MLSS 2021 TAIPEI

ELSA LAB

# Fundamentals and Applications of Deep Reinforcement Learning

8/4/2021

**Prof. Chun-Yi Lee**

cylee@cs.nthu.edu.tw

**Elsa Lab, Department of Computer Science**

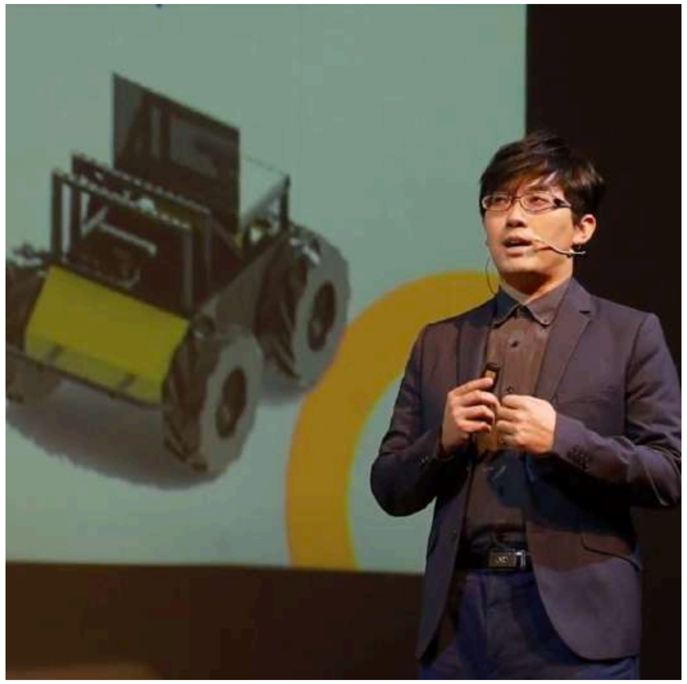
**National Tsing Hua University**

# Principle Investigator Biography

- **Dr. Chun-Yi Lee**
  - **Associate Professor** of Computer Science, National Tsing Hua University (**2015-present**)
  - NVIDIA Deep Learning Ambassador and Certified Instructor (**2017-present**)
  - Senior Hardware Engineer, Oracle America, Inc. (**2012-2015**)

- **Education**
  - Ph.D., Department of Electrical Engineering, **Princeton University**
  - M.S. and B.S., Department of Electrical Engineering, **National Taiwan University**

- **Honors and Awards**
  - **2020 Ta-You Wu Memorial Award**, Ministry of Science and Technology (MOST)
  - **2020 NVIDIA AI at the Edge Challenge** (**2nd Place**)
  - **2020** Distinguished Teaching Award, National Tsing Hua University (**Top 1%**)
  - **2019** Young Scholar Innovation Award, Foundation for the Advancement of Outstanding Scholarship
  - **2019** Young Scholar Research Award, Taiwan Semiconductor Industry Association
  - **2019** Young Scholar Fellowship, Ministry of Science and Technology of Taiwan (MOST)
  - **2018** Outstanding Young Engineer Award, the Chinese Institute of Electrical Engineering
  - **2018 NVIDIA Jetson Developer Challenge** (**Champion Grand Prize**)
  - **2018** Outstanding Teaching Award, National Tsing Hua University
  - **2018** Distinguished Young Scholar Research Award, National Tsing Hua University (**Top 1%**)
  - **2018** ECCV Person In Context (PIC) Challenge (**2nd Place**)
  - **2017-2018** NVIDIA GPU Grant
  - **2016 NVIDIA Intelligent Embedded Robotics Challenge** (**Champion**)
  - **2009** ICCD Best Paper Award



- **Research Domains**
  - **Intelligent Robotics**
  - **Deep Reinforcement Learning**
  - **Computer Vision for Robotics**
  - **Virtual-to-Real Learning for Robotics**
  - **Parallel Embedded Systems**
  - **Parallel Computing**

# Elsa Lab

**Elsa Lab** is supervised by **Prof. Chun-Yi Lee**, and is a professional research team dedicated to developing innovative **deep reinforcement learning** and **computer vision** technologies for **intelligent robotics** and **autonomous agents**.

