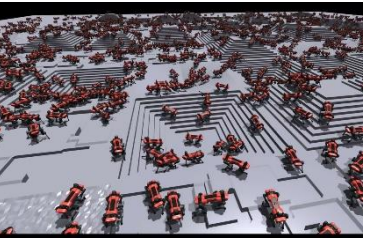
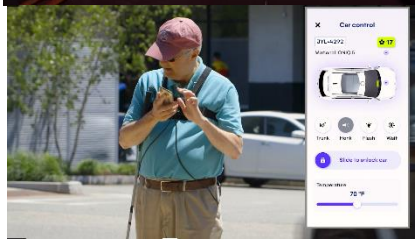
EC500: Robot Learning and Vision for Navigation



Eshed Ohn-Bar

January 30, 2023



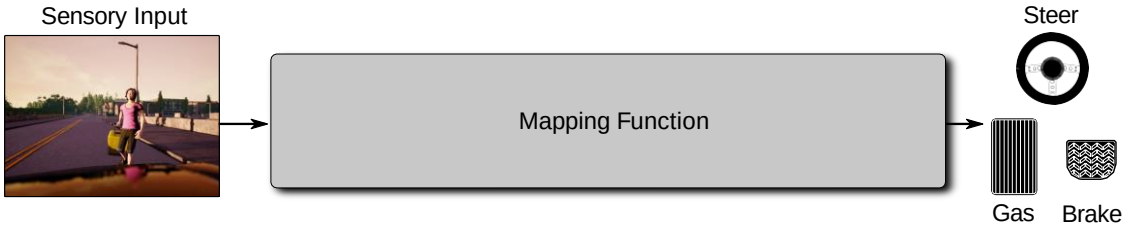
Annonuncements

- HW1 difficult, will take lots of time, plan appropriately.

- Questionnaires – those received mentioned experience with python and ML.

- Office Hours are on Mondays after class

Approaches for Sensorimotor Navigation

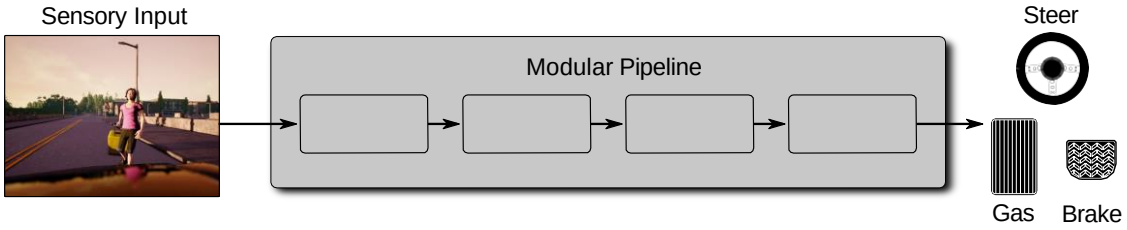


Major Paradigms:

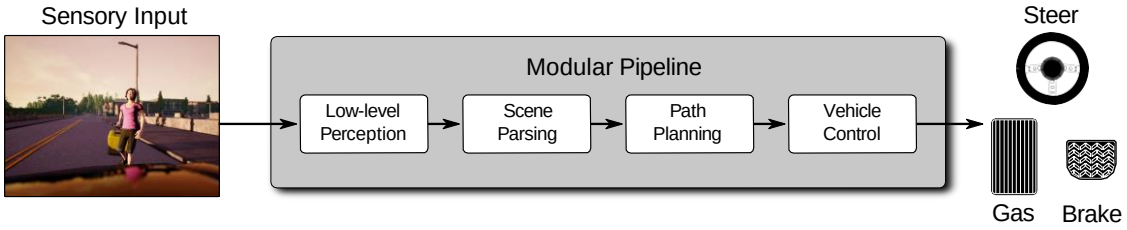
- Modular Pipelines
- End-to-End Learning (Deep Imitation Learning, Reinforcement Learning)
- Direct Perception

Modular Pipeline

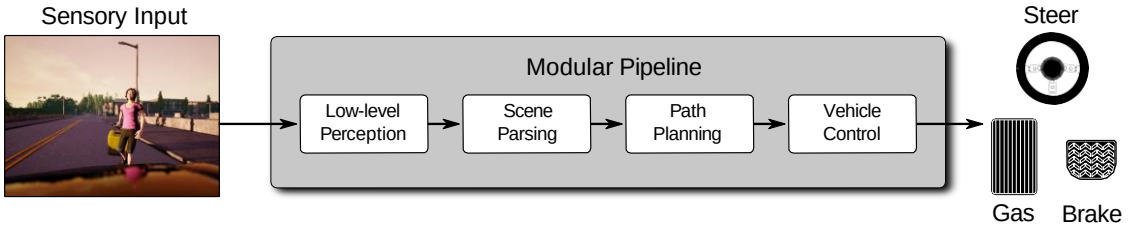
Modular Pipeline



Modular Pipeline



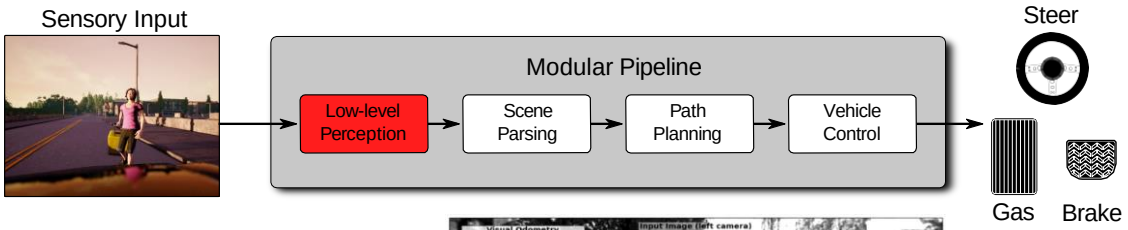
Modular Pipeline

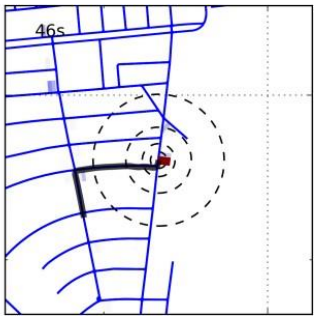


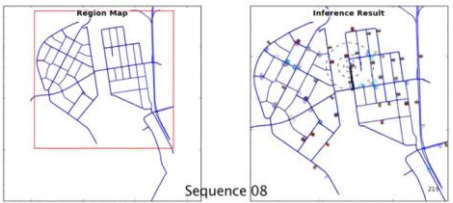
Examples:

- [Montemerlo et al., JFR 2008]
- [Urmson et al., JFR 2008]
- Waymo, Uber, Tesla, Zoox, …

