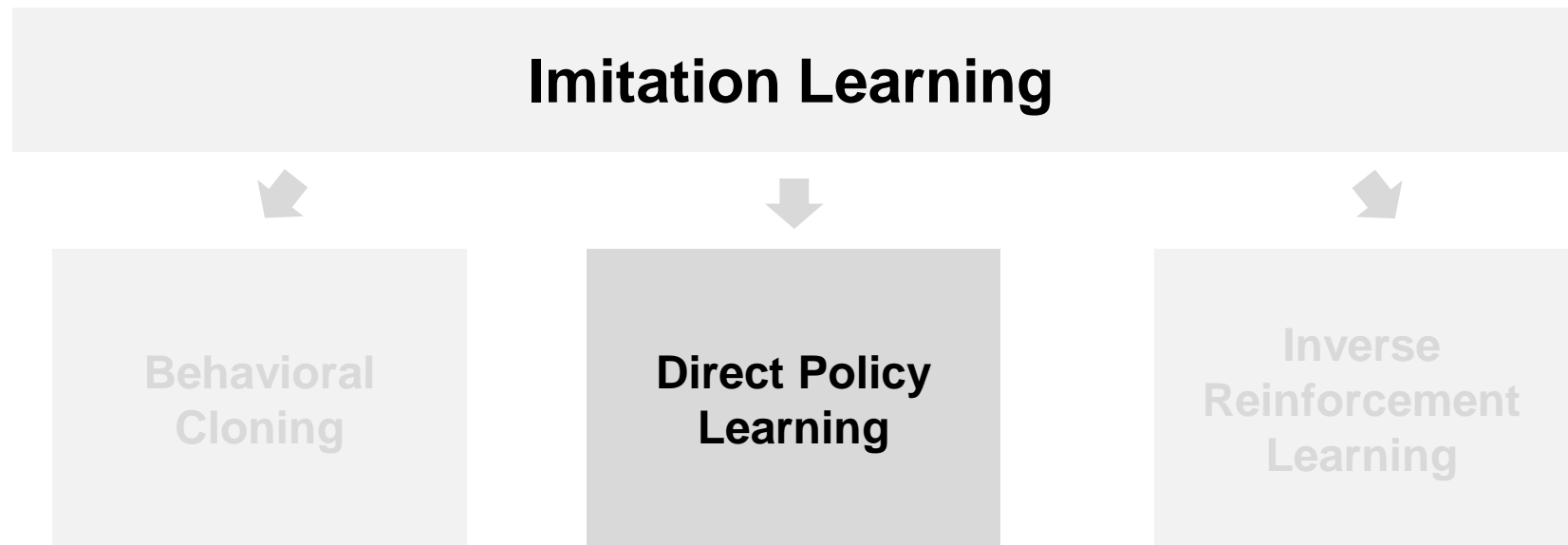


DISADVANTAGES

- No long-term planning
- 1-step deviations can lead to catastrophic error
- Collecting expert demonstrations for all possible situations is not possible
- Distribution mismatch between training and testing

Imitation Learning

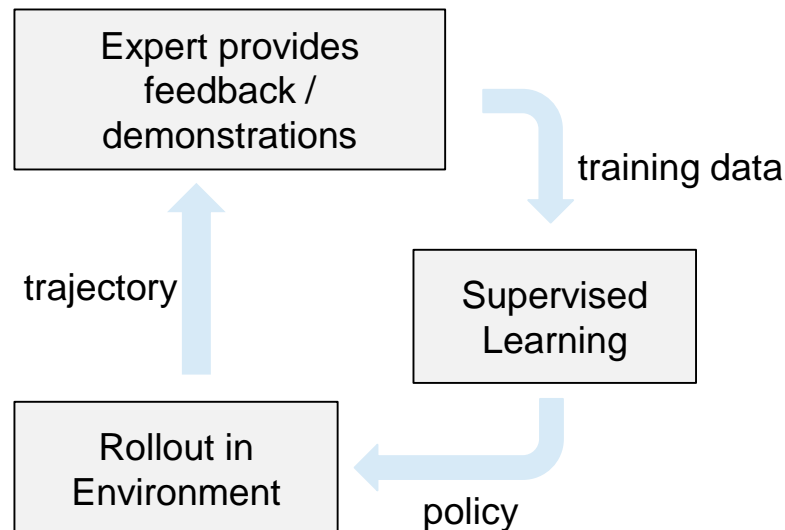
Different Methods of Imitation Learning



Imitation Learning

Direct Policy Learning

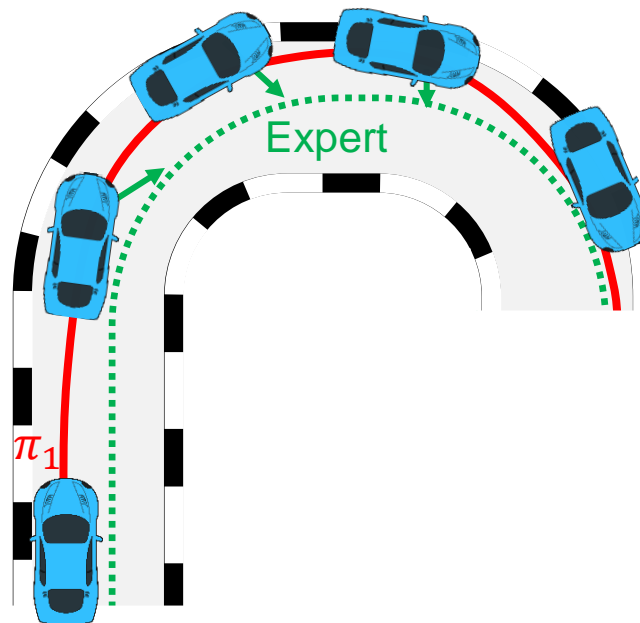
- Direct policy learning (DPL) is an improved version of behavioral cloning
- **Idea:** Get more labels of the expert action along the path taken by the policy computed by behavior cloning
- Expert provides feedback on the roll-out trajectory
- Loop continues until convergence \rightarrow ideally converges to π^*



Imitation Learning

Direct Policy Learning – What is Expert Feedback?

- Interactive expert is typically a human and is available at training time
- At any state expert can be queried to provide feedback on the state distributions induced by the policy (not only state distribution from expert policy)



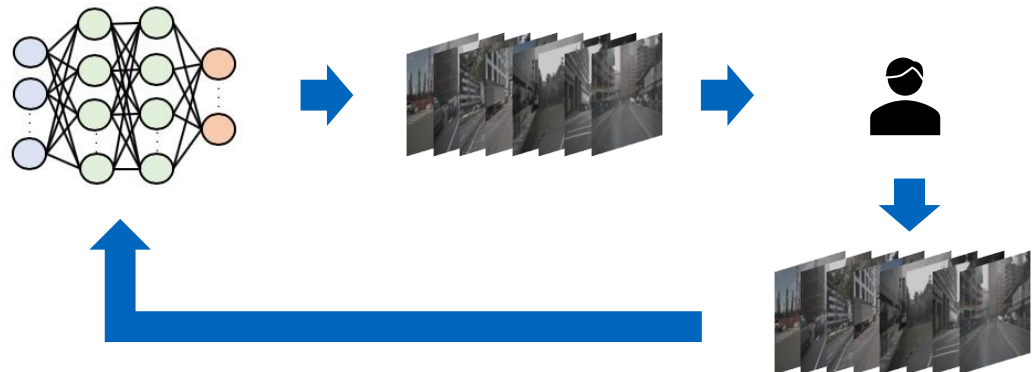
Imitation Learning

Direct Policy Learning – Naive Attempt

1. Fix data set, train π_θ



2. Fix π_θ , record data set
→ Collect expert feedback



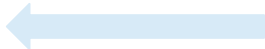
3. Repeat

- Not guaranteed to converge
- Can result in oscillation
- Previously used data were the imitator makes a mistake will get lost



Imitation Learning

Direct Policy Learning – General Algorithm

Initial predictor: π_0  Initial expert demonstrations

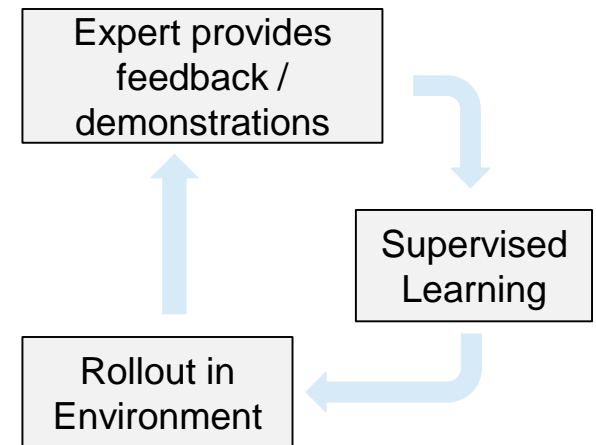
For $m = 1$

- Collect trajectories τ via rolling out π_{m-1}
- Collect interactive feedback $\{\pi^*(s) \mid s \in \tau\}$
- Use one of the following techniques to learn the policy:
 - **Data Aggregation** (e.g., DAgger) → „Append Dataset“
 - Train π_m on $P_1 \cup \dots \cup P_m$
 - **Policy Aggregation** (e.g., SEARN & SMILe) → „Append Policy“
 - Train π'_m on P_m
 - $\pi_m = \beta * \pi'_m + (1 - \beta) * \pi_{m-1}$ for $\beta \leq 1$