**Elsa Lab** welcomes full-time research assistants, Ph.D. students, master students, and undergraduate students.

**ELSA Lab**

http://elsalab.ai

# Agenda

- **Reinforcement Learning Backgrounds**
- **DRL Techniques**
- **Exploration**
- **Robotic Applications**
- **Summary**

MLSS 2021 TAIPEI

ELSA LAB

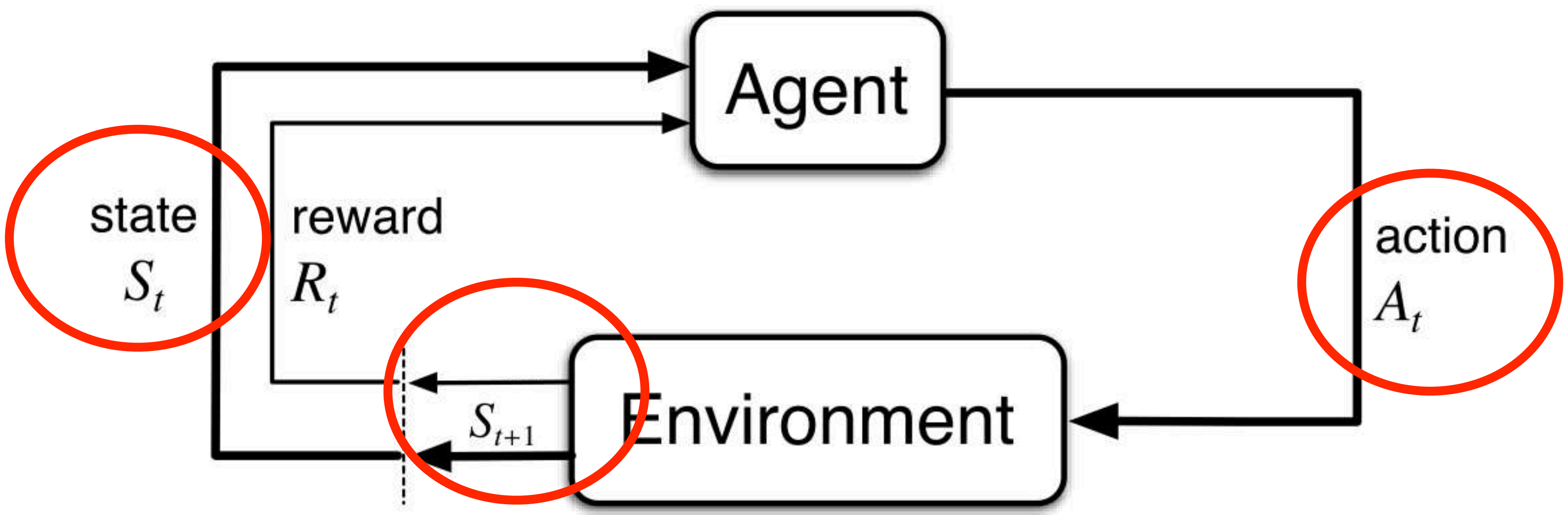
http://elsalab.ai

# Markov Decision Process (MDP)

At each timestep *t* …

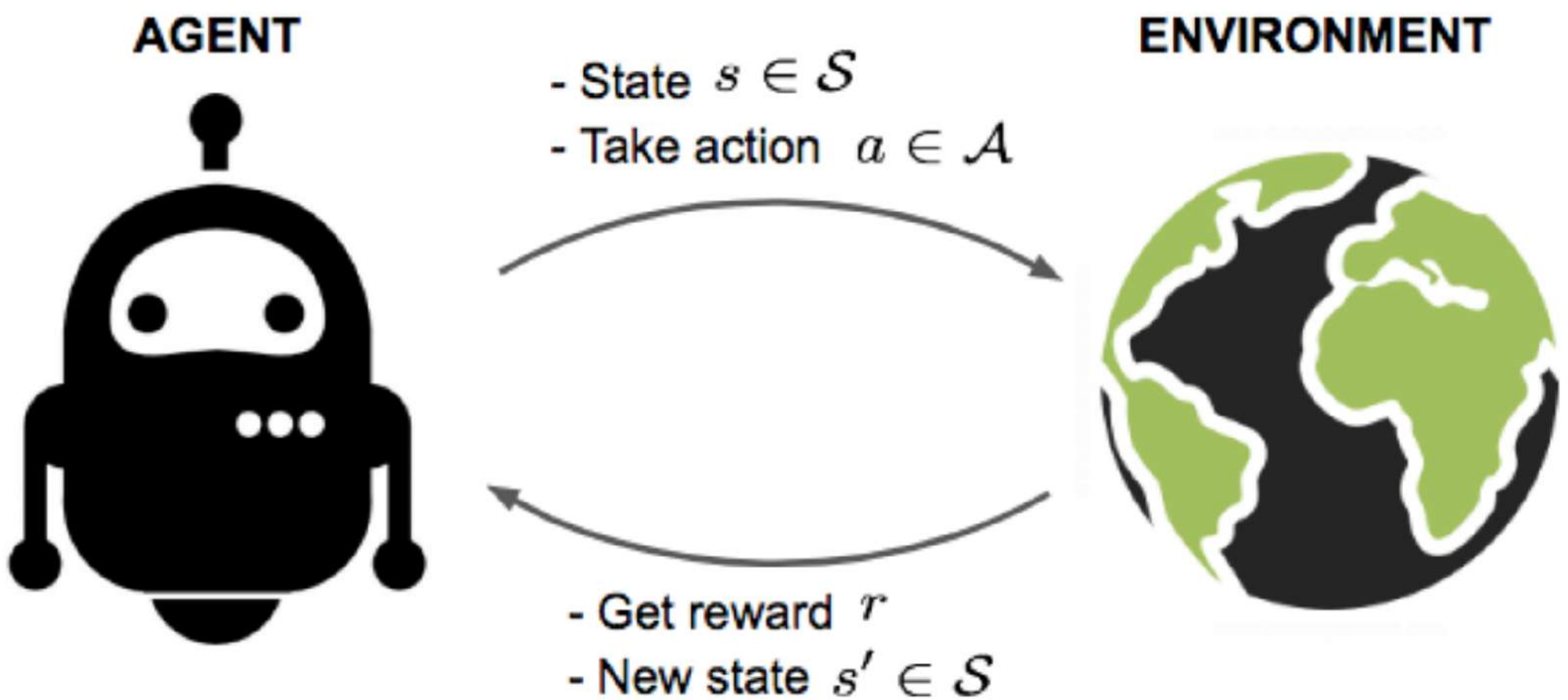
The agent observes the state $s_t$

Then take the action according to the policy $\pi : \mathcal{S} \mapsto \mathcal{A}$




Receive the **reward** and **next state**.