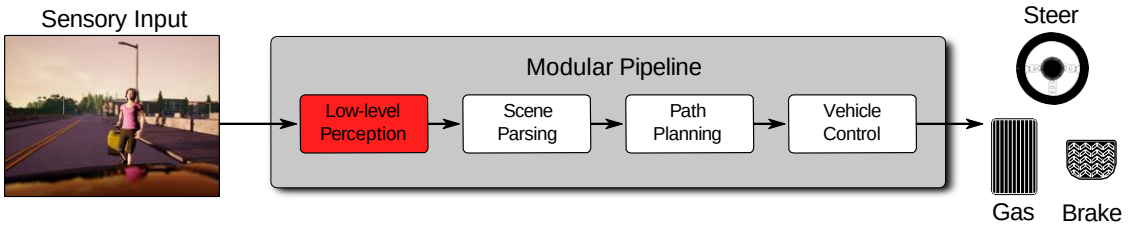
Modular Pipeline



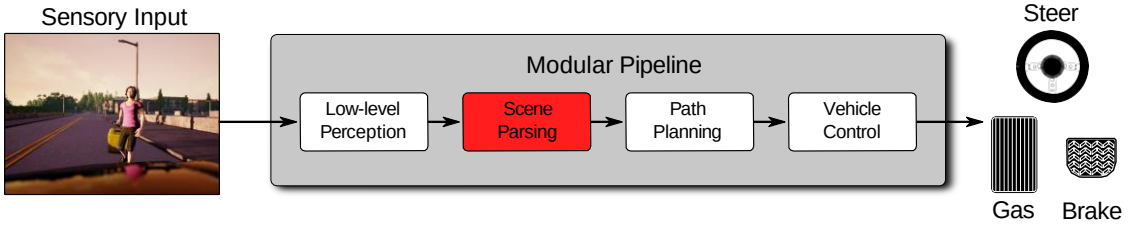


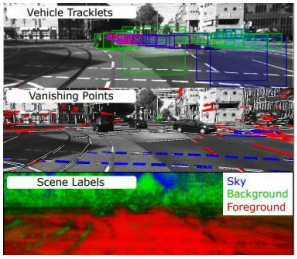


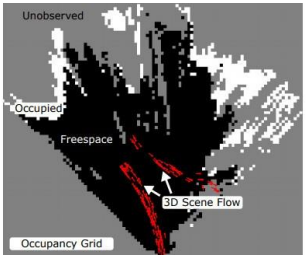
Modular Pipeline

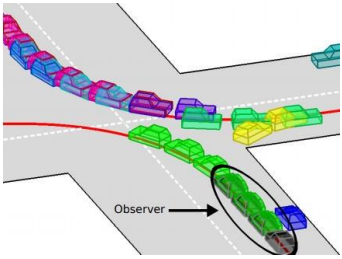


Modular Pipeline

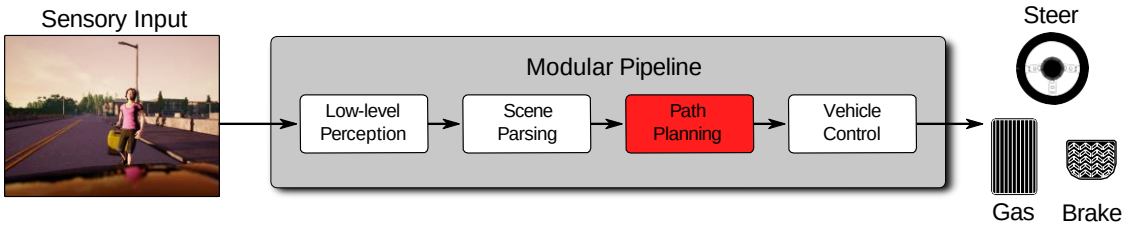


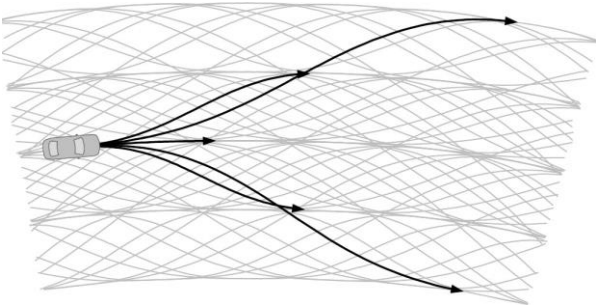




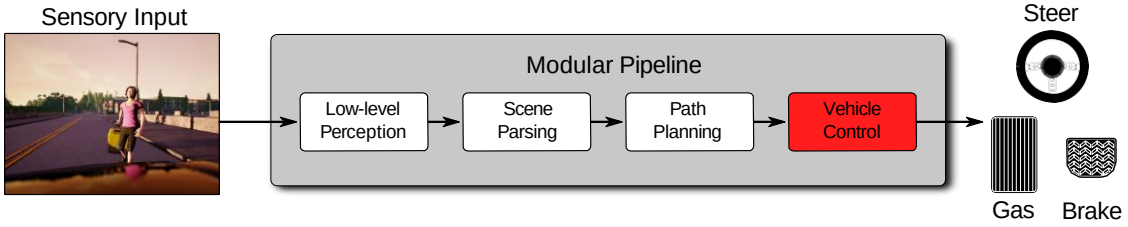


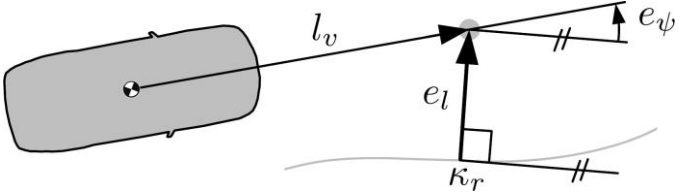
Modular Pipeline



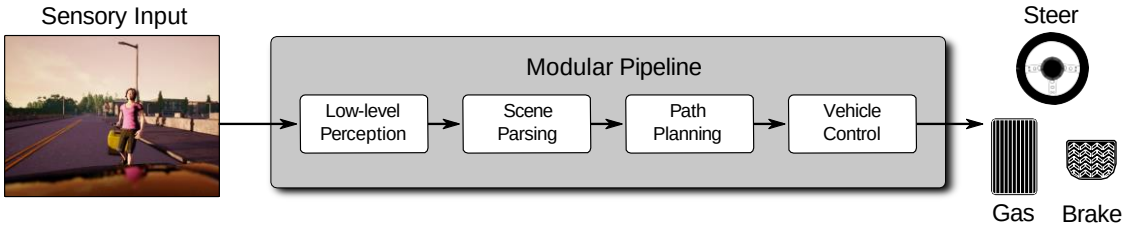


Modular Pipeline





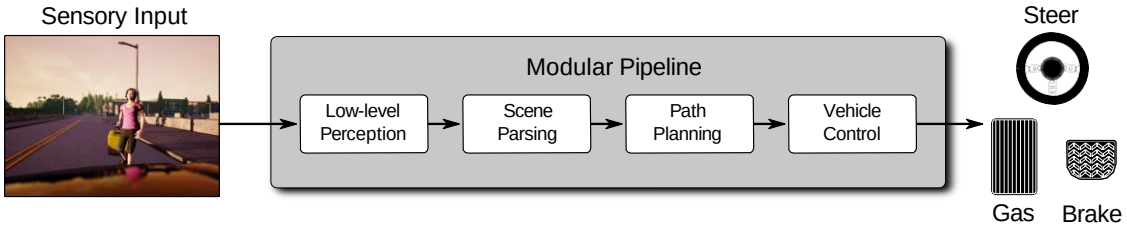
Modular Pipeline



Pros:

Cons:

Modular Pipeline

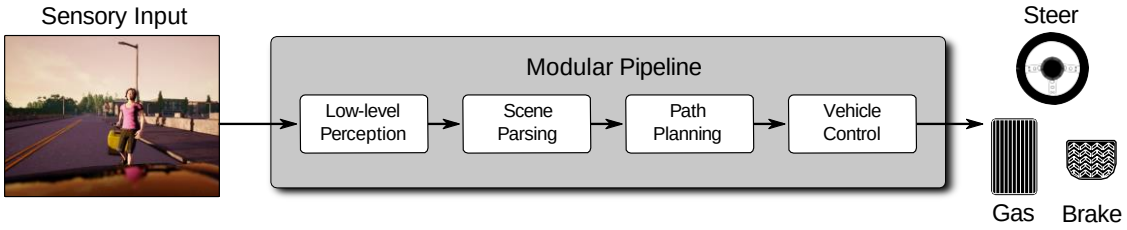


Pros:

- Small components, easy to develop in parallel

Cons:

Modular Pipeline

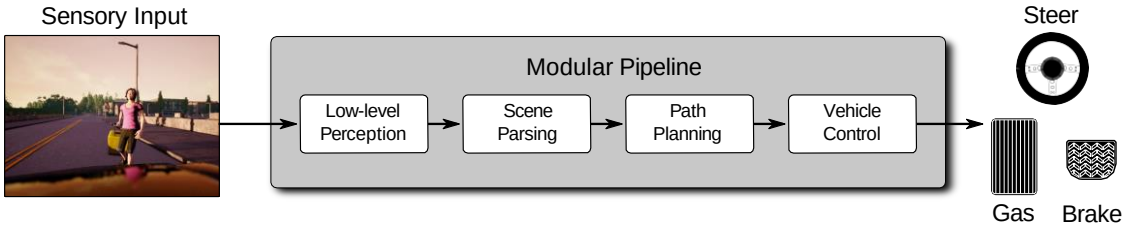


Pros:

- Small components, easy to develop in parallel
- Interpretability

Cons:

Modular Pipeline

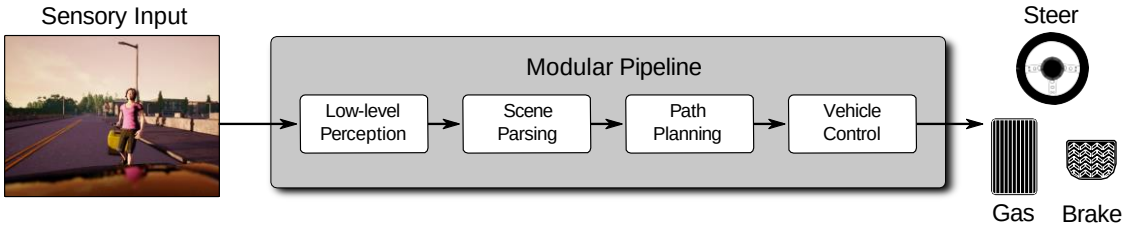


Pros:

- Small components, easy to develop in parallel
- Interpretability
- Engineered bias/prior, inductive structure

Cons:

Modular Pipeline

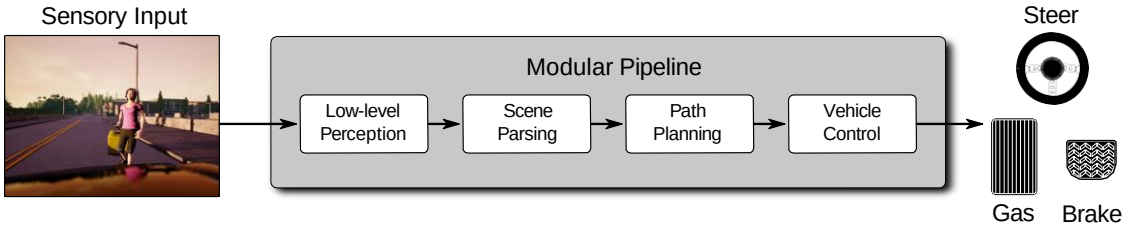


Pros:

- Small components, easy to develop in parallel
- Interpretability
- Engineered bias/prior, inductive structure

Cons:

Modular Pipeline



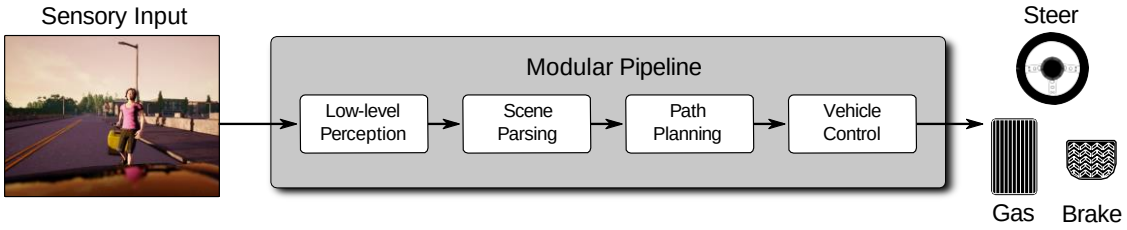
Pros:

- Small components, easy to develop in parallel
- Interpretability
- Engineered bias/prior, inductive structure

Cons:

- Piece-wise training (not jointly)

Modular Pipeline



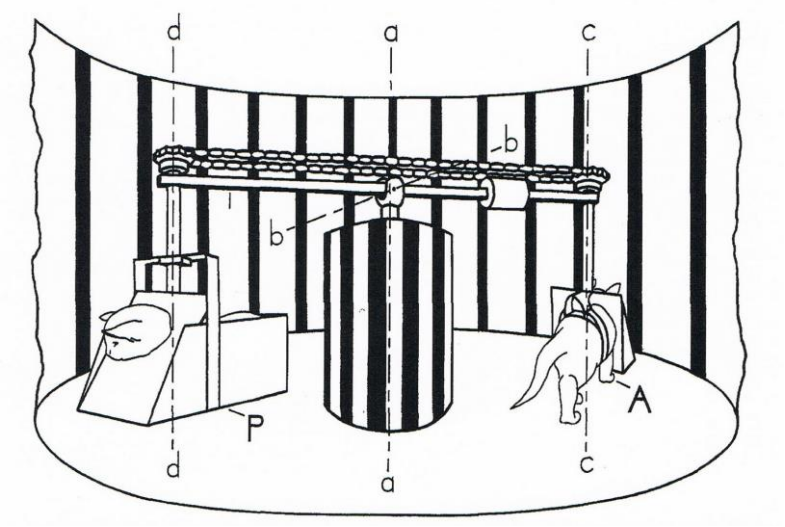
Pros:

- Small components, easy to develop in parallel
- Interpretability
- Engineered bias/prior, inductive structure

Cons:

- Piece-wise training (not jointly)
- Requires tedious and costly annotations

Experiment by Held and Hein: ***Coupling Perception and Action***



Held and Hein: Movement-produced stimulation in the development of visually guided behavior. Journal of Comparative and Physiological Psychology, 1963.

