Toward Deep Reinforcement Learning without a Simulator: An Autonomous Steering Example

Bar Hilleli

Supervised by: Prof. Ran El-Yaniv

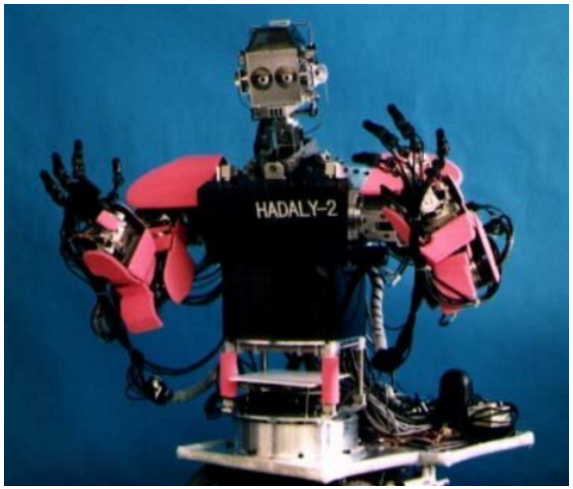
Technion - Israel Institute of Technology

barh@campus.technion.ac.il
rani@cs.technion.ac.il

July 16, 2017

Overview (1) - Motivation

- Consider designing a robot capable of performing some complex task (e.g., driving).
- Although performing such task might be natural for adult humans, designing a hard-coded algorithm for such robots can be a daunting challenge.



Overview (2) - Motivation

Obstacles in designing a hard-coded controller:

- Difficulties in accurately modeling the robot and its interaction with the environment.
- Creating hand-crafted features from high-dimensional sensor data.
- Adapt to new situations.

Therefore, approaches for **robot learning** such as *Imitation Learning* (IL) and *Reinforcement Learning* (RL) are often used.

Overview (3)

- We propose a general scheme that combines several learning techniques that might be used to tackle such challenges.
- It enables to **utilize the supervision and instruction abilities of a human** to train a robotic agent to perform complex tasks.
- Enables in principle the creation of an agent capable of performing tasks that the **instructor cannot perform himself**.
- It can be viewed as a collection of preliminary supervised learning stages that are combined with a successive RL stage.

Overview (4) - Scheme

Deep **convolutional neural networks** (CNNs) are used to implement the different parts of the scheme. The learning process consists of four phases:

1. **Imitation Learning (IL) - P_θ**
2. **Reward Induction (RI) - R_θ**
3. **Safety Module Construction - S_θ** (not going to talk about)
4. **Reinforcement Learning (RL) - Q_θ**

Overview (5) - Scheme

1. **IL** - the agent learns to mimic the human instructor using supervised learning and reach a basic performance level.
2. **RI** - learning an instantaneous reward model from instructor feedback.
3. **Safety Module** - learning a *safety network* to be integrated in the RL procedure.
4. **RL** - the reward function and the safety network are used to learn an agent policy without any human supervision.

Overview (6) - Scheme motivation

The proposed approach somewhat resembles the **natural teaching-learning** procedure used by **humans** to teach each other complex tasks:

1. the student **mimics** the actions demonstrated by the instructor.
2. the instructor provides **real-time feedback** and the student improves performance by both optimizing a policy as well as **learning the feedback function** itself.
3. the student continues to teach herself (without the instructor), using the reward function previously induced by the instructor.

Overview (7)