Imitation Learning

Direct Policy Learning – Summary

- Reduction to sequence of supervised learning problems
 - Constructed from rollouts of previous policies
 - Requires interactive expert feedback
- Two approaches: Data Aggregation & Policy Aggregation
 - Ensures convergence



Imitation Learning

Discussion of Direct Policy Learning



ADVANTAGES

- More robust than standard behavioral cloning

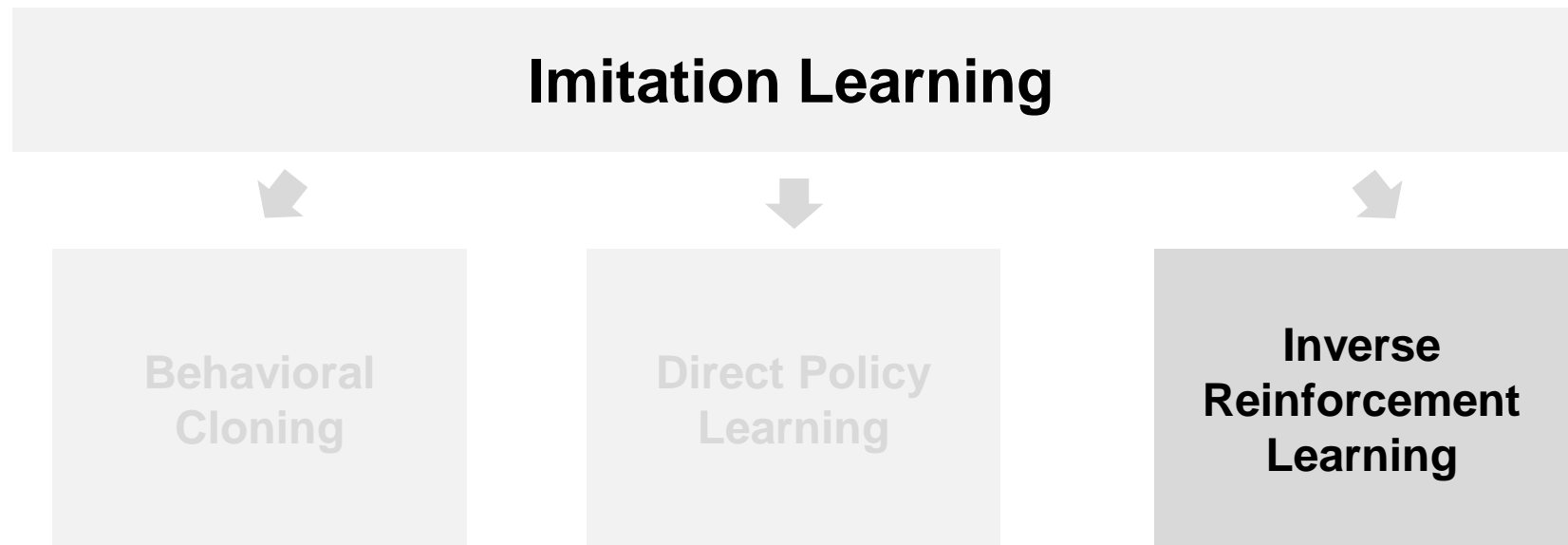


DISADVANTAGES

- Interactive expert is required
- Interaction with the environment

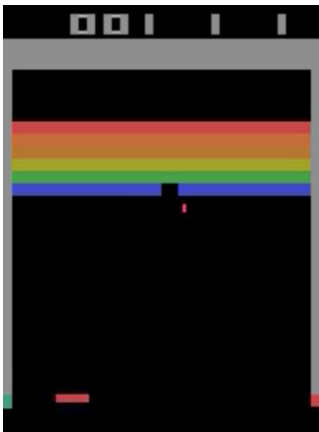
Imitation Learning

Different Methods of Imitation Learning



Imitation Learning

Inverse Reinforcement Learning – Reward Function



What is the reward?



<https://gym.openai.com/envs/Breakout-v0/>

Dario, Jack. „Faulty Reward Functions in the Wild“, 2016

Finn, Chelsea, Sergey Levine, and Pieter Abbeel. „Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization.“, 2016

Jamie Page Deaton, „What is the DARPA Urban Challenge?“

Imitation Learning

Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL)

- Inferring reward function from demonstrations / roll-outs of expert policy
→ Sometimes more efficient to learn reward function than directly learn the policy from the demonstrations

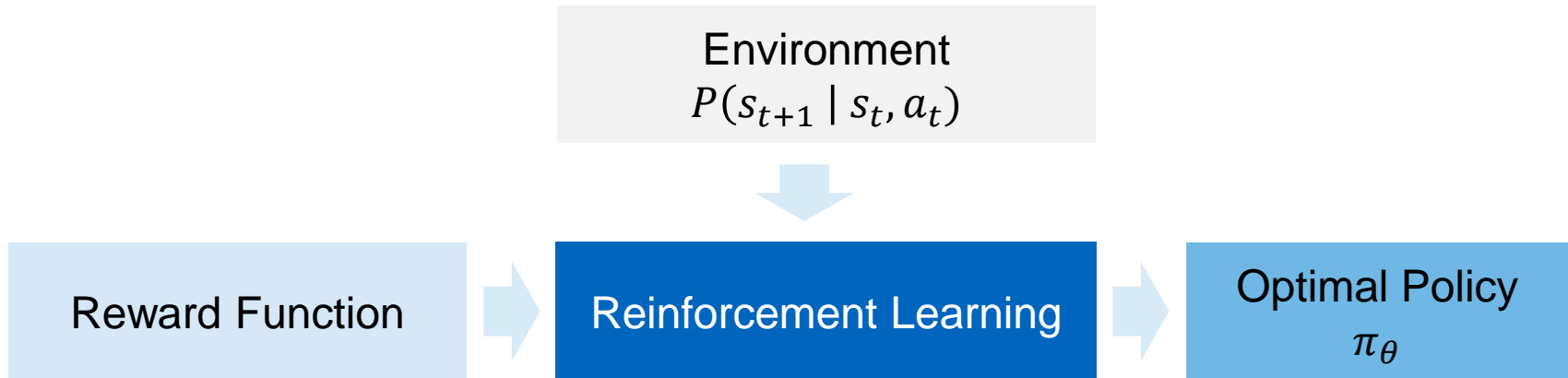
Challenges

- Underdefined problem
→ Many reward functions correspond to the same policy
- Difficult to evaluate a learned reward
- Demonstrations may not be precisely optimal



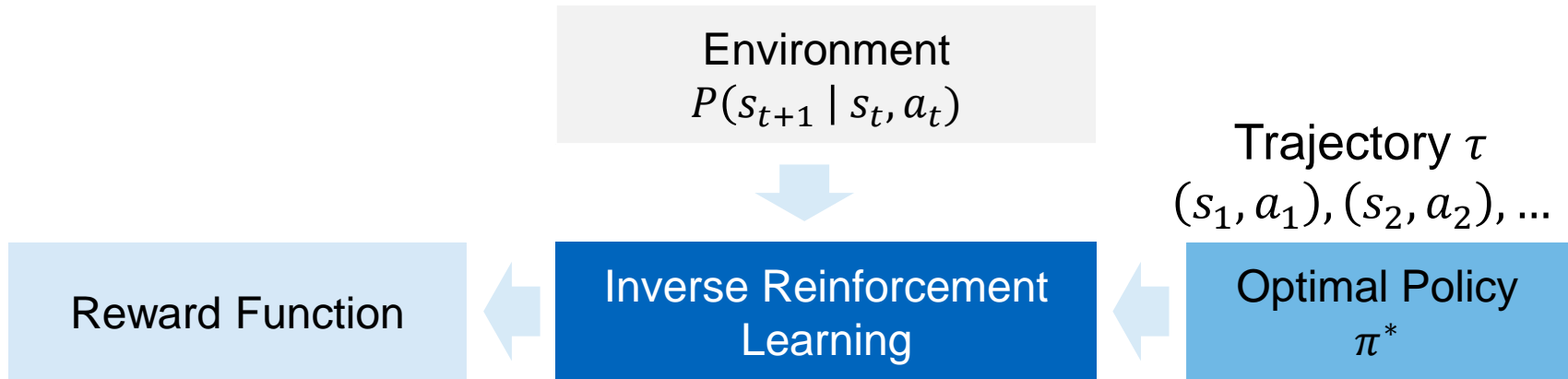
Imitation Learning

Schematic Reinforcement Learning



Imitation Learning

Schematic Inverse Reinforcement Learning

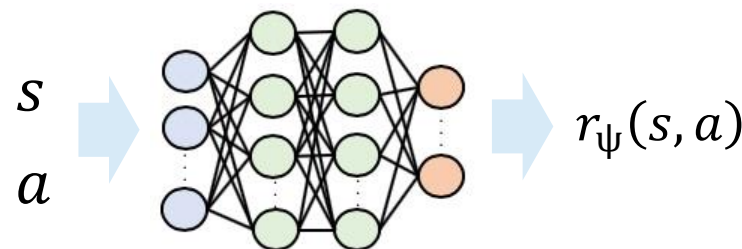


➡ Using the reward function to find a policy π_θ

Imitation Learning

Inverse Reinforcement Learning – High-Level Recipe

- Expert demonstrations: $\mathcal{D} = \{\tau_1, \tau_2, \dots, \tau_L\}$
- Learn reward function: $r_\psi(s, a)$
 - linear feature reward function: $r_\psi(s, a) = \sum_i \psi_i f_i(s, a) = \psi^T f(s, a)$
 - neural net reward function:
 - ...