Modular Pipeline

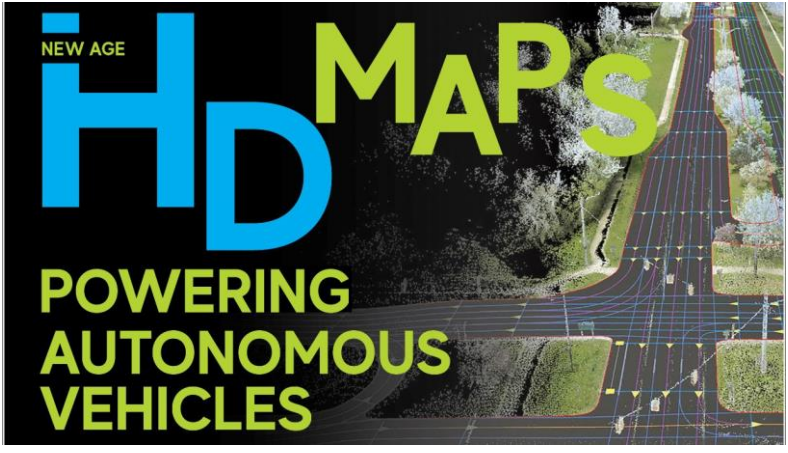
- Piece-wise training dificult: not all objects are equally important!

Ohn-Bar and Trivedi: Are All Objects Equal? Deep Spatio-Temporal Importance Prediction in Driving Videos. PR, 2017.

Modular Pipeline

Cityscapes Dataset

Modular Pipeline

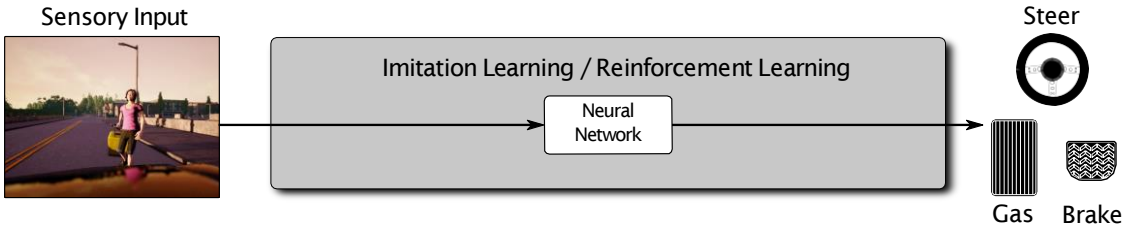


https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/

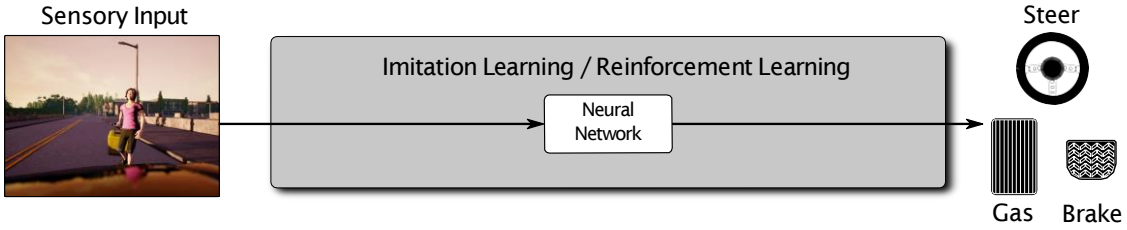
- HD Maps are expensive to create (data collection & annotation effort)

HD maps: Centimeter precision lanes, markings, traffic lights/signs, human annotated

Deep End-to-End Learning



Deep End-to-End Learning for Navigation



Examples:

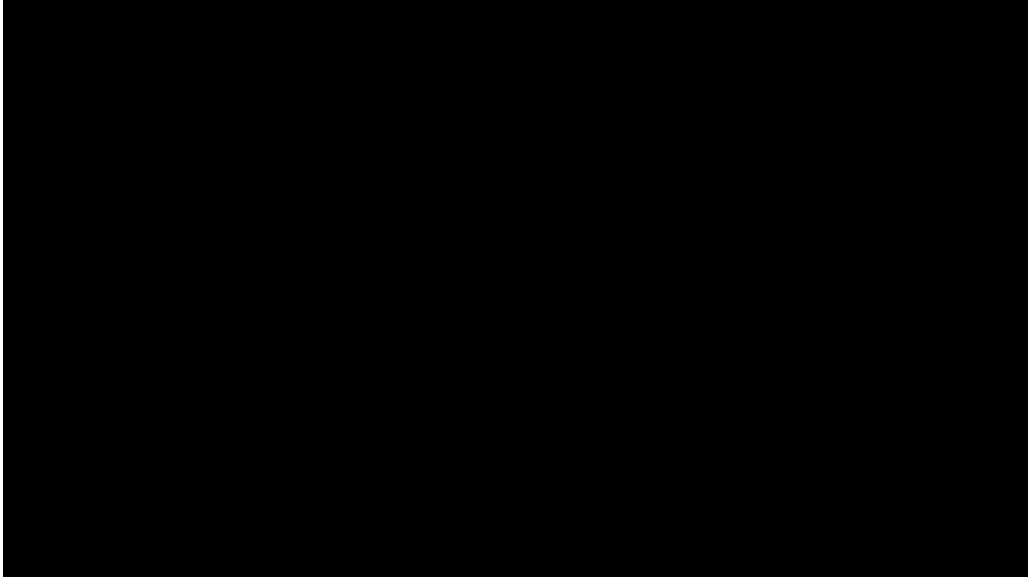
[Pomerleau, NIPS 1989]

[Bojarski, Arxiv 2016]

[Codevilla et al., ICRA 2018]

Imitation Learning


End-to-end Reinforcement learning
CARLA
Speed x3



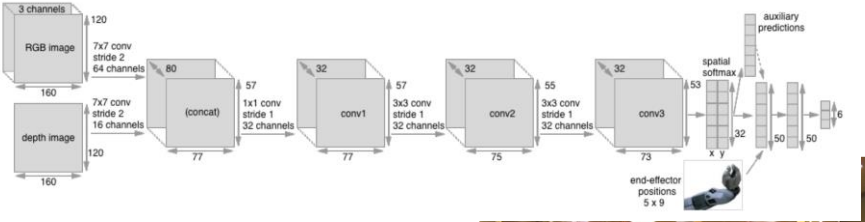
Muelling et al., Learning to select and generalize striking movements in robot table tennis, 2013

Robot Motor Skill Coordination with EM-based Reinforcement Learning

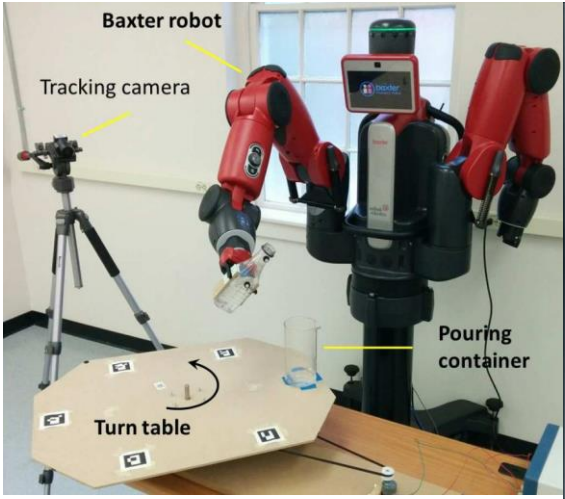
Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell

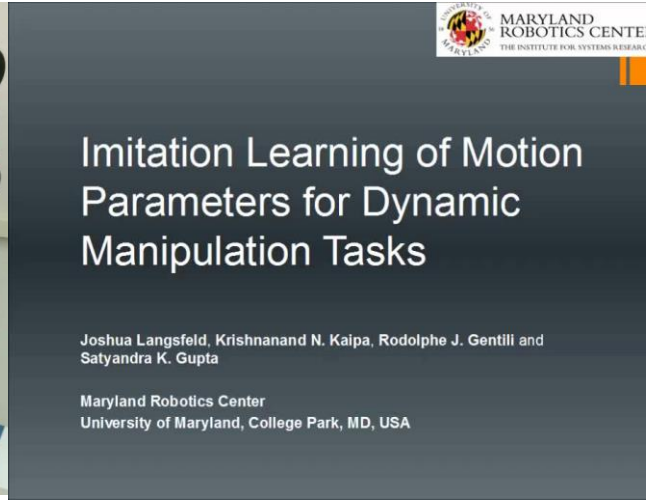
Italian Institute of Technology

Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation, ICRA 2018



Imitation Learning: Manipulation

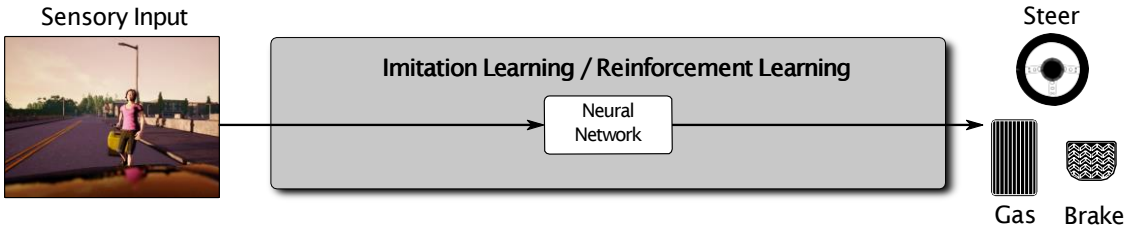




Towards Imitation Learning of Dynamic Manipulation Tasks: A Framework to Learn from Failures