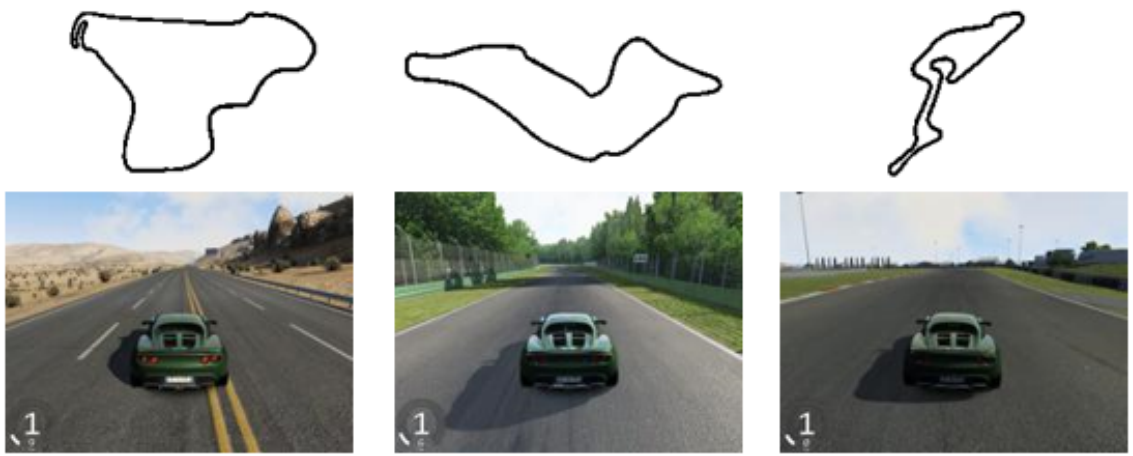
- Implemented and applied to the problem of **autonomous highway steering**.
- The input to the CNNs are the **raw image pixels**.
- Four networks are trained: P_θ, R_θ, S_θ, Q_θ.

Overview (8) - No Simulator

- The entire system is implemented **without** any access to the internal state of the game simulator (which is a purchased executable code).
- Therefore, valuable information such as the car's distance from the roadside or its angle with respect to the road could not be extracted and utilized in the learning process.
- It is impossible to accelerate the **real-world clock**.
- This limitation is hard to overcome and requires appropriate and extremely **effective RL**, as we aim to achieve here.

Overview (9) - Work environment

- The platform - one of the most popular and realistic computer racing games: **Assetto Corsa.**



Video

Click to watch video

Imitation Learning (1)

- **Imitation learning** (IL), which is also known as *Behavioral cloning* (BC) or *Learning from demonstrations* (LFD), aims at **finding a mapping** $f^{\star} : s \rightarrow a$ from a given world state, s, to a desired action a.
- A teacher performs the task at hand - driving the car **while ignoring the lane marks on the road**.
- Examples, which are simply tuples of the form (s, a), are recorded.
- They are used to **learn the mapping** in a supervised learning manner.

Imitation Learning (2)

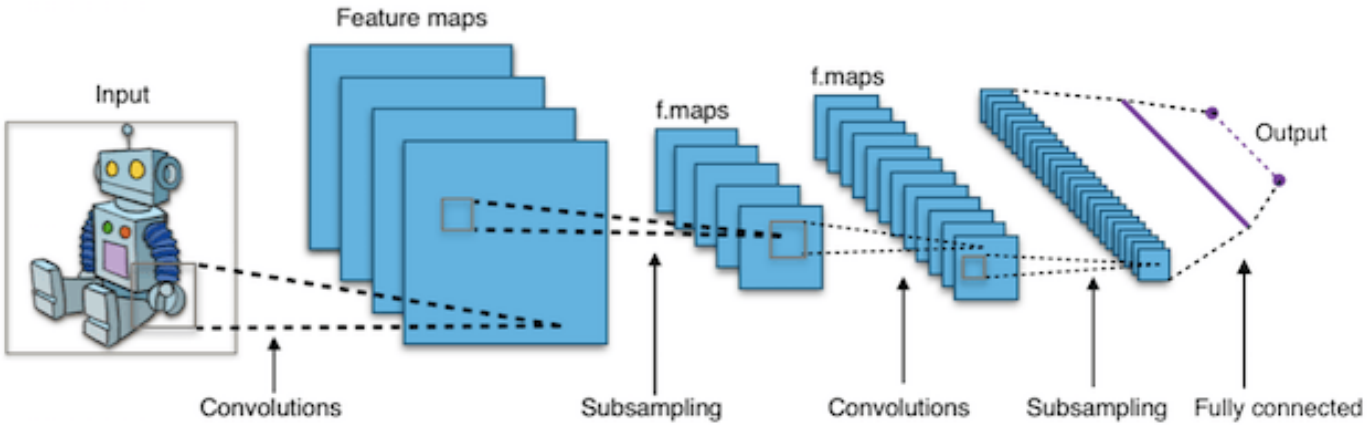
Cons

- The performance of an agent trained in this manner is **upper bounded by the demonstrator level**.
- **Sparse sampling** - if the training sample isn't sufficiently diverse/representative the agent will not be exposed to unexpected difficult states, and can suffer from very poor and unpredictable performance when such states are encountered in production.