- Learn policy given reward function (RL)
- Compare learned policy with expert

Imitation Learning

Inverse Reinforcement Learning – Key Papers

- **Apprenticeship Learning via Inverse Reinforcement Learning** – Abbeel & Ng, ICML 2004
- **A Game-Theoretic Approach to Apprenticeship Learning** – Syed & Schapire, NIPS 2007
- **Maximum Entropy Inverse Reinforcement Learning** – Ziebart et al., AAAI 2008
- **Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization** – Finn et al., ICML 2016

Imitation Learning

Comparison of Imitation Learning Techniques

	Output	Interaction with Environment	Interactive Expert	Pre-collected Demonstrations
Behavioral Cloning	Policy	No	No	Yes
Direct Policy Learning	Policy	Yes	Yes	Optional
Inverse Reinforcement Learning	Reward Function	Yes	No	Yes

End-to-End Autonomous Driving

Prof. Dr. Markus Lienkamp

Tobias Betz, M. Sc.

Agenda

1. Introduction
2. Methods
 - 2.1 Imitation Learning
 - 2.2 Reinforcement Learning**
3. Combination of Modules
4. Limitations
5. Summary



Additional Slides

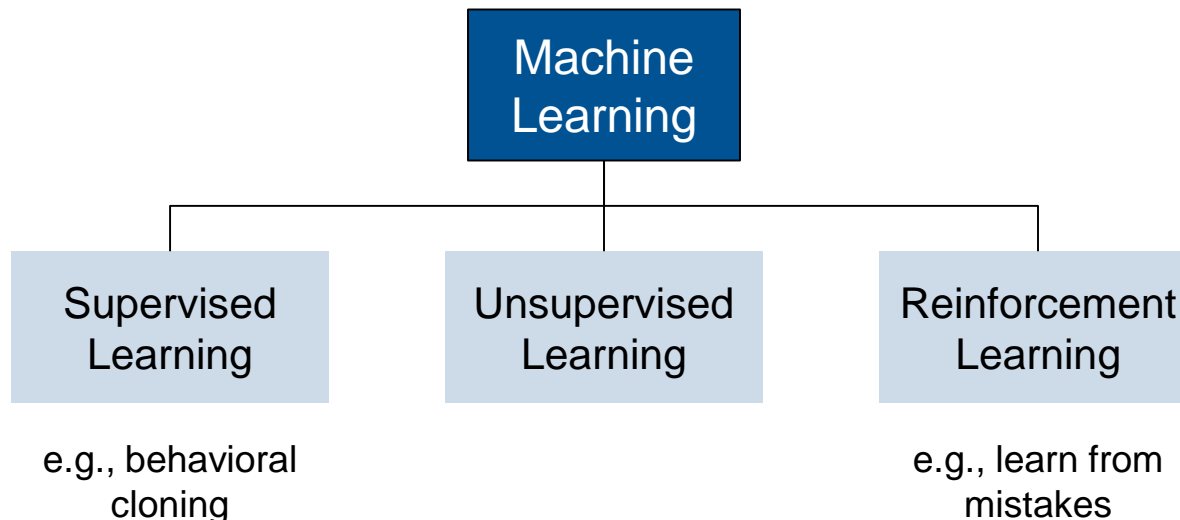
In this lecture we cannot cover the full theory behind reinforcement learning.

See „Artificial Intelligence in Automotive Technology – Lecture 11: Reinforcement Learning“ for theoretical background.

Reinforcement Learning

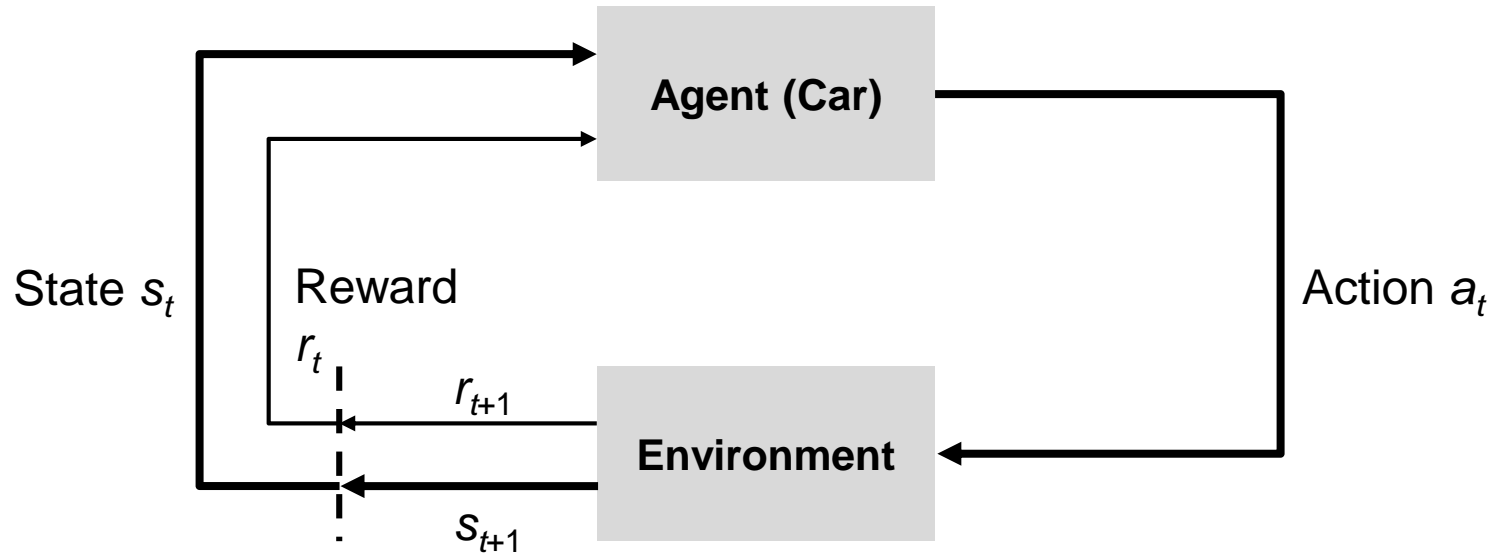
Motivation

- Supervised learning requires training data, which might not cover all situations encountered on real-world
- Due to the trial-and-error nature of reinforcement learning (RL), it is better at exploration of unseen situations, which is beneficial in the context of autonomous driving



Reinforcement Learning

Basics



- Agent interacts with the environment in a continuous form
- At each timestep t , environment reacts to the selected actions a_t and presents new observations (state s_{t+1}) to the agent
- Agent also receives a new reward r_{t+1} and seeks to maximize rewards over time

Reinforcement Learning

Basics

Policy:

- Same meaning as in imitation learning, it characterizes the agents behaviour by mapping states to actions

Reward:

- Every actions taken by the agent results in a reward (positive or negative) given to the agent:
→ reward function
- In the context of autonomous driving, reward functions can be based on e.g., speed, distance travelled without crash, distance travelled towards goal, collision damage
- Goal is to maximize the reward over time



Reinforcement Learning

Basics

Exploitation vs. Exploration:

- Agent tries to maximize reward and chooses actions that were rewarding in the past; this might not be the optimal decision
- To discover more efficient actions, new actions must be tested

➡ Balance of exploitation and exploration is needed

Learning/Training:

- Iterative learning process of RL follows a trial-and-error fashion
→ simulations are required as real-world learning is dangerous
- Vast amount of episodes is required to improve the agent's performance

Reinforcement Learning

Model-based RL and Model-free RL

■ Model-based RL

- „Model“ of the environment is learned by the agent, to predict the next state and next reward
- More data-efficient than model-free RL
- Only as good as the model learned
- Also called indirect methods