ASSISTER: Assistive Navigation via Conditional Instruction Generation, ECCV 2022

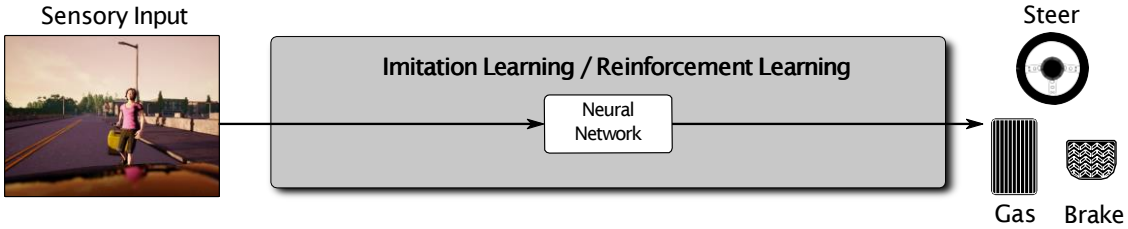
Deep End-to-End Learning for Navigation



Pros:

Cons:

Deep End-to-End Learning for Navigation

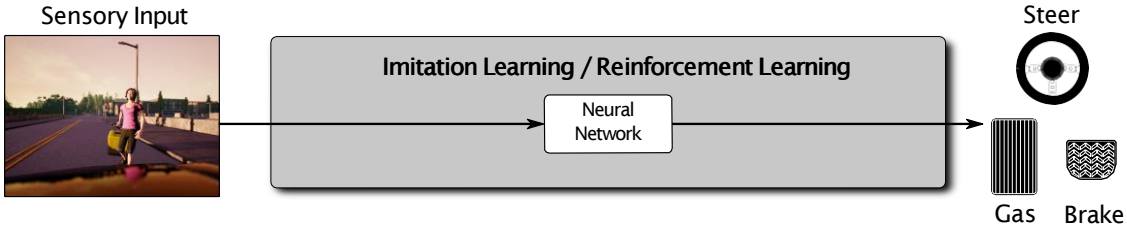


Pros:

- Task-driven training/optimization
- Cheap annotations

Cons:

Deep End-to-End Learning for Navigation



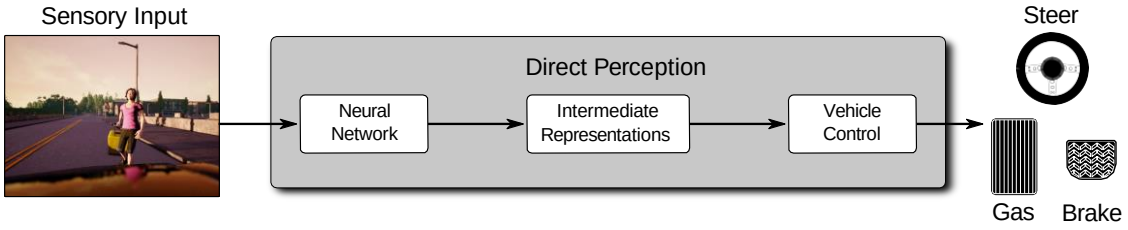
Pros:

- Task-driven training/optimization
- Cheap annotations

Cons:

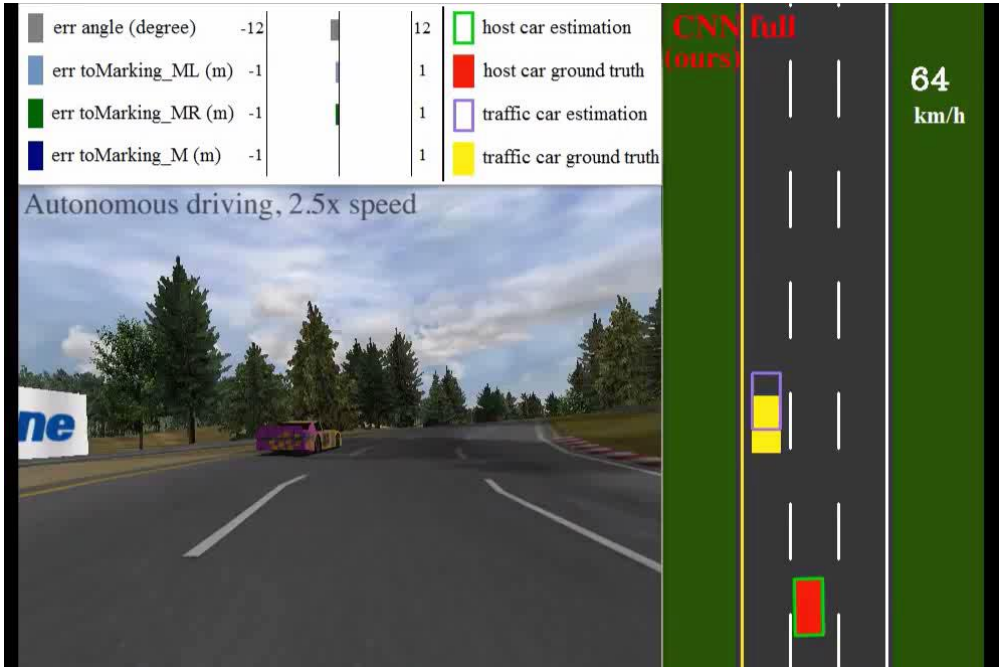
- Overfitting / Generalization
- Interpretability?

Direct Perception for Navigation



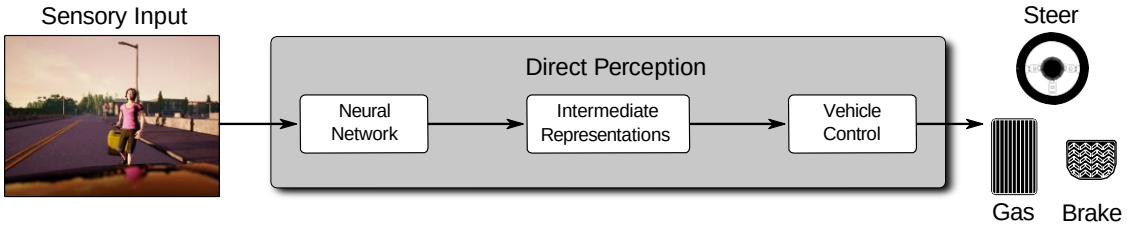
Examples:

- [Chen et al., ICCV 2015]
- [Sauer et al., CoRL 2018]
- Affordances [J. J. Gibson]


err angle (degree)
-12
12
err toMarking_ML (m)
-1
1
err toMarking_MR (m)
-1
1
err toMarking_M (m)
-1
1
host car estimation
host car ground truth
traffic car estimation
traffic car ground truth
Autonomous driving, 2.5x speed
CNN (ours)
full
64
km/h

[Sauer et al., Conditional Affordance Learning, CoRL 2018]

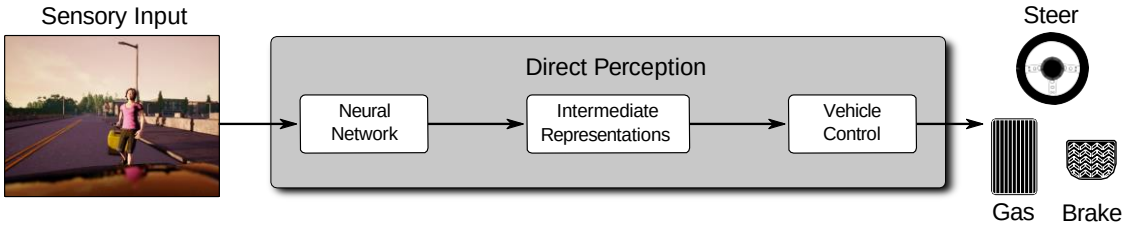
Direct Perception for Navigation



Pros:

Cons:

Direct Perception for Navigation

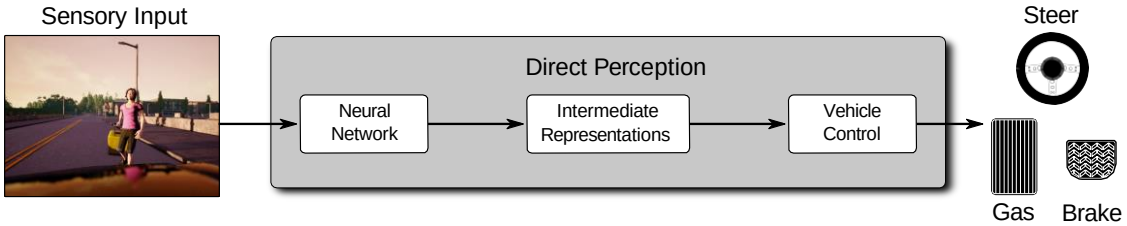


Pros:

- Compact Representation
- Interpretability

Cons:

Direct Perception for Navigation



Pros:

- Compact Representation
- Interpretability

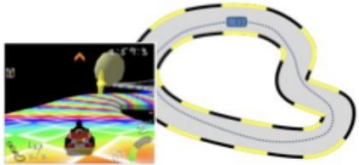
Cons:

- What if representation recognition fails?
- **How to choose representations?**

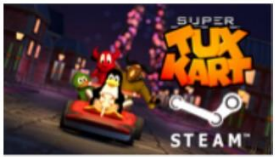
Imitation Learning

Imitation in Babies

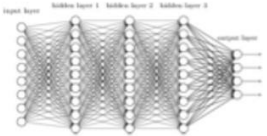
Ingredients of Imitation Learning



Demonstrations or Demonstrator

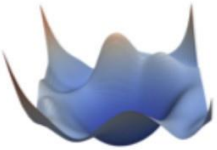


Environment / Simulator



Policy Class

Loss Function



Learning Algorithm

Formal Definition of the Imitation Learning Problem

Components:

Formal Definition of the Imitation Learning Problem

Components:

- State: $s \in \mathcal{S}$ may be partially observed (e.g., game screen)

Formal Definition of the Imitation Learning Problem

Components:

- State: $s \in \mathcal{S}$ — may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ — may be discrete or continuous (e.g., turn angle, speed)

Formal Definition of the Imitation Learning Problem

Components:

- State: $s \in \mathcal{S}$ — may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ — may be discrete or continuous (e.g., turn angle, speed)
- Policy: $\pi_\theta : \mathcal{S} \to \mathcal{A}$ — we want to learn the policy parameters θ

Formal Definition of the Imitation Learning Problem

Components:

- State: $s \in \mathcal{S}$ — may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ — may be discrete or continuous (e.g., turn angle, speed)
- Policy: $\pi_\theta : \mathcal{S} \to \mathcal{A}$ — we want to learn the policy parameters θ
- Optimal action: $a^* \in \mathcal{A}$ — provided by expert demonstrator

Formal Definition of the Imitation Learning Problem

Components:

- State: $s \in \mathcal{S}$ — may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ — may be discrete or continuous (e.g., turn angle, speed)
- Policy: $\pi_\theta : \mathcal{S} \to \mathcal{A}$ — we want to learn the policy parameters θ
- Optimal action: $a^* \in \mathcal{A}$ — provided by expert demonstrator
- Optimal policy: $\pi^* : \mathcal{S} \to \mathcal{A}$ — provided by expert demonstrator

Formal Definition of the Imitation Learning Problem

Components:

- State: $s \in \mathcal{S}$ — may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ — may be discrete or continuous (e.g., turn angle, speed)
- Policy: $\pi_\theta : \mathcal{S} \to \mathcal{A}$ — we want to learn the policy parameters θ
- Optimal action: $a^* \in \mathcal{A}$ — provided by expert demonstrator
- Optimal policy: $\pi^* : \mathcal{S} \to \mathcal{A}$ — provided by expert demonstrator
- State dynamics: $P(s_{i+1}|s_i, a_i)$ — simulator, typically not known to policy
  Often deterministic: $s_{i+1} = T(s_i, a_i)$ — deterministic mapping

Formal Definition of the Imitation Learning Problem

Components:

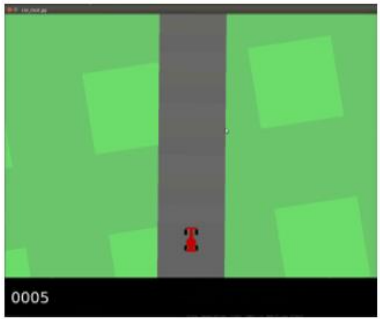
- State: $s \in \mathcal{S}$ — may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ — may be discrete or continuous (e.g., turn angle, speed)
- Policy: $\pi_\theta : \mathcal{S} \to \mathcal{A}$ — we want to learn the policy parameters θ
- Optimal action: $a^* \in \mathcal{A}$ — provided by expert demonstrator
- Optimal policy: $\pi^* : \mathcal{S} \to \mathcal{A}$ — provided by expert demonstrator
- State dynamics: $P(s_{i+1}|s_i, a_i)$ — simulator, typically not known to policy
  Often deterministic: $s_{i+1} = T(s_i, a_i)$ — deterministic mapping
- Rollout: Given s_0, sequentially execute $a_i = \pi_\theta(s_i)$ & sample $s_{i+1} \sim P(s_{i+1}|s_i, a_i)$ yields trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$

Formal Definition of the Imitation Learning Problem

Components:

- State: $s \in \mathcal{S}$ — may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ — may be discrete or continuous (e.g., turn angle, speed)
- Policy: $\pi_\theta : \mathcal{S} \to \mathcal{A}$ — we want to learn the policy parameters θ
- Optimal action: $a^* \in \mathcal{A}$ — provided by expert demonstrator
- Optimal policy: $\pi^* : \mathcal{S} \to \mathcal{A}$ — provided by expert demonstrator
- State dynamics: $P(s_{i+1}|s_i, a_i)$ — simulator, typically not known to policy
  Often deterministic: $s_{i+1} = T(s_i, a_i)$ — deterministic mapping
- Rollout: Given s_0, sequentially execute $a_i = \pi_\theta(s_i)$ & sample $s_{i+1} \sim P(s_{i+1}|s_i, a_i)$
  yields trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$
- Loss function: $\mathcal{L}(a^*, a)$ — loss of action a given optimal action a^*

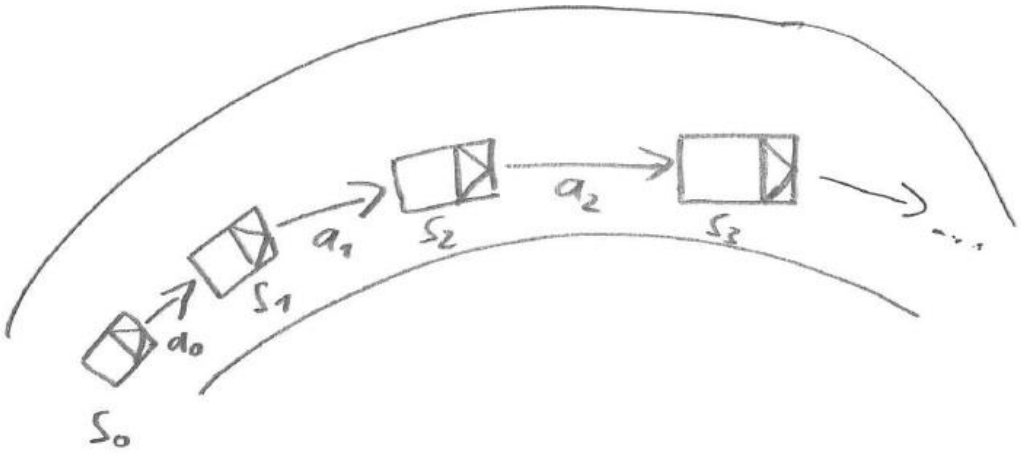
Deterministic vs. Stochastic



$$s_{i+1} = T(s_i, a_i)$$

$$s_{i+1} \sim P(s_{i+1}|s_i, a_i)$$

Rollout - Example

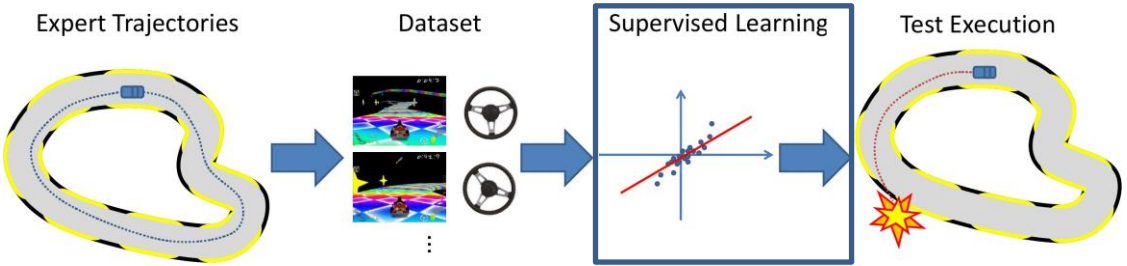


Formal Definition of the Imitation Learning Problem

Components:

- State: $s \in \mathcal{S}$ — may be partially observed (e.g., game screen)
- Action: $a \in \mathcal{A}$ — may be discrete or continuous (e.g., turn angle, speed)
- Policy: $\pi_\theta : \mathcal{S} \to \mathcal{A}$ — we want to learn the policy parameters θ
- Optimal action: $a^* \in \mathcal{A}$ — provided by expert demonstrator
- Optimal policy: $\pi^* : \mathcal{S} \to \mathcal{A}$ — provided by expert demonstrator
- State dynamics: $P(s_{i+1}|s_i, a_i)$ — simulator, typically not known to policy
  Often deterministic: $s_{i+1} = T(s_i, a_i)$ — deterministic mapping
- Rollout: Given s_0, sequentially execute $a_i = \pi_\theta(s_i)$ & sample $s_{i+1} \sim P(s_{i+1}|s_i, a_i)$
  yields trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$
- Loss function: $\mathcal{L}(a^*, a)$ — loss of action a given optimal action a^*

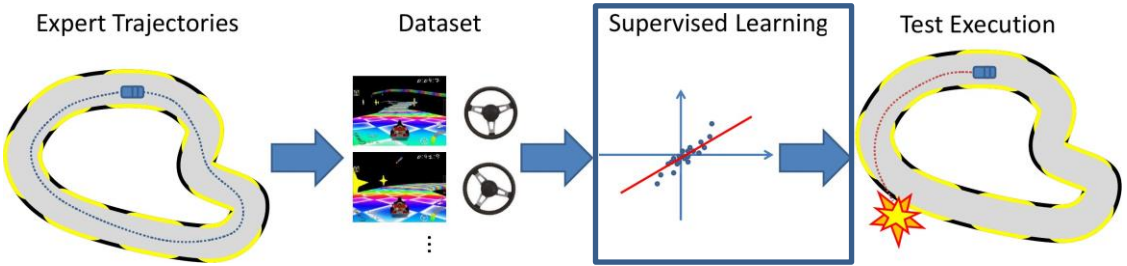
Behavior Cloning in a Nutshell



Hard coding policies is often dificult ⇒ Rather use a data-driven approach!