Noting these limitations, we use imitation learning in our scheme only to achieve a basic performance level.

Imitation Learning (3)

- World states are **game images** (raw pixels), and actions are **keyboard keys**.
- The negative log-likelihood is used as a loss function for training the **policy network**, denoted by $\boldsymbol{P_\theta}$.
- $NLL(\theta, \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{i=0}^{|\mathcal{D}|} logP_\theta\left(a_i|s_i\right)$



Reward Induction (1)

- It's not possible to define a reward signal based on parameters like the car's distance from the roadside (without explicit image processing) — we don't have a simulator.
- The reward signal for the RL is **induced** from the instructor's feedback.

Reward Induction (2)

Basic flow:

1. The agent drives the car using P_θ.
2. The driving instructor **provides binary labeling**, l, for the encountered states.
3. The tuples (s, l) are recorded.
4. A **reward network**, denoted by $\boldsymbol{R_\theta}$, is trained using the recorded data in a supervised learning manner.

R_θ has an identical architecture to that of the policy network, except for the final fully connected layer that now has one node instead of three.

Reward Induction (3)

- The task we consider: **driving in a designated lane** (the second lane from the right).
- Reminder: in the IL stage the human demonstrator **ignored** the lane marks.
- R_θ maps a game image into the instantaneous reward, $r \in [-1, 1]$, corresponding to that state, $R_\theta : s \rightarrow r$.

Reward Induction (4)

- R_θ is trained to give high reward only for states where the car was in that specific lane, achieving **97%** validation accuracy.

- The MSE loss function is used:

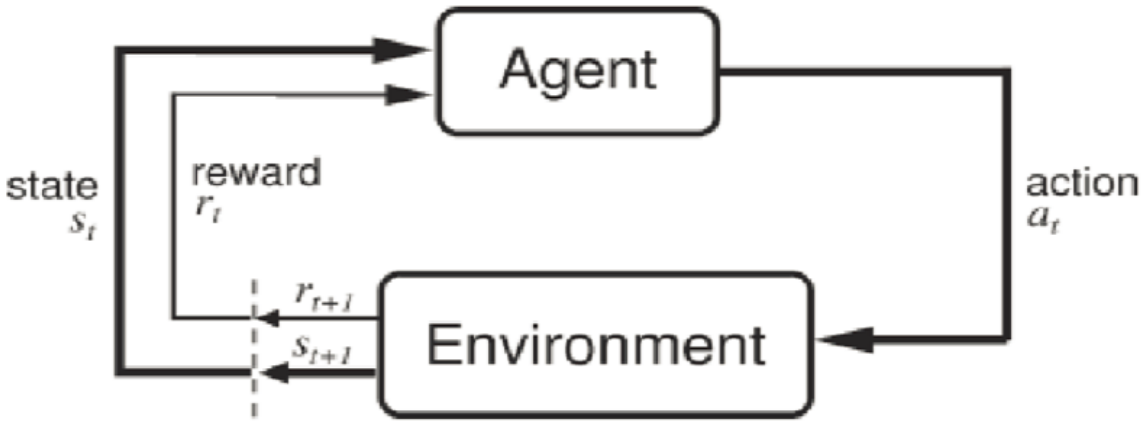
$$\mathbb{E}_{s,l\sim\mathcal{D}}\left[\left(l - R_\theta\left(s\right)\right)^2\right]$$

where $l \in \{-1, 1\}$ is the corresponding label given by the human instructor.