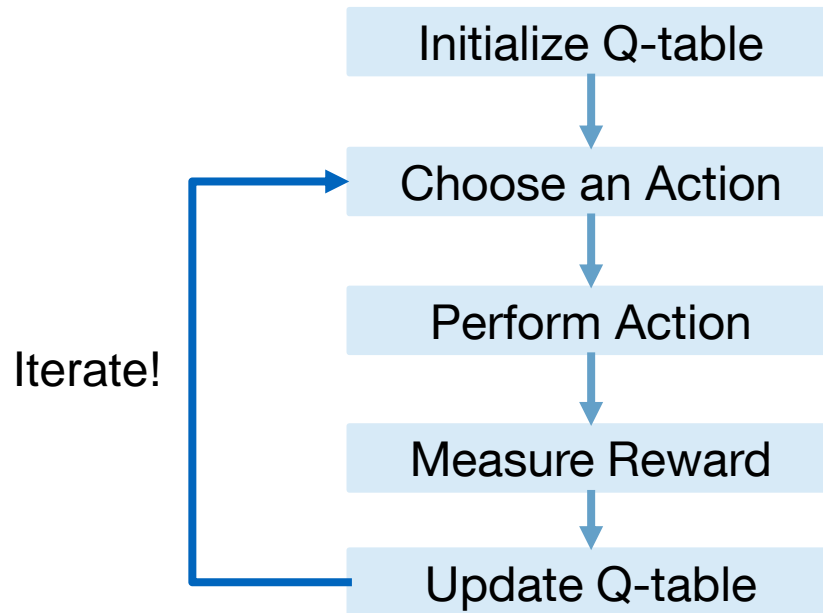
■ Model-free RL

- Sample reward and transition function by interacting with the world
- Also called direct methods
- E.g., Q-Learning, Actor-Critic, Deep Deterministic Policy Gradient (DPG)

Reinforcement Learning

Q-Learning

Q-learning is a model-free RL algorithm. It is used to learn each value of the Q-table. The Q-table helps the agent to select the best action a for each state s .



	Actions			
	↑	↓	←	→
State 1	1	3	2	0
State 2	2	2	5	1
...
State n

Example Q-table

Reinforcement Learning

Q-Learning

Basic update rule of Q-learning:

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \left[\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q(s_{t+1}, a)}_{\text{maximum expected future reward}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right]$$

The equation illustrates the Q-learning update rule. The new Q-value is calculated by adding the learning rate (α) multiplied by the difference between the target value and the old value to the old value. The target value is the sum of the reward (r_t) and the discounted maximum expected future reward (γ * max_a Q(s_{t+1}, a)).

Reinforcement Learning

Q-Learning

Policy that maximizes the reward:

$$\pi(s) = \underset{a \in A}{\operatorname{argmax}} Q(s, a)$$

- Q-table can be used to select action with highest Q-value
- In practice, Q-table is impractical ...



Image size: 66 x 200
3 channel RGB image
256 bits per pixel



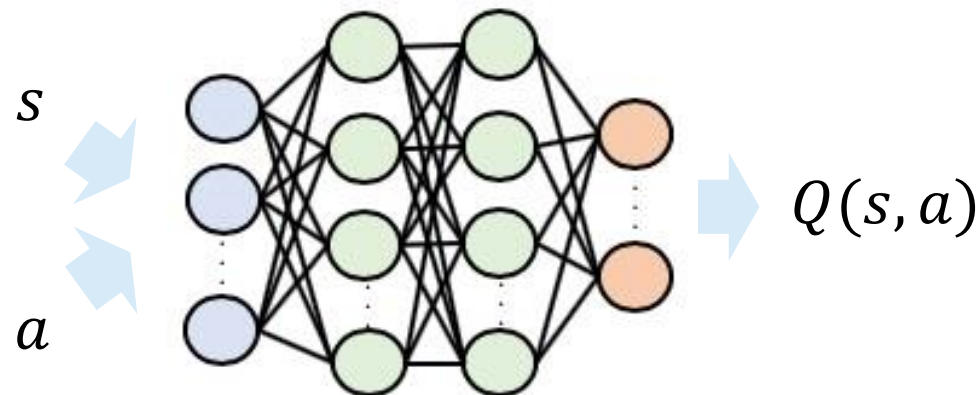
$256^{66 * 200 * 3}$
rows in the Q-table!

Reinforcement Learning

Deep Q-Learning

- Often the state space is too large to handle it with Q-tables
- In this case we use a neural network to approximate the Q-function

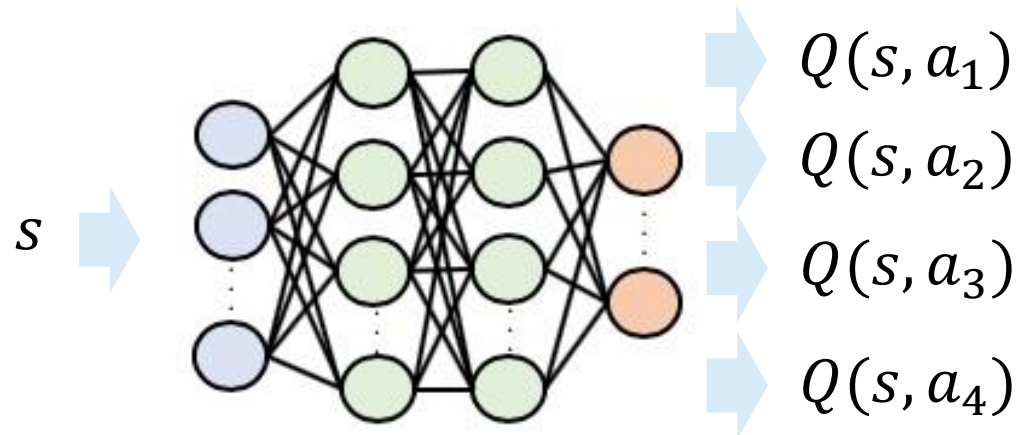
$$Q(s, a; \theta) \sim Q^*(s, a)$$



- Several network predictions for a single cost calculation, as we have multiple possible actions!

Reinforcement Learning

Deep Q-Learning



- Only state s is input of the neural network
- Network predicts Q-values for all possible actions at once
- But still the action space is very limited to discrete actions
(→ Deep Deterministic Policy Gradient)

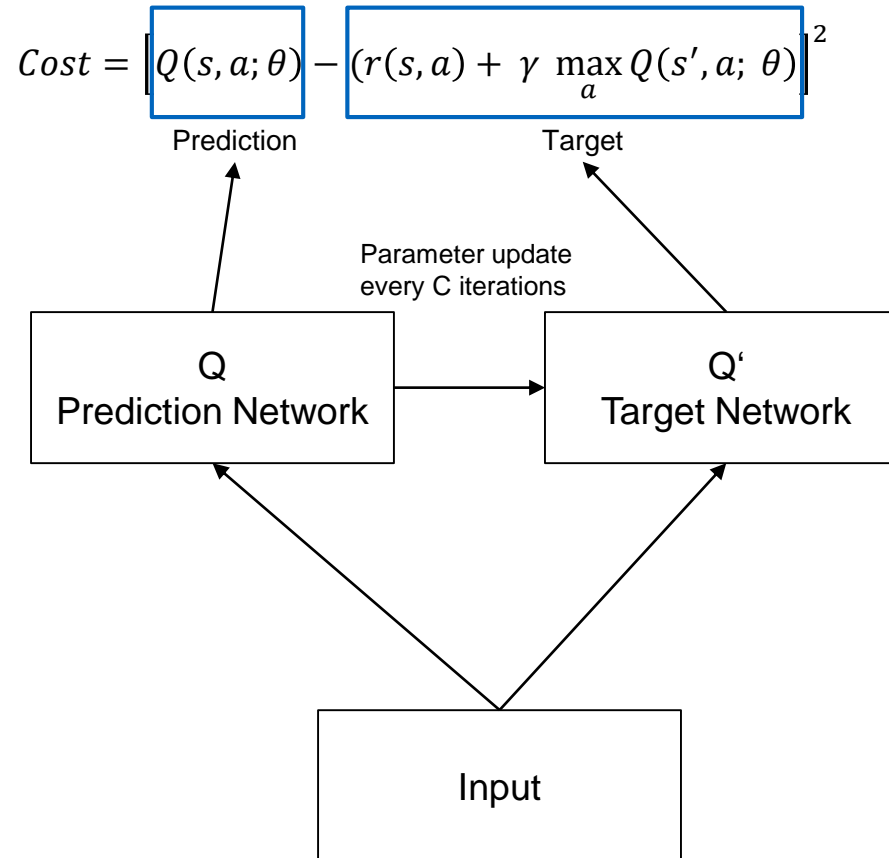
Additional Slides

Steps for training a Deep Q-Network (DQN):

1. Preprocess and feed the state s to the DQN, which will return the Q-values of all possible actions for the state
2. Select an action using the epsilon-greedy policy. With the probability epsilon, we select a random action a (exploration) and with probability $1-\text{epsilon}$, we select an action that has a maximum Q-value, such as $a = \text{argmax}(Q(s, a, \theta))$
3. Perform this action in a state s and move to a new state s' to receive a reward. This state s' is the preprocessed image of the next game screen. We store this transition in our replay buffer as $\langle s, a, r, s' \rangle$
4. Next, sample some random batches of transitions from the replay buffer and calculate the loss
5. It is known that: $Cost = \left[Q(s, a; \theta) - (r(s, a) + \gamma \max_a Q(s', a; \theta)) \right]^2$ which is just the squared difference between target Q and predicted Q
6. Perform gradient descent with respect to our actual network parameters in order to minimize this loss
7. After every C iterations, copy our actual network weights to the target network weights
8. Repeat these steps for M number of episodes