- **Given:** demonstrations or demonstrator
- **Goal:** train a policy to mimic decision
- **Variants:** behavior cloning (this lecture), inverse optimal control, …
- Learning is safe
- Applications in driving, flying, manipulation, teaching robots, teleoperation…

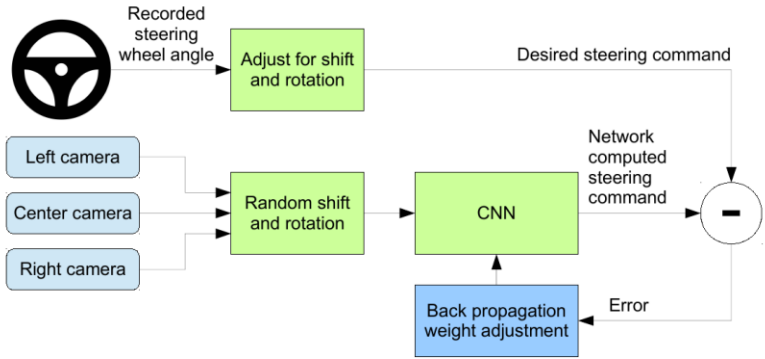
Behavior Cloning in a Nutshell



- Assumes actions in the expert trajectories are i.i.d (Independent and identically distributed)

- Train a classifier or regressor to map observations to actions at each time step of the trajectory

PilotNet: System Overview

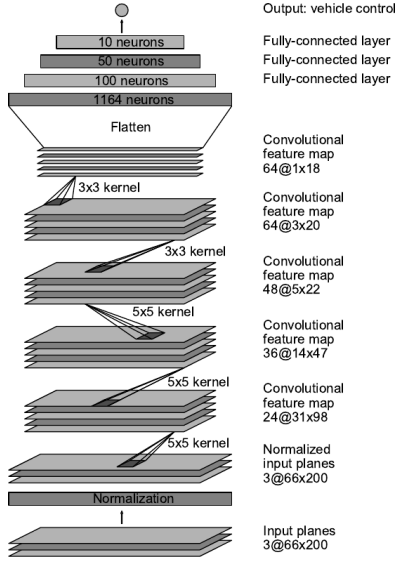


- Steering command represented as $1/r$ where r = turning radius in meters
- Data augmentation by 3 cameras and virtually shifted / rotated images assuming the world is flat (homography), adjusting the steering angle appropriately

Bojarski et al.: End-to-End Learning for Self-Driving Cars. Arxiv, 2016.

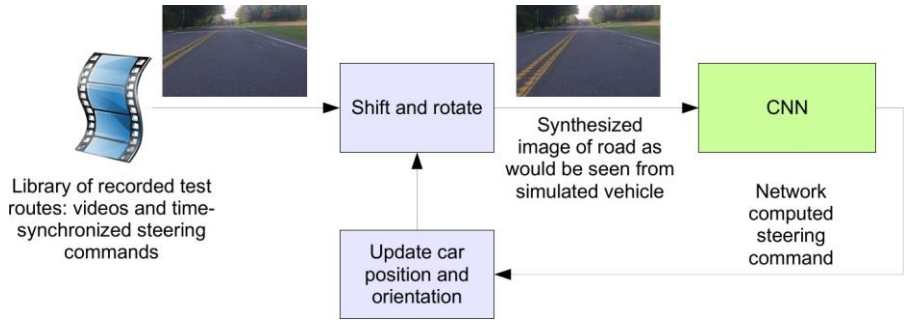
PilotNet: Architecture

- Convolutional network (250k param)
- Input: YUV image representation
- 1 Normalization layer
  - Not learned
- 5 Convolutional Layers
  - 3 strided 5x5
  - 2 non-strided 3x3
- 3 Fully connected Layers
- Output: turning radius
- No lateral control!
- Trained on 72h of driving



Bojarski et al.: End-to-End Learning for Self-Driving Cars. Arxiv, 2016.

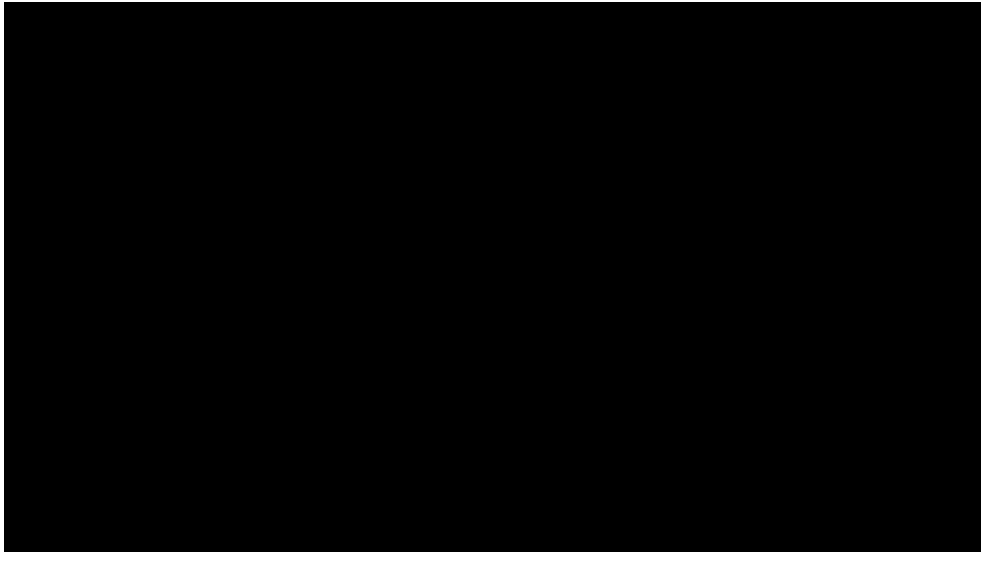
PilotNet: Simulation



- Simulator takes pre-recorded videos and extracts lane center
- Creates novel viewpoints and applies CNN commands to vehicle model
- Updates the next image based on the resulting vehicle pose

Bojarski et al.: End-to-End Learning for Self-Driving Cars. Arxiv, 2016.

PilotNet: Video



Bojarski et al.: End-to-End Learning for Self-Driving Cars. Arxiv, 2016.

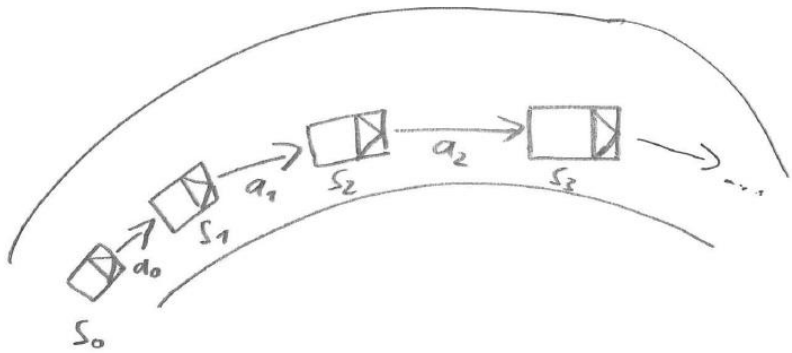
Issue of Domain/Covariate Shift

- Humans can do things that robots can’t do (and vice versa)

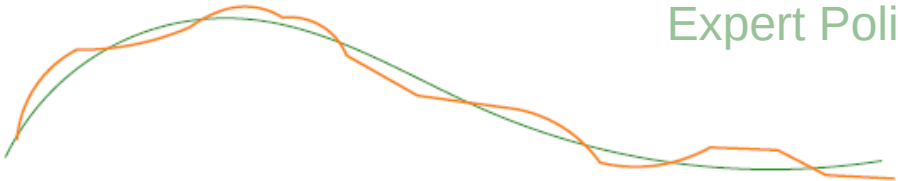
- Humans understand differently from robots

Problem: Actions are interdependent along the trajectory!

But, behavior cloning only learns/predicts one step at a time?
(that is, trained over pairs of (s_t, a_t))



Errors Independent in Time



Error at time t with probability ε

$E[\text{Total errors}] \lesssim \varepsilon T$

Compounding Errors

Neural Net Policy

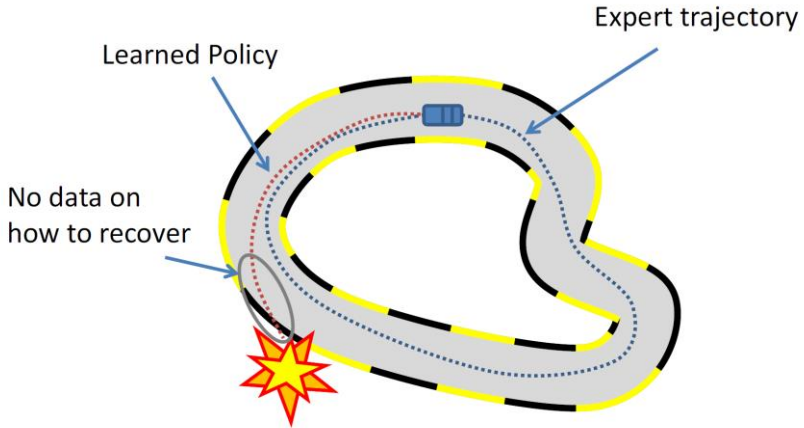
Expert Policy

error at time t with probability ε

$E[\text{Total errors}] \lesssim \varepsilon(T + (T-1) + (T-2) + \ldots + 1) \propto \varepsilon T^2$

Data Distribution Mismatch

$$P(s|\pi_\theta) \; != \; P(s|\pi^*)$$



Formal Definition of Imitation Learning

General Imitation Learning:

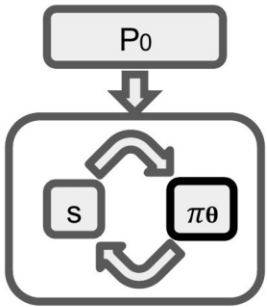
$$\underset{\theta}{\operatorname{argmin}}\ \mathbb{E}_{s\sim P(s|\pi_\theta)}\left[\mathcal{L}\left(\pi^*(s), \pi_\theta(s)\right)\right]$$

- State distribution $P(s|\pi_\theta)$ depends on rollout determined by current policy π_θ

Behavior Cloning:

$$\underset{\theta}{\operatorname{argmin}}\ \underbrace{\mathbb{E}_{(s^*,a^*)\sim P^*}\left[\mathcal{L}\left(a^*, \pi_\theta(s^*)\right)\right]}_{=\sum_{i=1}^{N}\mathcal{L}\left(a_i^*, \pi_\theta(s_i^*)\right)}$$

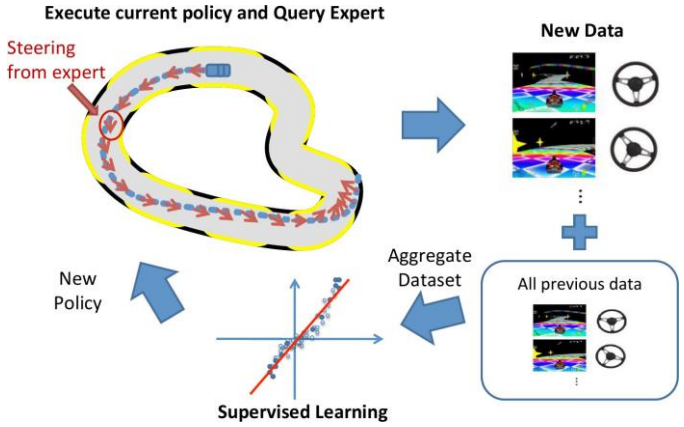
- State distribution P^* provided by expert
- Reduces to supervised learning problem



Challenges of Behavior Cloning

- Behavior cloning reasons only about immediate next step
- Behavior cloning makes IID assumption
  - Next state is sampled from states observed during expert demonstration
  - Thus, next state is sampled independently from action predicted by current policy
- What if π_θ makes a mistake?
  - Enters new states that haven't been observed before
  - New states not sampled from same (expert) distribution anymore
  - Cannot recover, catastrophic failure in the worst case
- What can we do to overcome this train/test distribution mismatch?