(a) "Bad" states

(b) "Good" states

Reinforcement Learning (1) - background



- *Reinforcement Learning* (RL) describes a set of learning problems where an **agent interacts with an environment**.
- RL algorithms seek to find a policy, π, that **maximize the reward received** over time from the environment.

Reinforcement Learning (2)

- The **action-value function** for policy π, denoted by $Q^{\pi}(s, a)$, is the **expected reward of taking action *a* in state *s* under policy π** and is formally defined as,

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} | s_t = s, a_t = a\right]$$

  γ - discount factor. r - instantaneous reward.

- We use a variant of the **Q-learning** algorithm, which aims to find the **optimal action-value function**, denoted by $Q^{\star}(s, a)$, and defined as,

$$Q^{\star}(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

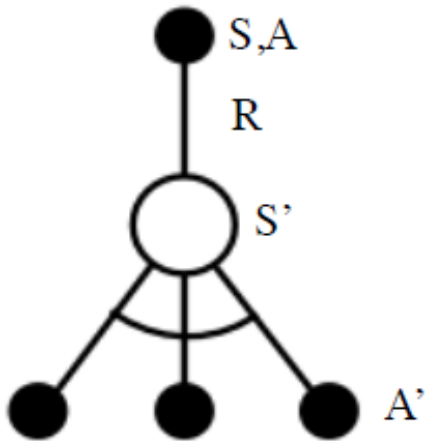
Reinforcement Learning (3)

- The *Bellman optimality equation* for $Q^\star(s,a)$ is

$$Q^\star(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} + \gamma \max_{a'} Q^\star(s_{t+1}, a') | s_t = s, a_t = a\right]$$

- The Q-learning update is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

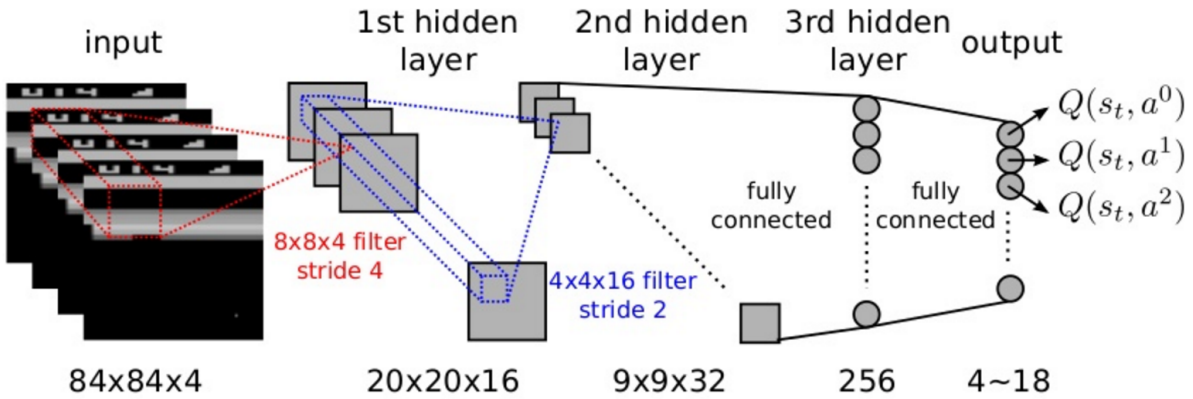


Reinforcement Learning (4)

- Dealing with a very large state space of images, we **approximate** the optimal action-value function using a **deep Q-network** (DQN) with **parameters θ**:

$$Q_\theta(s, a) \approx Q^\star(s, a).$$

- A deep Q-network is a neural network that for a given state outputs a vector of action values.