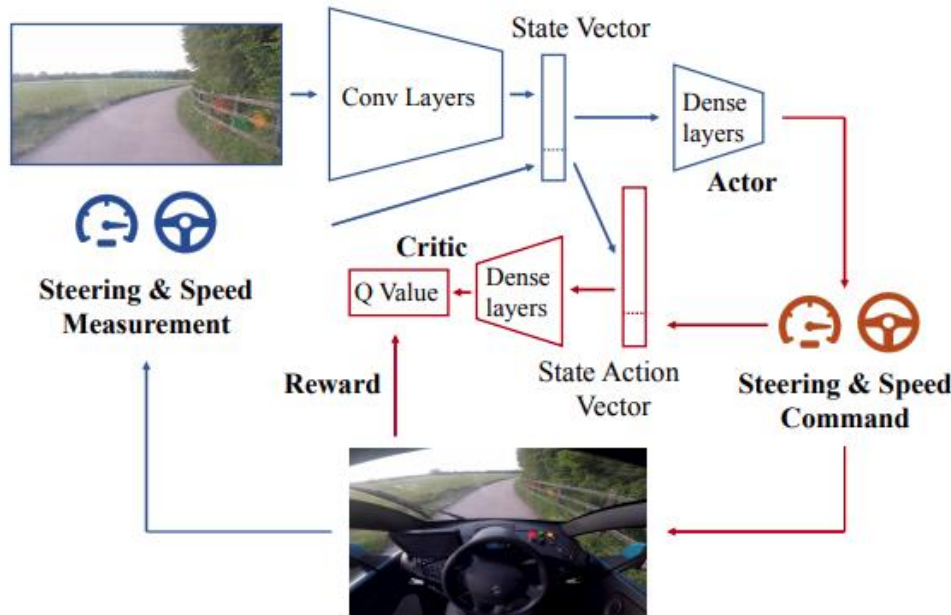
Additional Slides

To ensure a stable training there are two separate networks used to estimate the cost function. The target network has the same architecture as the function approximator but with frozen parameters. For every C iterations, the parameters from the prediction network are copied to the target network.



Additional Slides

It is not possible to straightforwardly apply Q-learning to continuous action spaces, because in continuous spaces finding the policy requires an optimization of a_t at every timestep; this optimization is too slow to be practical with large, unconstrained function approximators and nontrivial action space. Deep Deterministic Policy Gradient (DDPG) is closely connected to Q-learning, but can be applied to tasks where continuous actions spaces are needed. In addition to learning an approximator to $Q(s, a)$, it learns an approximator to $\pi(s)$. It is a actor-critic and model-free algorithm.



Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ .
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
Initialize replay buffer R .
for episode = 1, M **do**
 Initialize a random process \mathcal{N} for action exploration
 Receive initial observation state s_1
 for $t = 1, T$ **do**
 Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
 Execute action a_t and observe reward r_t and observe new state s_{t+1}
 Store transition (s_t, a_t, r_t, s_{t+1}) in R
 Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
 Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'}))|\theta^{Q'}$
 Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
 Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

Update the target networks:

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'} \end{aligned}$$

end for
end for

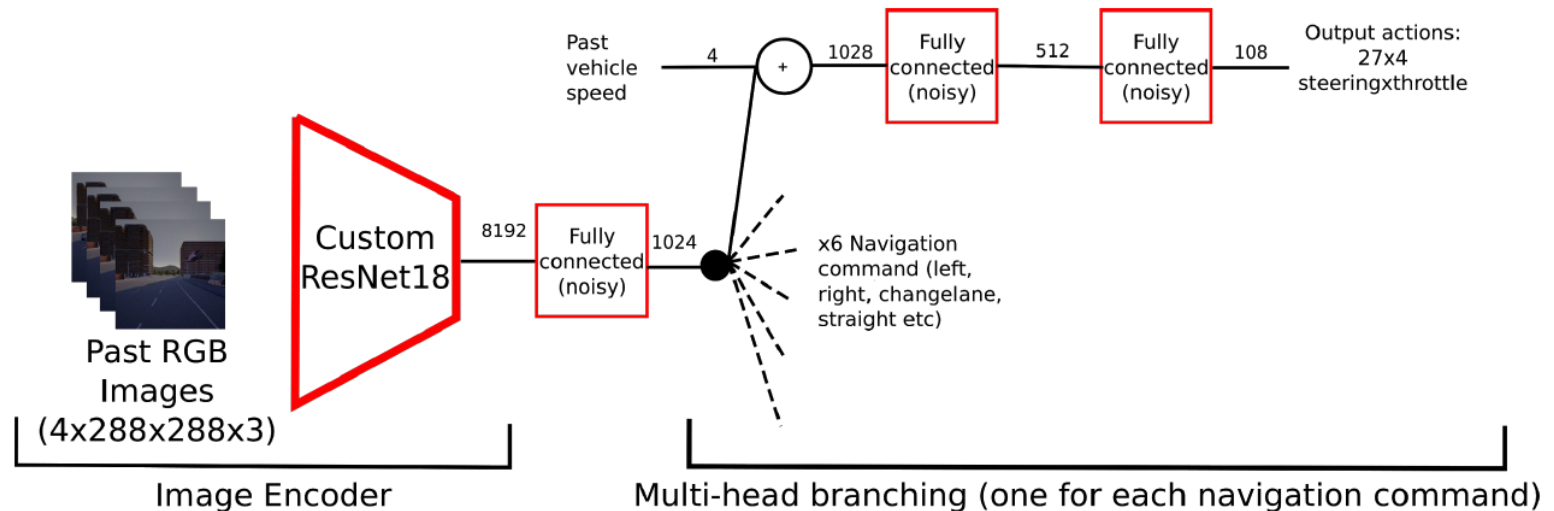
Demonstration of a DDPG algorithm in a real vehicle:

https://www.youtube.com/watch?v=eRwTbRtnT1I&ab_channel=wayve

Reinforcement Learning

Urban Driving

- In urban scenarios multiple tasks need to be handled: lane keeping, traffic light detection, pedestrian and vehicle avoidance, and handling intersections

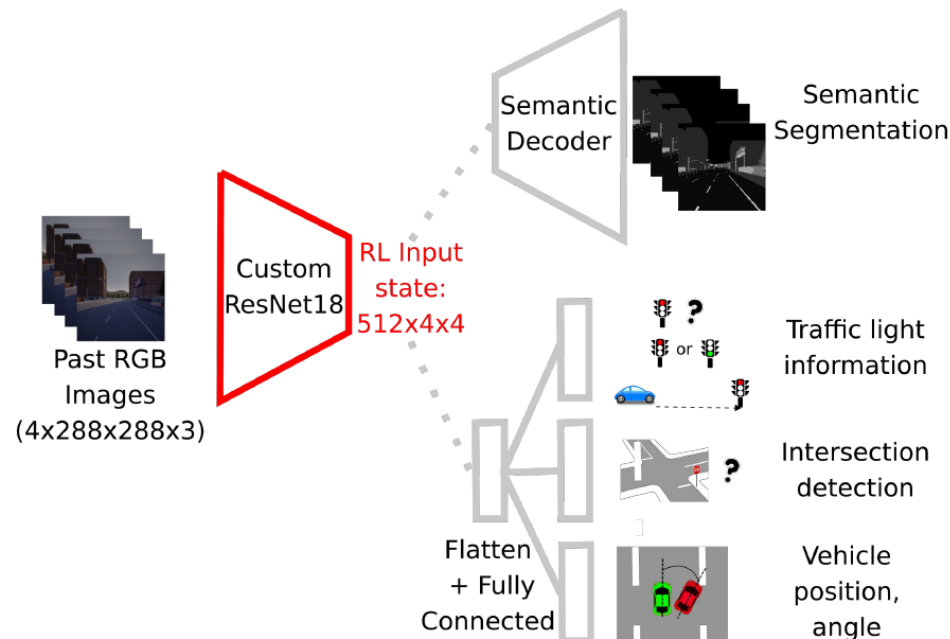


- Handling navigation commands with branched output
- Problem:** RL training signal is too weak to converge complete network