DAGGER: Dataset Aggregation



Idea: Iteratively build a set of inputs that the final policy is likely to encounter based on previous experience. Query expert for aggregate dataset.

Ross, Gordon and Bagnell: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. AISTATS, 2011.

DAGGER: Dataset Aggregation

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

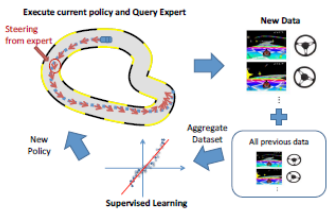
- Let T be the number of rollout time steps and ϵ be the probability of a mistake
- Behavior cloning: $O(T^2\epsilon)$ regret, DAGGER: no regret (in suitable conditions)
  - Regret = loss of current policy - loss of perfect policy
- But requires to flexibly run the environment and query the expert during training

Ross, Gordon and Bagnell: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. AISTATS, 2011.

DAGGER: In Simulation

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by labelling additional data points resulting from applying the current policy

1. train $\pi_\theta(u_t|o_t)$ from human data $\mathcal{D}_{\pi^*} = \{o_1, u_1, ..., o_N, u_N\}$
2. run $\pi_\theta(u_t|o_t)$ to get dataset $\mathcal{D}_\pi = \{o_1, ..., o_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions u_t
4. Aggregate: $\mathcal{D}_{\pi^*} \leftarrow \mathcal{D}_{\pi^*} \cup \mathcal{D}_\pi$
5. GOTO step 1.



Problems:

- execute an unsafe/partially trained policy
- repeatedly query the expert

Control Notation

o - observation

u - action

Ross, Gordon and Bagnell: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. AISTATS, 2011.

DAGGER: In Real Platform

Application on drones: given RGB from the drone camera predict steering angles

Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013

Fig. 2. One frame from MAV camera stream. The white line indicates the current yaw commanded by the current DAgger policy π_{n-1} while the red line indicates the expert commanded yaw. In this frame DAgger is wrongly heading for the tree in the middle while the expert is providing the correct yaw command to go to the right instead. These expert controls are recorded for training later iterations but not executed in the current run.

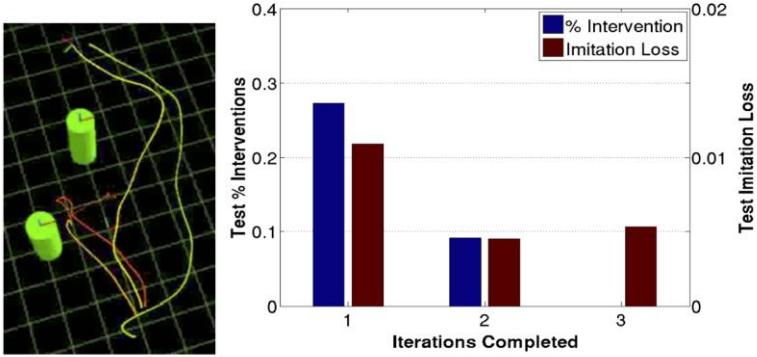
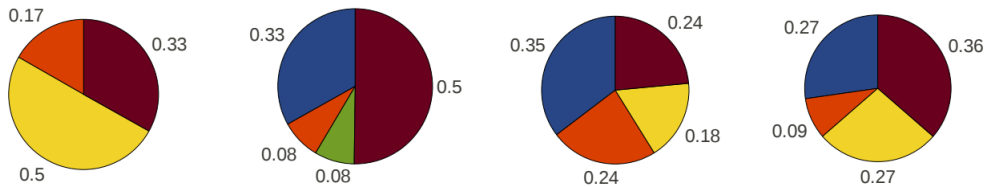


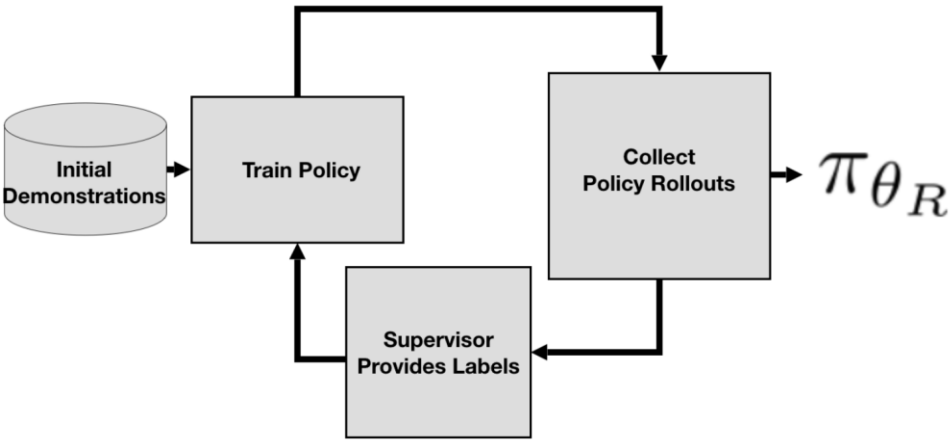
Fig. 4. Left: Improvement of trajectory by DAgger over the iterations. The rightmost green trajectory is the pilot demonstration. The short trajectories in red & orange show the controller learnt in the 1^{st} and 2^{nd} iterations which fail. The 3^{rd} iteration controller succesfully avoids both obstacles and is similar to the demonstrated trajectory. Right: Percentage of scenarios the pilot had to intervene and the imitation loss (average squared error in controls of controller to human expert on hold-out data) after each iteration of Dagger. After 3 iterations, there was no need for the pilot to intervene and the UAV could successfully avoid all obstacles



(a) Pilot (b) After iter. 1 (c) After iter. 2 (d) After iter. 3

Fig. 8. Percentage of failures of each type for DAgger over the iterations of training in the high-density region. Blue: Large Trees, Orange: Thin Trees, Yellow: Leaves and Branches, Green: Other obstacles (poles, signs, etc.), Red: Too Narrow FOV. Clearly, a majority of the crashes happen due to a too narrow FOV and obstacles which are hard to perceive, such as branches and leaves.

Reducing Shift with On-Policy Methods



The DAgger Algorithm.

DAGGER: In Real Platform

Safe? Executing an imperfect policy.

Difficult for the expert
(especially for drone application, no feedback)

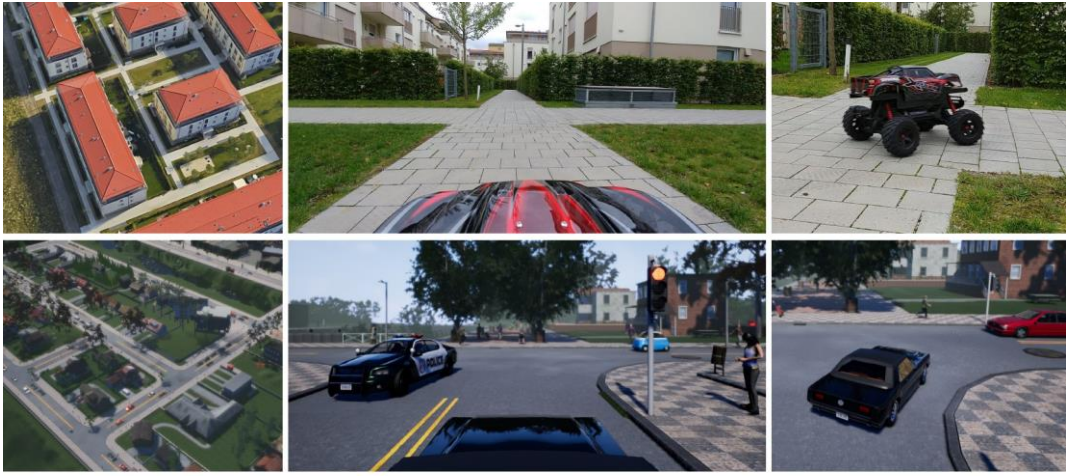
Expert's reaction time to the drone's behavior is large, this causes imperfect actions to be commanded.

Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013

When else does
behavior cloning fail?

Conditional Imitation Learning

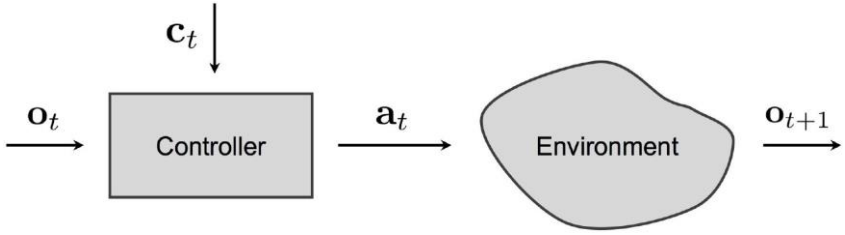
Conditional Imitation Learning



(a) Aerial view of test environment (b) Vision-based driving, view from onboard camera (c) Side view of vehicle

Codevilla, Mu ̈ller, Lo ́pez, Koltun and Dosovitskiy: End-to-End Driving Via Conditional Imitation Learning. ICRA, 2018.