Reinforcement Learning (5)

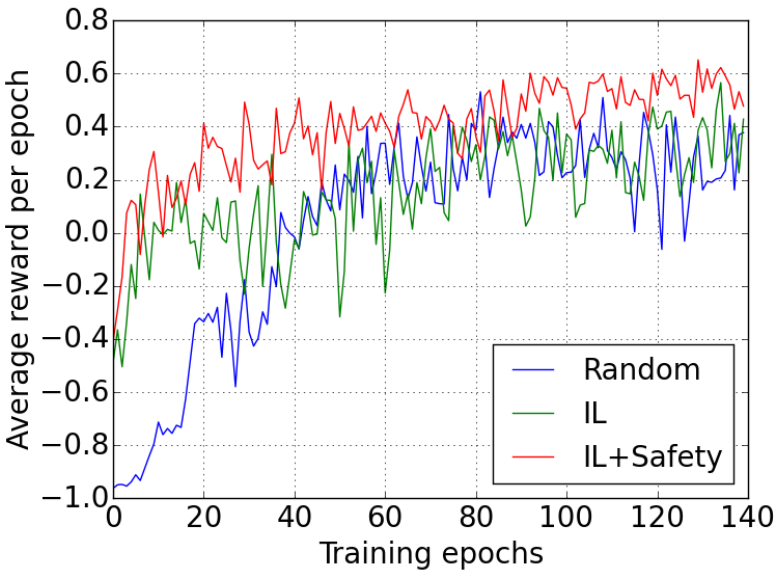
We used the DDQN algorithm, which is a slight modification of the DQN algorithm:

- Take action a_t according to ϵ-greedy policy.
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory $\mathcal{B}$.
- Sample random mini-batch of transitions (s, a, r, s') from $\mathcal{B}$.
- Compute Q-learning targets w.r.t. old, fixed parameters $\tilde{\theta}$.
- Optimize MSE between Q-network and Q-learning targets

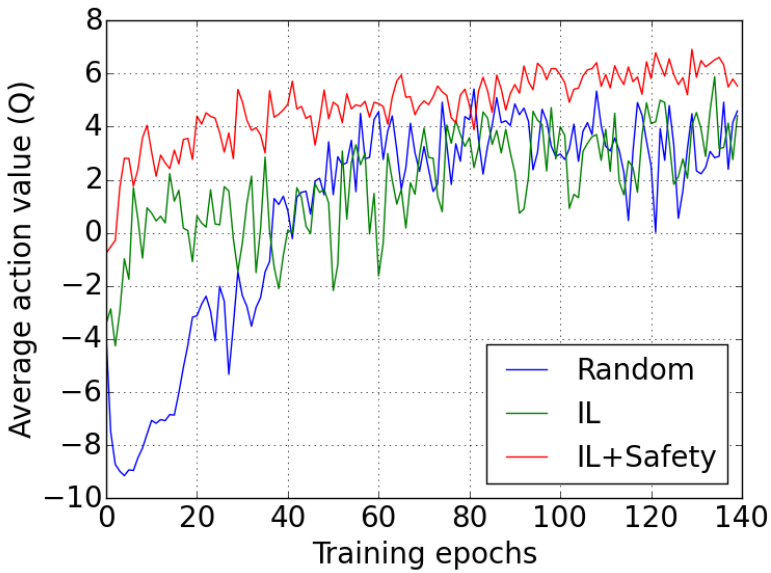
$$L(\theta) = \mathop{\mathbb{E}}_{s,a,r,s' \sim \mathcal{B}} \left[\left(r + \gamma \max_{a'} Q_{\tilde{\theta}}(s', a') - Q_{\theta}(s, a) \right)^2 \right]$$

- Using variant of stochastic gradient descent.

Experiments and Results (1)



(a) Average reward



(b) Average action-value

Figure: Agent's RL training curves. An epoch is defined to be 15,000 sample frames. (a) Average reward per epoch. (b) Average action-value per epoch. The average reward achieved by the imitation policy, P_θ, was -0.45.

Experiments and Results (2)

Table: **Advantage of using the safety module and IL initialization in decreasing number of accidents.**
Averaged number of accidents per epoch throughout the agent's RL training. The training is divided into four different episodes as follows: epochs 0-3, epochs 3-12, epochs 12-39 and epochs 39-120. The number of accidents in the last episode is averaged over 81 epochs, but in the final epochs both the **IL** and **IL + Safety** agents can complete tracks without accidents.