Reinforcement Learning

Urban Driving

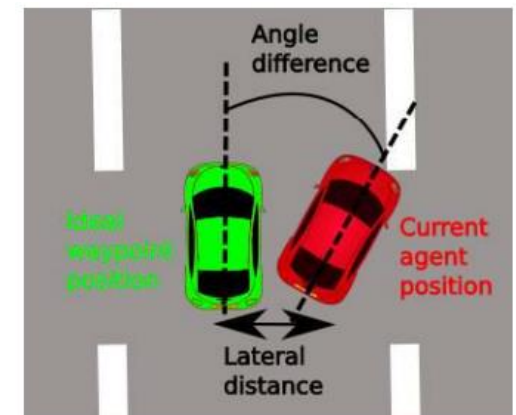
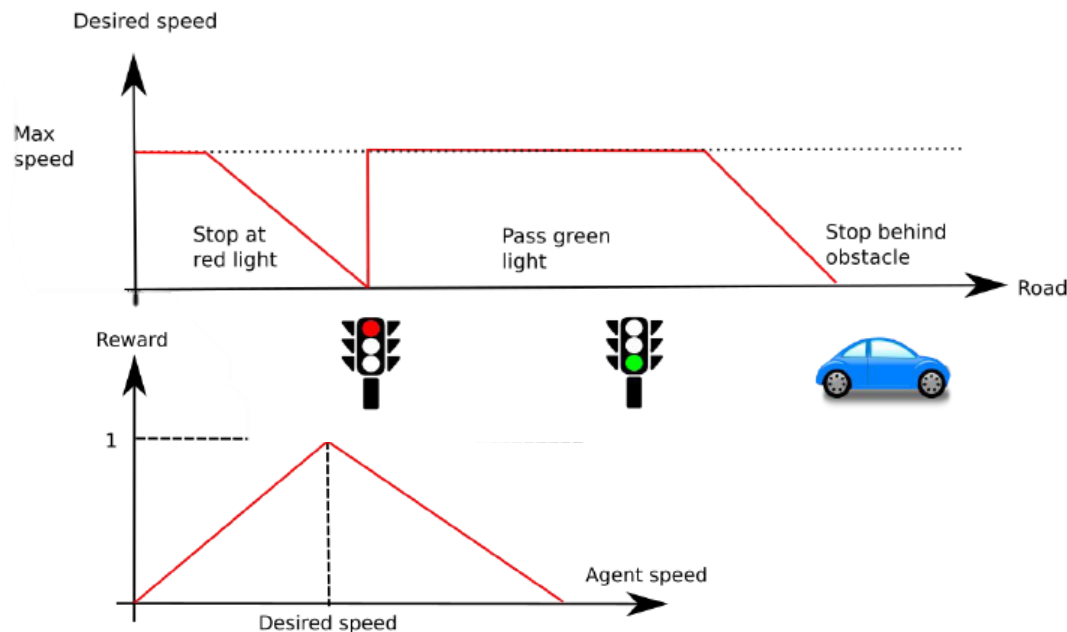
- The idea is to split the training in two phases:
 - First an encoder backbone (Resnet-18) is trained to predict affordances such as traffic light state or distance to center of the lane (new RL state).
 - The RL signal is only used to train the last part of the network.



Reinforcement Learning

Urban Driving – Reward Shaping

- The reward relies on three main components:
 - Speed
 - Position (0, when agent is exactly in the middle of the lane)
 - Rotation (inversely proportional to angle difference)



Reinforcement Learning

Urban Driving – Demonstration

End-to-End Model-Free Reinforcement
Learning for Urban Driving using
Implicit Affordances

Reinforcement Learning

Summary

- Agent performs an action based on the current state and receives a reward for the selected action. The goal is to maximize the reward by learning the optimal policy.
- Model-based RL learns a model of the environment (transition function and reward function) to predict the next state and next reward, which is used to choose the action.
- Model-free RL directly estimates the optimal action without using a model of the environment.
- Q-learning is a simple model-free RL approach that directly estimates the optimal Q-values of each action in each state, from which a policy may be derived by choosing the action with the highest Q-value in the current state.
- Deep Q-learning approximates the Q-value function with a neural network, which is beneficial in higher state spaces.

End-to-End Autonomous Driving

Prof. Dr. Markus Lienkamp

Tobias Betz, M. Sc.

Agenda

1. Introduction
2. Methods
 - 2.1 Imitation Learning
 - 2.2 Reinforcement Learning
3. **Combination of Modules**
4. Limitations
5. Summary



Combination of Modules

Motivation

- End-to-End Learning combines the entire driving task in a single neural network
- Research on the modular approach already resulted in good results for certain algorithms (e.g., segmentation, control, object detection)
→ hard to surpass by a neural network combining all modules

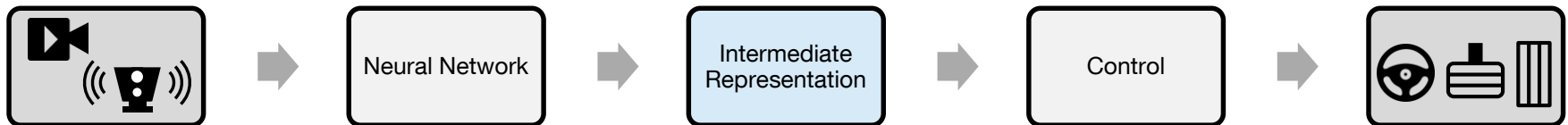
Combining the advantages of end-to-end learning and modular approach

- using conventional algorithms for certain known modules
- other modules are learned by neural network in an end-to-end fashion

Combination of Modules

Direct Perception

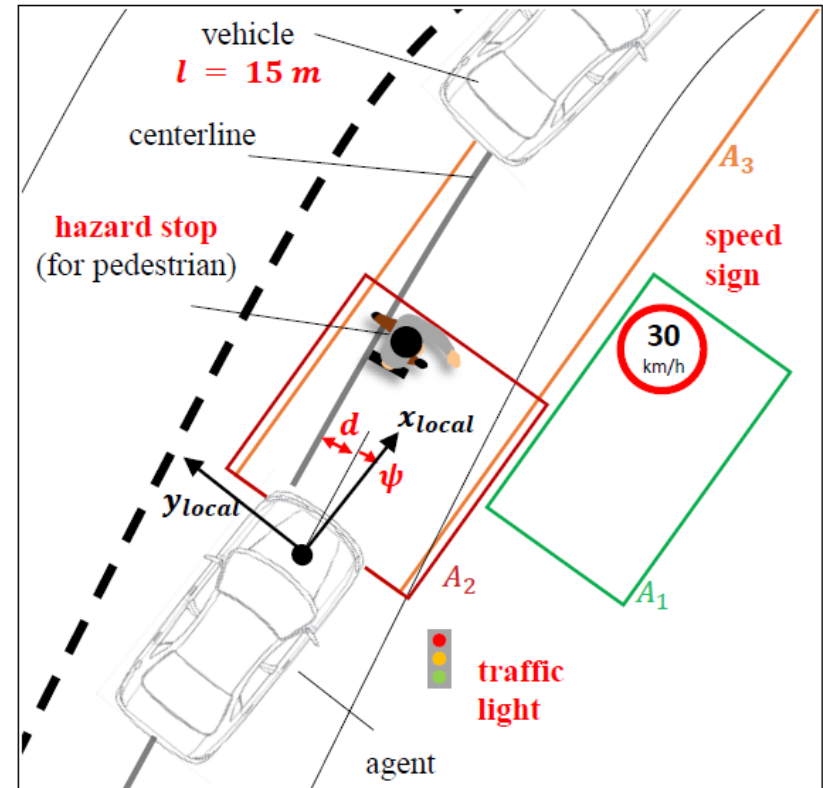
- Instead of predicting the control outputs by end-to-end learning, the neural network outputs a low-dimensional intermediate representation of the environment
- This intermediate representation is used by conventional control algorithms to maneuver the car
- Benefits:
 - Doesn't require to learn the complex sensorimotor control problem end-to-end
 - Doesn't need pixel- or box-level labels (for conventional perception algorithms)
 - Intermediate representation can be trained and validated before deployment



Combination of Modules

Direct Perception - Intermediate Representation

- Representation should be low-dimensional but still comprise all necessary information to make driving decisions
- Specific attributes of the environment limit the action space of the vehicle
- E.g.: Relative angle, distance to centerline, distance to vehicles in current or adjacent lanes, speed sign, red traffic light, hazard stop, ...



Additional Slides

Longitudinal control is subdivided into several states

- Cruising
- Following
- Over limit
- Red light
- Hazard stop

Each state is implemented by a standard controller
(e.g., cruising → PID)

Lateral control is implemented using a Stanley controller with error metrics distance to centerline and relative angle.

End-to-End Autonomous Driving

Prof. Dr. Markus Lienkamp

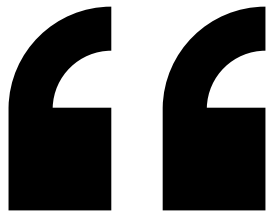
Tobias Betz, M. Sc.