# Deep Reinforcement Learning

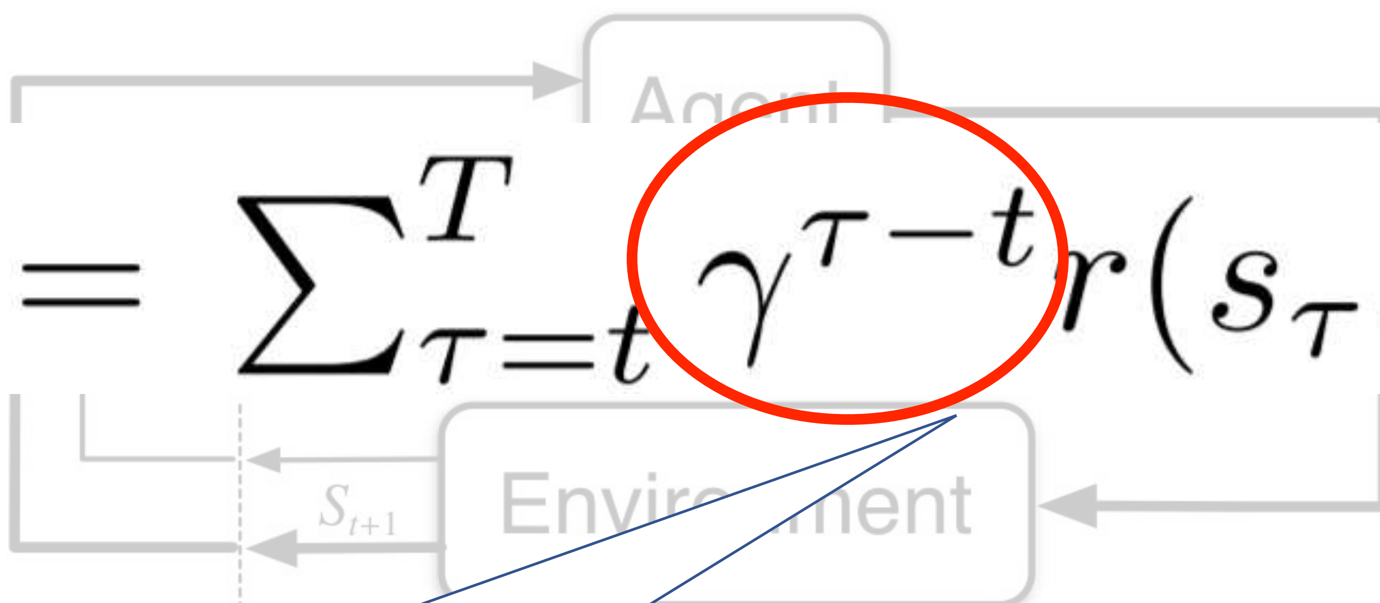



- The goal of the agent is to maximize the expected sum of rewards

$$\sum_t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[ r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

# The Goal of MDP

**<u>maximize the discounted accumulated reward at each timestep t</u>.**




$$G_t = \sum_{\tau=t}^{T} \gamma^{\tau - t} r(s_\tau, a_\tau)$$

Why discounting ?
In theory, it is for **convergence proof.**
In practice, it is to **prevent unrealistic planning**.

# How does RL solve MDP

- Recap: the objective of MDP

$$G_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$$

- RL searches $\pi^*$ according to the following criteria:

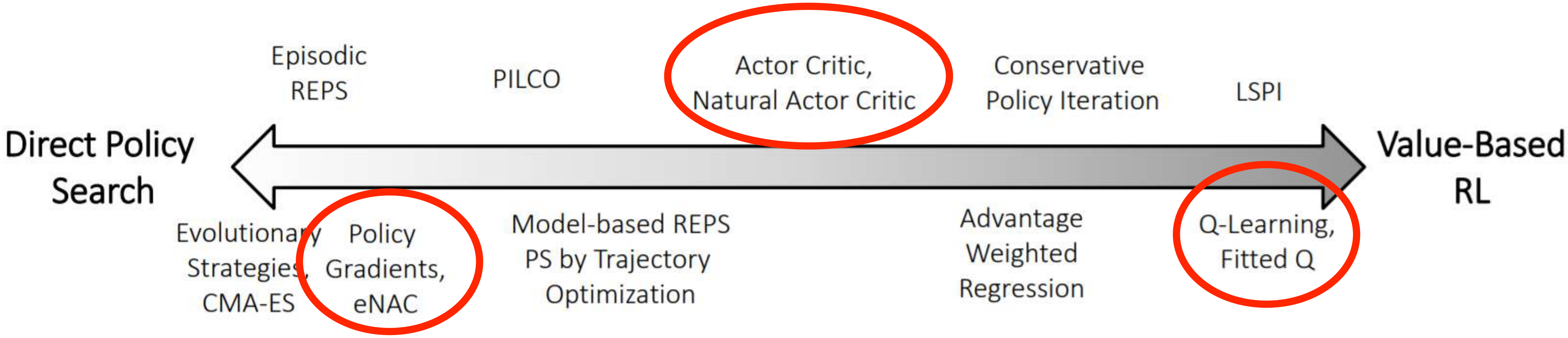
$$\pi^* = \arg\max_{\pi} \mathbb{E}\big[G_t \big| s_t = s, a_i = \pi(s_i)\big], \forall s \in \mathcal{S}$$