	0-3	**3-12**	**12-39**	**39-120**
Random	116.6	104.7	68.5	41
IL	67	65.3	68.1	44.5
IL + Safety	47	46.1	36.1	23.3

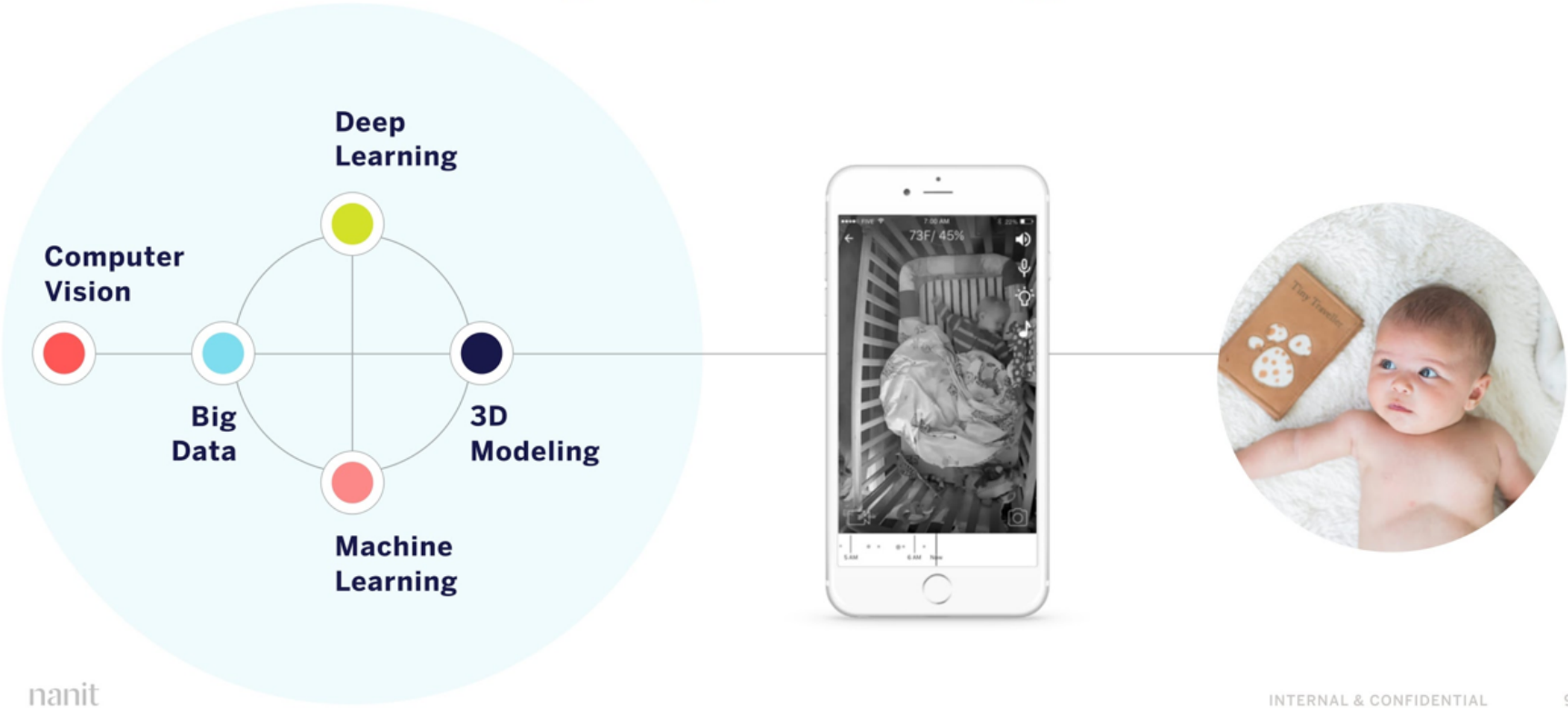
Nanit - The Baby Moniter That Thinks

nanit

Nanit - Machine Learning

MEET NANIT

Nanit Makes Cutting-Edge Technology Accessible



nanit

INTERNAL & CONFIDENTIAL 9

Thank you

Safety Module (1)

- We incorporate the notion of safety into the RL process using a **safety module** that is composed of:
  - A **safety network**, denoted by S_θ.
  - A **safe driving policy**, denoted by P_θ^{safety}.
- S_θ is able to identify risky states encountered by the agent.
  $S_\theta : s \rightarrow [-1, 1]$
- P_θ^{safety} takes control of the agent's controller when it is in an unsafe state in order to return the agent to a safe state.