Agenda

1. Introduction
2. Methods
 - 2.1 Imitation Learning
 - 2.2 Reinforcement Learning
3. Combination of Modules
4. **Limitations**
5. Summary



Limitations



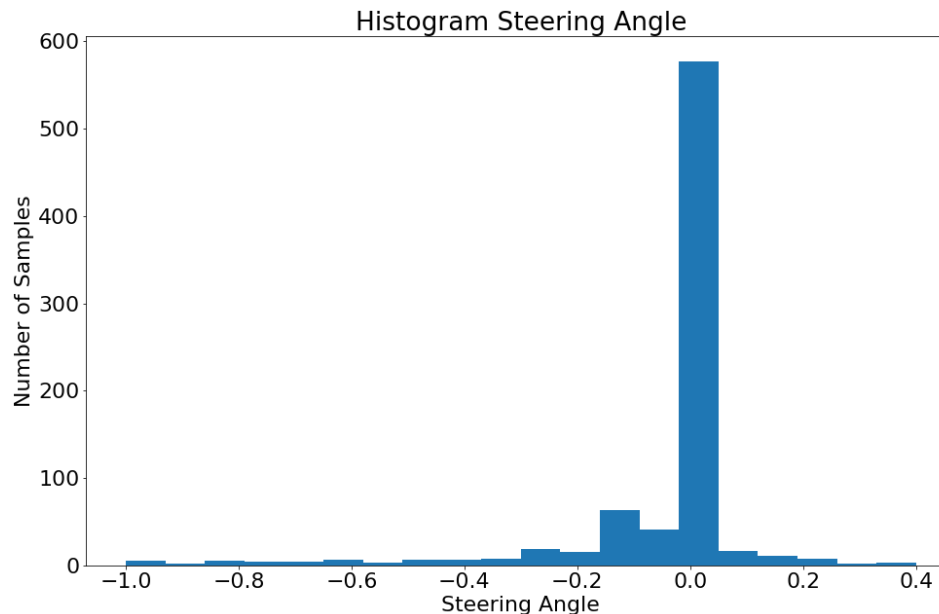
“Behavior cloning in particular has been successfully used to learn simple visuomotor policies end-to-end, but scaling to the full spectrum of driving behaviors remains an unsolved problem” (Codevilla et al. 2019)



Limitations

Dataset bias

- Behavioral cloning benefits from large datasets (e.g., collection by a large fleet), **but:** susceptible for dataset biases
 - Most of the real-world driving consists of a few simple behaviors (e.g., driving straight), difficult edge cases are underrepresented
- ➔ Performance of the network can degrade with increasing amount of data collection



Limitations

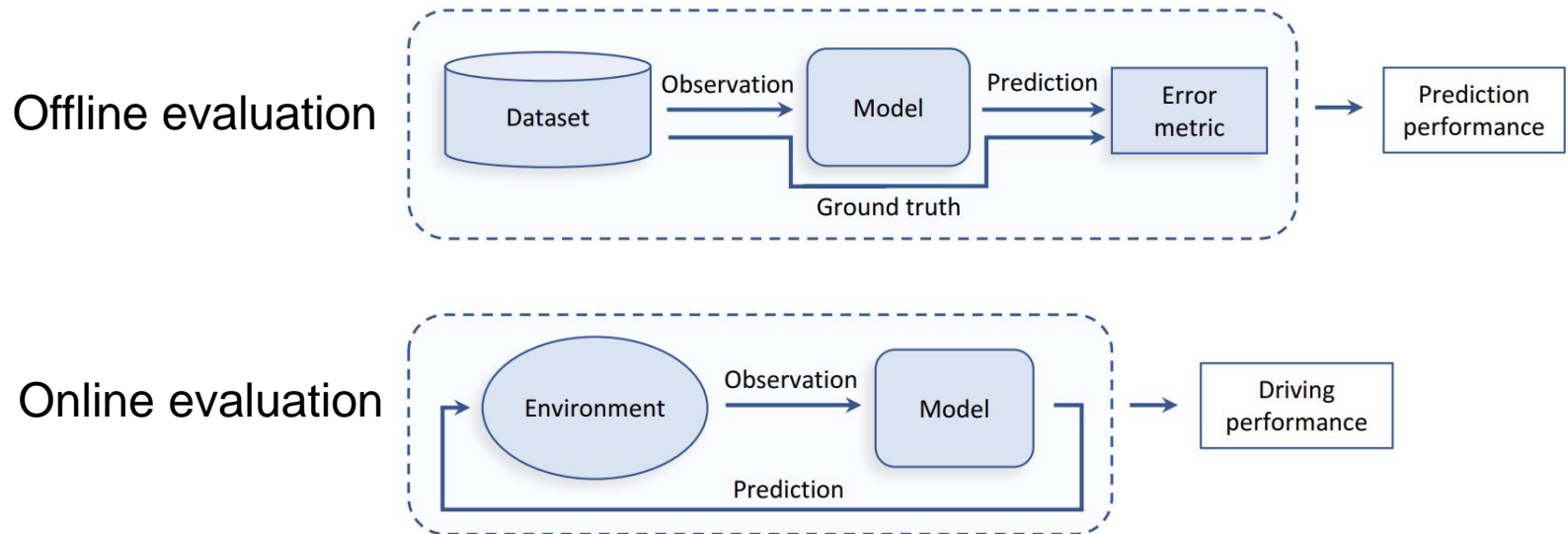
Casual Confusion

- Behavioral cloning can suffer from the inertia problem: a failure mode of casual confusion
- Occurs when the vehicle is stopped (e.g., at a red traffic light).
 - Most of the similar situations in the dataset show the car staying static
→ high probability that the car stays static
 - Network learns a false correlation between low speed and no acceleration
- This leads to a driving behavior in which the car has difficulties restarting once stopped

Limitations

Offline Evaluation Metrics Problematic

- Gap between offline evaluation metrics and the actual online driving performance



Limitations

Transferring Models from Simulation to Real-world

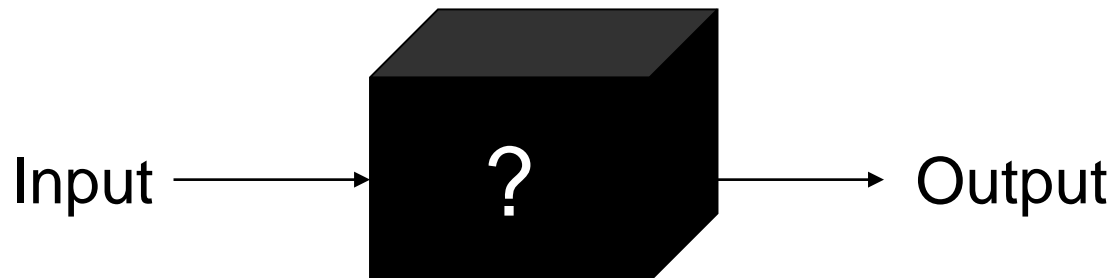
Even if the model performs well in a photorealistic simulation, transferring it to real-world is still very challenging.



Limitations

Lack of Interpretability

- Due to nature of end-to-end learning, driving task is encoded in a single model
 - Model lacks of interpretability and prevents deeper insights into the modes of operation
 - Especially important in case of failures (legal responsibility)
 - Interpretability is needed for customers to build trust in autonomous vehicles



End-to-End Autonomous Driving

Prof. Dr. Markus Lienkamp

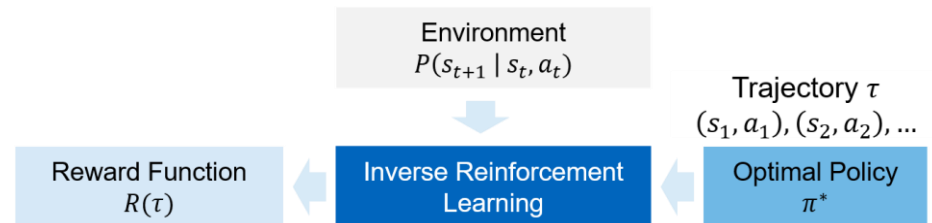
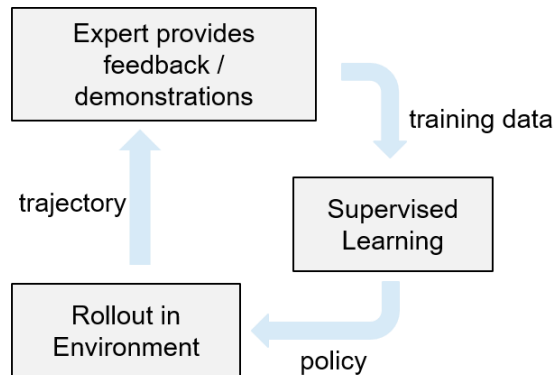
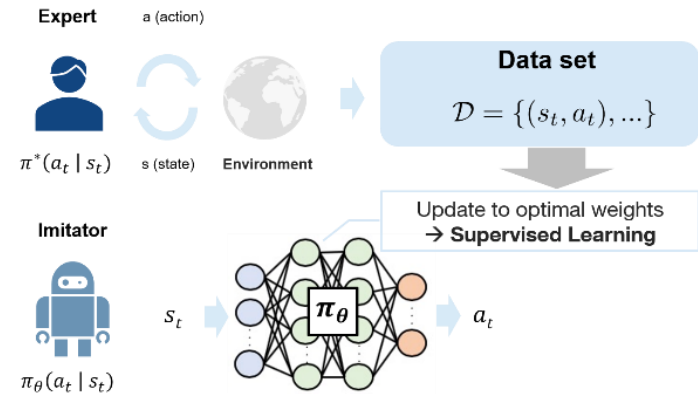
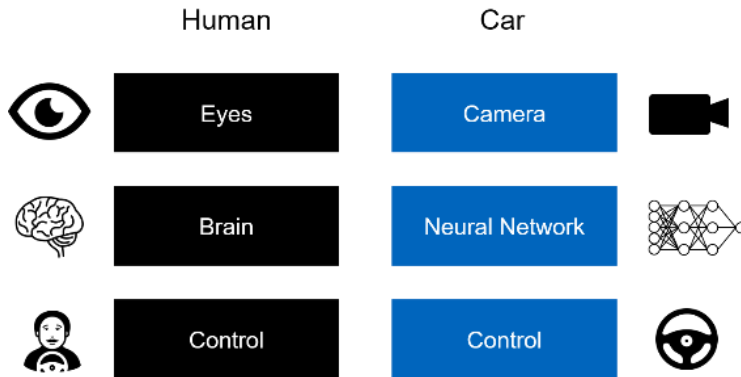
Tobias Betz, M. Sc.