Conditional Imitation Learning



Idea:

- Condition controller on **navigation command** $c \in \{left, right, straight\}$
- High-level navigation command can be provided by consumer GPS, i.e., telling the vehicle to **turn left/right** or go **straight** at the next intersection
- This removes the task ambiguity induced by the environment (not: noise)
- Observation o_t: current image Action a_t: steering angle & acceleration

Codevilla, Müller, López, Koltun and Dosovitskiy: End-to-End Driving Via Conditional Imitation Learning. ICRA, 2018.

Comparison to Behavior Cloning

BC

- Training Set:

$$\mathcal{D} = \{(a_i^*, s_i^*)_{i=1}^N\}$$

- Objective:

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(a_i^*, \pi_\theta(s_i^*)\right)$$

- Assumption:

$$\exists f(\cdot) : a_i = f(s_i)$$

Often violated in practice! Why?

Conditional BC

- Training Set:

$$\mathcal{D} = \{(a_i^*, s_i^*, c_i^*)_{i=1}^N\}$$

- Objective:

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(a_i^*, \pi_\theta(s_i^*, c_i^*)\right)$$

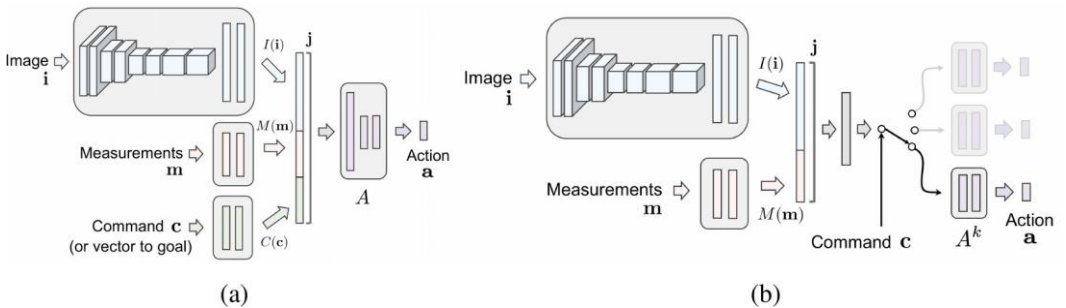
- Assumption:

$$\exists f(\cdot, \cdot) : a_i = f(s_i, c_i)$$

Better assumption!

Codevilla, Muller, Lopez, Koltun and Dosovitskiy: End-to-End Driving Via Conditional Imitation Learning. ICRA, 2018.

Conditional Imitation Learning: Network Architecture

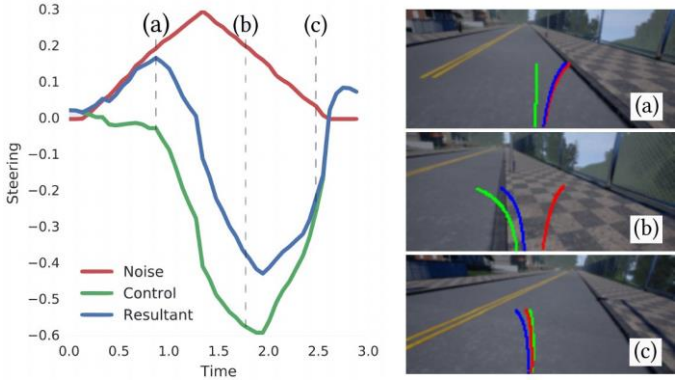


- This paper proposes two network architectures:
  - (a) Extract features $C(c)$ and concatenate with image features $I(\mathbf{i})$
  - (b) Command c acts as switch between specialized submodules
- Measurements m capture additional information (here: speed of vehicle)

Codevilla, Muller, Lopez, Koltun and Dosovitskiy: End-to-End Driving Via Conditional Imitation Learning. ICRA, 2018.

Does CIL also suffer from catastrophic drift?

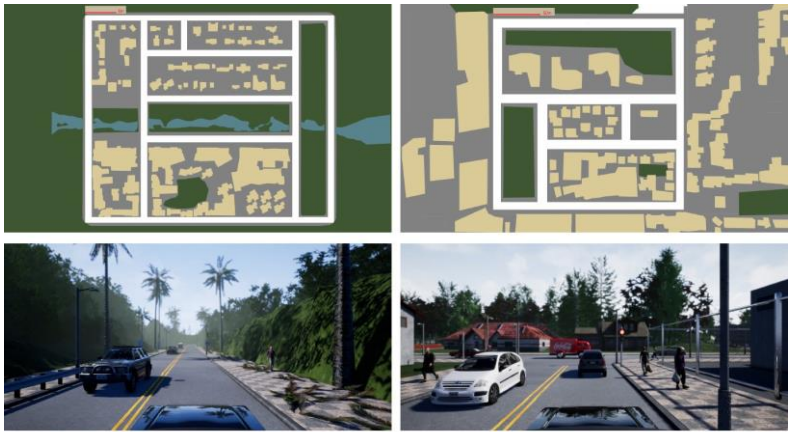
Conditional Imitation Learning: Noise Injection



- Temporally correlated noise injected into trajectories ⇒ drift
- Record driver's (=expert's) corrective response ⇒ recover from drift

Codevilla, Muller, Lopez, Koltun and Dosovitskiy: End-to-End Driving Via Conditional Imitation Learning. ICRA, 2018.

CARLA Simulator



Town 1 (training) Town 2 (testing)

http://www.carla.org

Codevilla, Muller, Lopez, Koltun and Dosovitskiy: End-to-End Driving Via Conditional Imitation Learning. ICRA, 2018.

Conditional Imitation Learning: Results

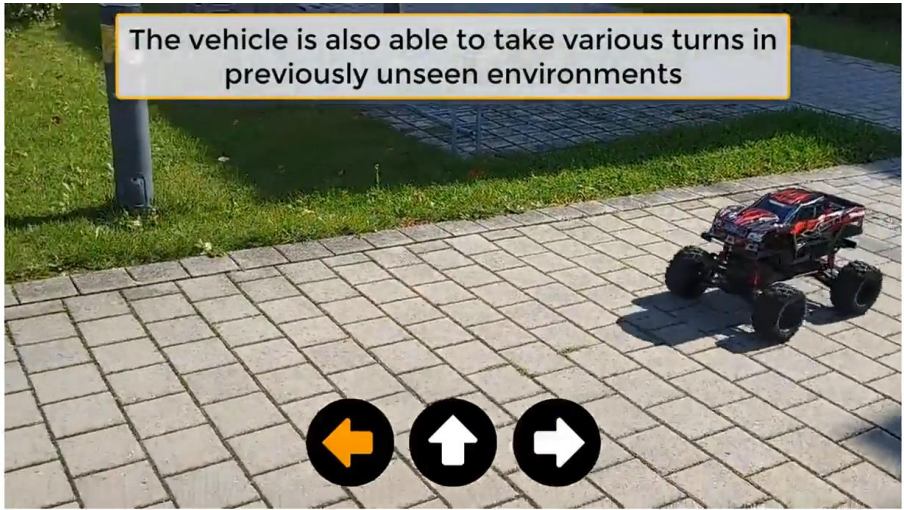
Model	Success rate		Km per infraction	
	Town 1	Town 2	Town 1	Town 2
Non-conditional	20%	26%	5.76	0.89
Goal-conditional	24%	30%	1.87	1.22
Ours branched	**88%**	**64%**	2.34	1.18
Ours cmd. input	78%	52%	3.97	1.30
Ours no noise	56%	22%	1.31	0.54
Ours no aug.	80%	0%	4.03	0.36
Ours shallow net	46%	14%	0.96	0.42

Methods:

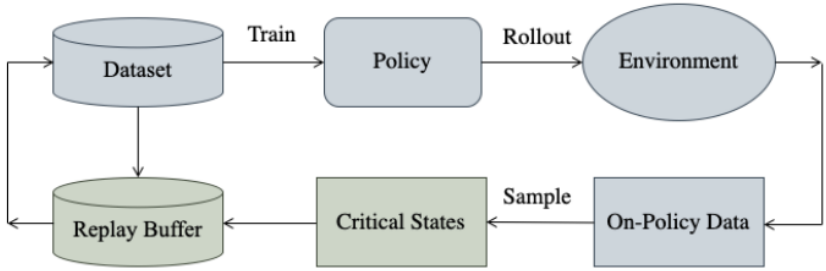
- Goal-conditional: provided with vector to goal (i.e., compass)
- Ours branched: Network architecture (b)
- Ours cmd. input: Network architecture (a)
- Ours no noise: no noise injected into trajectories (see previous slide)
- Ours no aug.: no augmentation (contrast, brightness, blur, noise, region dropout)

Codevilla, Muller, Lopez, Koltun and Dosovitskiy: End-to-End Driving Via Conditional Imitation Learning. ICRA, 2018.

Conditional Imitation Learning: Results



DAgger with Critical States and Replay Buffer



Key Ideas:

1. Sample **critical states** from the collected on-policy data based on the utility they provide to the learned policy in terms of driving behavior
2. Incorporate a **replay buffer** which progressively focuses on the high uncertainty regions of the policy's state distribution

Interpretability: GradCAM Attention Maps

CILRS [Codevilla et al. 2019]

Our Approach

Prakash, Behl, Ohn-bar, Chitta and Geiger: Exploring Data Aggregation in Policy Learning for Vision-based Urban Autonomous Driving. CVPR, 2020.

Further Readings

- Ohn-Bar et. al., Learning Situational Driving, CVPR 2020

- Ross, Gordon and Bagnell: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. AISTATS, 2011.

- Pomerleau: ALVINN: An Autonomous Land Vehicle in a Neural Network. NIPS, 1988.

- Bojarski et al.: End-to-End Learning for Self-Driving Cars. Arxiv, 2016.

- Levine et. al. :End-to-End Training of Deep Visuomotor Policies , JMLR, 2016

- Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013