Safety Module (2)

- S_θ has the same architecture as R_θ^{lane}, and is trained in the same way as R_θ^{lane}, but with different dataset (including labeling).
- The safe driving policy is the learned policy from the IL stage, P_θ.
- By using the proposed safety module, we were able to train our agent faster and with **fewer accidents**.

Safety Module (3) - output example

(a) "Unsafe" states

(b) "Safe" states

Figure: S_θ outputs a high score for "safe" states (b), and low score for "unsafe" states (a).

Safety Module (4)

- When unsafe states are encountered, P_θ^{safety} takes control of the car until it is out of danger.

- $$\textbf{driving policy} = \begin{cases} P_\theta^{safety}, & \text{if } S_\theta < threshold \\ Q_\theta, & \text{if } S_\theta > threshold \end{cases}.$$

  Where Q_θ is a Q-network that represents the RL policy (it will be explained in the next slides).

Reinforcement Learning - safe RL background

- In **safe RL**, the learning agent seeks to find a policy that maximizes the long-term future reward received from the environment while **respecting some safety constraints** (e.g., damage avoidance).
- Two main approaches:
  - Transforming the **optimization criteria** to include a notion of risk (e.g., variance of return, worst-outcome)
  - Modifying the **agent's exploration process** while utilizing prior knowledge of the task to avoid risky states.
- Our proposed method is related to the second approach.

Experiments and Results (4)

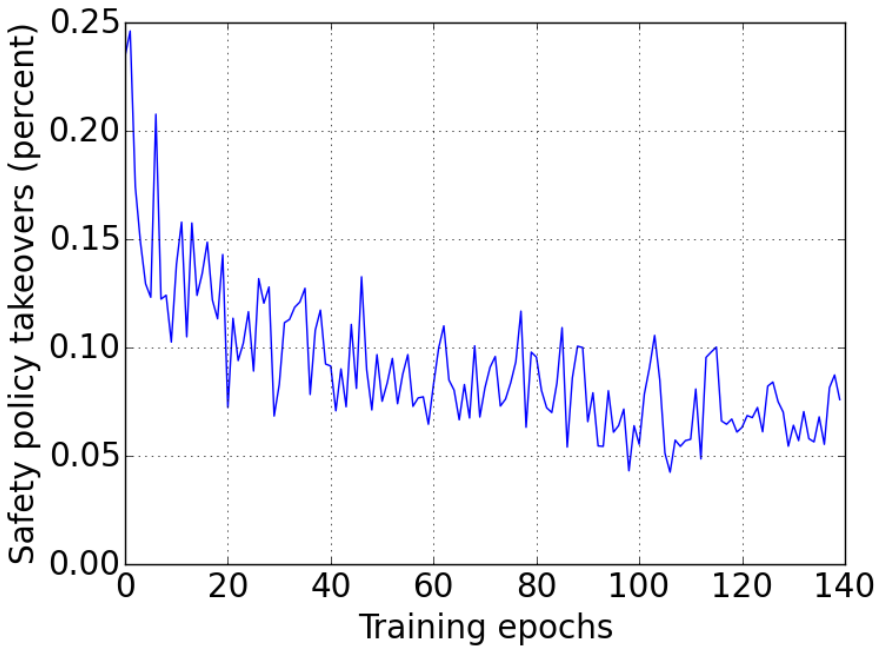


Figure: Fraction of **safety policy takeovers** per decision in each epoch during the RL. The Q-network was initialized with the IL policy network parameters (IL initialization).