Agenda

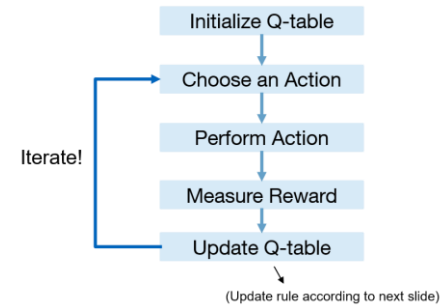
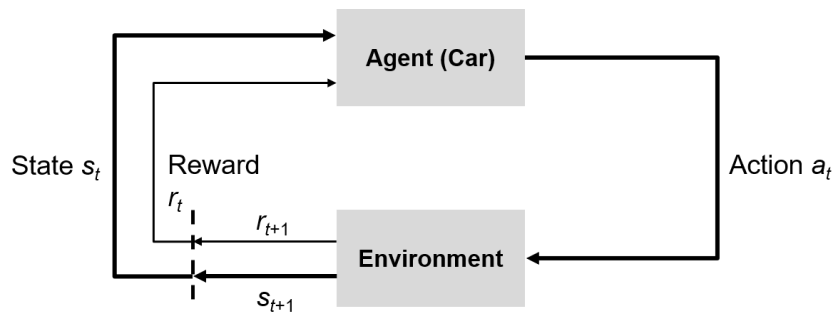
1. Introduction
2. Methods
 - 2.1 Imitation Learning
 - 2.2 Reinforcement Learning
3. Combination of Modules
4. Limitations
5. **Summary**



Summary



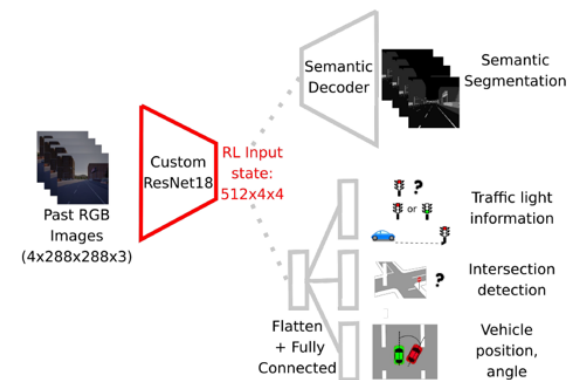
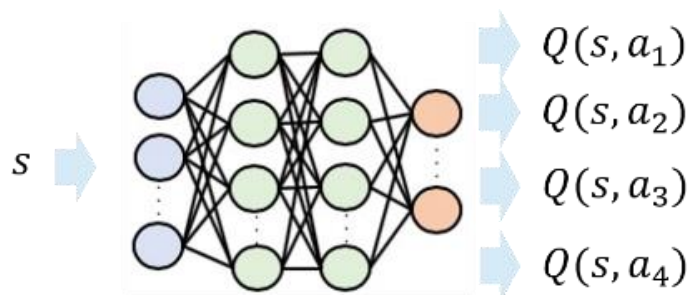
Summary



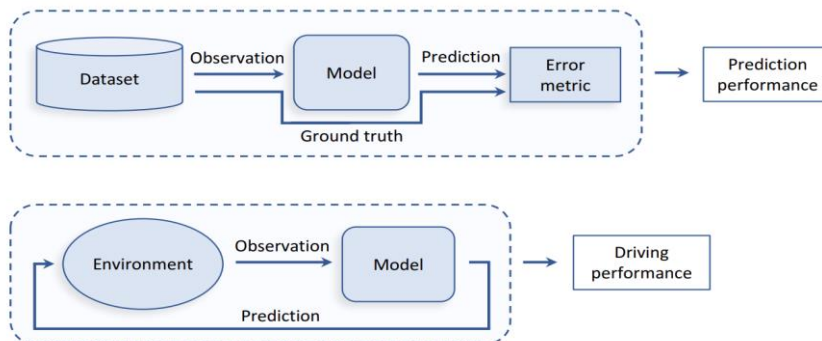
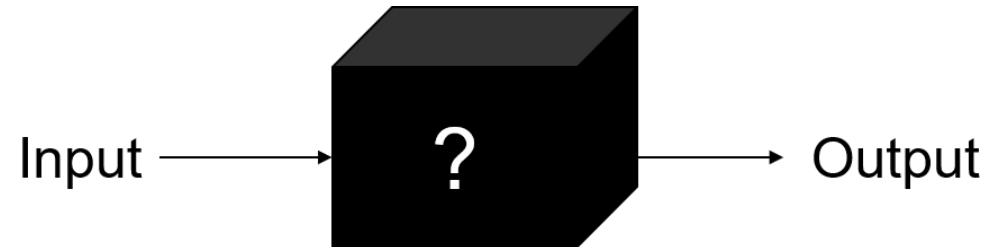
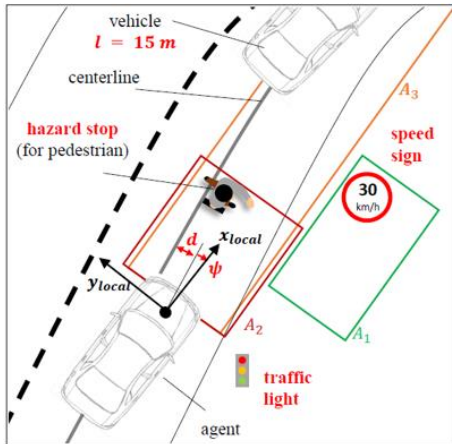
		Actions			
		↑	↓	←	→
States	State ₁	1	3	2	0
	State ₂	2	2	5	1

	State _n

Example Q-table



Summary



Summary

What did we learn today:

- **Imitation learning** has the goal to learn the **expert behavior** from demonstrations.
- Imitation learning can be categorized into three different main techniques: **behavioral cloning**, **direct policy learning** and **inverse reinforcement learning**.
- **Behavioral cloning** reduces the imitation task to a **supervised learning** problem.
- The **compounding error** is the **main problem** of **behavioral cloning**.
- The **integration of route information** can be done with different **conditional imitation learning** methods.
- In **direct policy learning** an **interactive demonstrator** provides **feedback** on the roll-out trajectory.

Summary

What did we learn today:

- The problem of **reward function designing** and the resulting core idea behind **inverse reinforcement learning**.
- The **differences** between normal **reinforcement learning** and **inverse reinforcement learning**.
- The **differences** of the discussed **imitation learning methods**.
- **Reinforcement learning** for autonomous driving follows a trial-and-error strategy and does not require labelled training data.
- Reinforcement learning can be categorized into **model-free and model-based approaches**. Model-based approaches build their own model of the environment and use this information to estimate the optimal action.
- **Q-learning** is a simple model-free RL algorithm, where the action with the largest Q-value is selected.

Summary

What did we learn today:

- For larger state spaces, **deep Q-learning** is used, where the Q-table is approximated by a neural network
- **Deep Deterministic Policy Gradient (DDPG)** is based on Q-learning and can handle continuous action spaces.
- **Combination of modules** aims to use benefits of both modular approaches and end-to-end self-driving, through a mix of using single algorithms and combination of different modules of the autonomous driving software stack.
- End-to-end self-driving has great potential, but still faces challenges today. **Limitations** include dataset bias, casual confusion, lack of interpretability, offline evaluation metrics problematic, and transferring models from simulation to real-world.

Acknowledgement

- Imitation Learning Tutorial ICML 2018
 - Yisong Yue and Hoang M. Le (Caltech)
 - <https://sites.google.com/view/icml2018-imitation-learning/>
- Imitation Learning (Stanford CS234)
 - Prof. Emma Brunskill
 - <https://web.stanford.edu/class/cs234/slides/lecture7.pdf>
- Imitation Learning (Berkeley Deep RL Course CS294-112)
 - Alina Vereshchaka
 - <http://rail.eecs.berkeley.edu/deeprlcourse-fa18/>
- Deep Reinforcement Learning – Imitation Learning
 - Hung-yi Lee (National Taiwan University)
 - https://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS18.html