Experiments and Results (5)

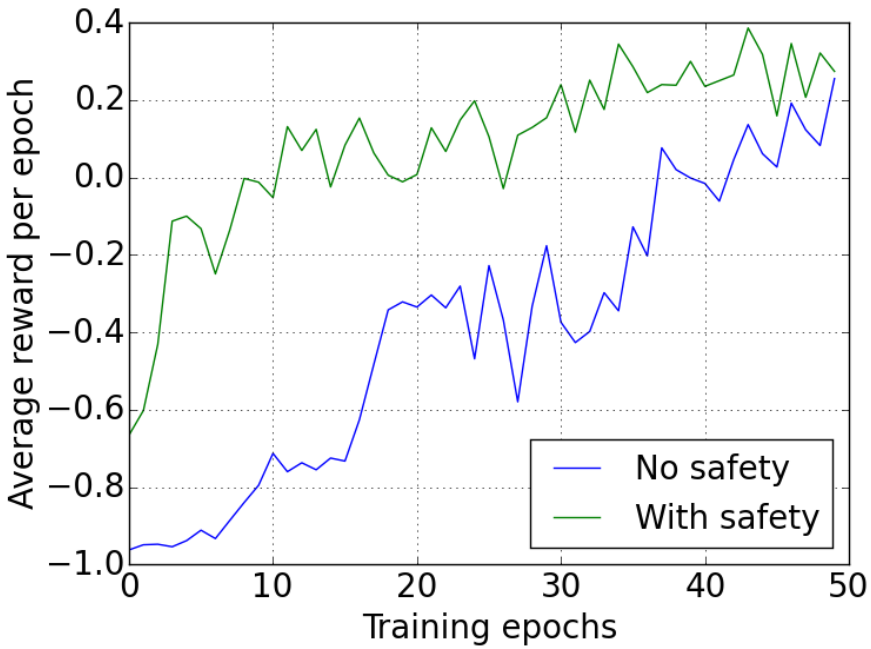


Figure: **Safety module advantage.**
Each point in the graph is the average reward per epoch. The Q-network was initialized with random parameters in both cases.

Experiments and Results (6)

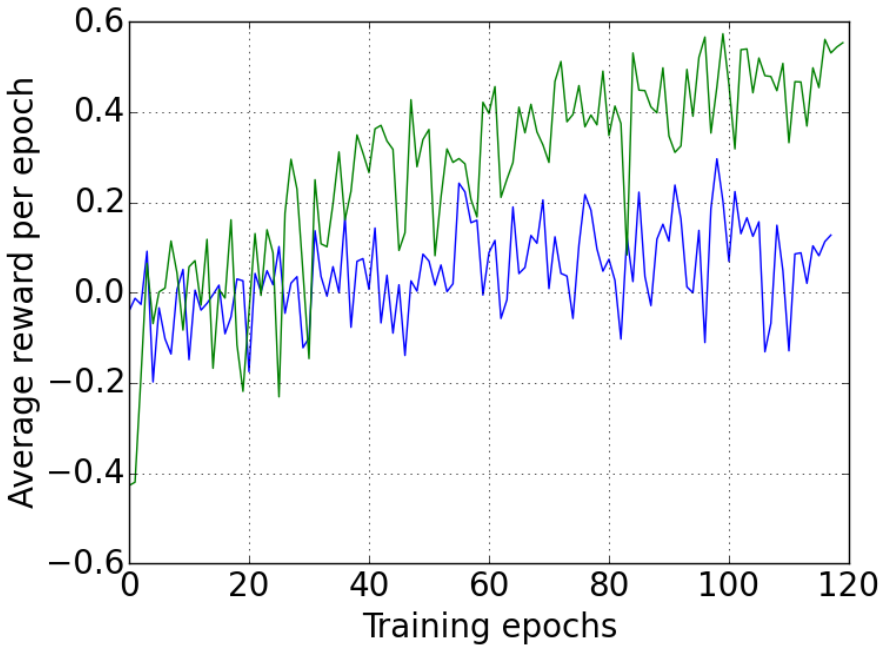


Figure: The **importance of a high-quality reward signal for the RL.** Agent's RL training curves. Each point in the graphs is the average reward achieved per epoch from the reward network R_θ^{lane}. Green: reward is received from R_θ^{lane}. Blue: reward is received from $R_\theta^{0.2\ lane}$.