- Policy can be a deterministic function or a probability distribution

$$a_t = \pi(s_t) \quad \text{Deterministic function}$$
$$a_t \sim \pi(a_t | s_t) \quad \text{Probability distribution}$$

# Categories of RL

We only list a few branches of RL algorithms.



# Policy gradient

- Parameterize the policy as a probability distribution with $\theta$: $\pi_\theta$
- Learn a policy by maximizing the objective function: $J(\theta)$

$$J(\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta}\left[Q^\pi(s, a)\right] \quad d^\pi(s) = \lim_{t \to \infty} p(s = s_t | s_0, \pi)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta}\left[Q^\pi(s, a) \nabla_\theta \log \pi_\theta(a|s)\right]$$

$$Q^\pi(s, a) = \mathbb{E}\left[G_t | s_t = s, a_t = a, \pi\right]$$

- Take an action by:

$$a_t \sim \pi_\theta(a|s_t)$$

# Q-learning

- Learn the state-action value function:

$$Q(s,a) \approx \mathbb{E}\big[G_t | s_t = s, a_t = a\big]$$

- Update the Q-function according to:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r(s_t,a_t) + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1},a) - Q(s_t,a_t))$$

- Take an action by:

$$a_t = \operatorname*{argmax}_a Q(s_t,a)$$

# Comparison - Q-learning v.s. Policy Gradient

- Q-learning is an **off-policy** algorithm as the training data can be generated by an arbitrary policy

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a) - Q(s_t, a_t))$$

- Policy gradient is an **on-policy** algorithm as the training data must be from the current policy

The stationary distribution of the current policy

$$J(\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta}\left[Q^\pi(s, a)\right]$$

# Actor-Critic

- Learn the **policy** and **state/state-action value function** concurrently

$$\pi_\theta(a_t|s_t) \qquad V(s) = \mathbb{E}\left[G_t|s_t = s\right] \qquad Q(s,a) = \mathbb{E}\left[G_t|s_t = s, a_t = a\right]$$

- Use **policy gradient** to learn a policy:

$$\text{maximize} \quad J(\theta^\pi) = \mathbb{E}_{s\sim d^{\pi_\theta}, a\sim\pi_\theta}\left[V_{\theta_V}(s)\right] \qquad J(\theta^\pi) = \mathbb{E}_{s\sim d^{\pi_\theta}, a\sim\pi_\theta}\left[Q_{\theta_Q}(s,a)\right]$$

- Use **Q-learning** to learn a state/state-action value function:

$$\text{minimize} \quad J(\theta^V) = \mathbb{E}_{s_t\sim d^{\pi_\theta}, a_t\sim\pi_\theta}\left[\left(r(s_t,a_t) + \gamma V_{\theta^V}(s_{t+1}) - V_{\theta^V}(s_t)\right)^2\right]$$

$$J(\theta^Q) = \mathbb{E}_{s_t\sim d^{\pi_\theta}, a_t\sim\pi_\theta}\left[\left(r(s_t,a_t) + \gamma \max_a Q_{\theta^Q}(s_{t+1},a) - Q_{\theta^Q}(s_t,a_t)\right)^2\right]$$

- Take action by:

$$a_t \sim \pi_\theta(a|s_t)$$

# Agenda

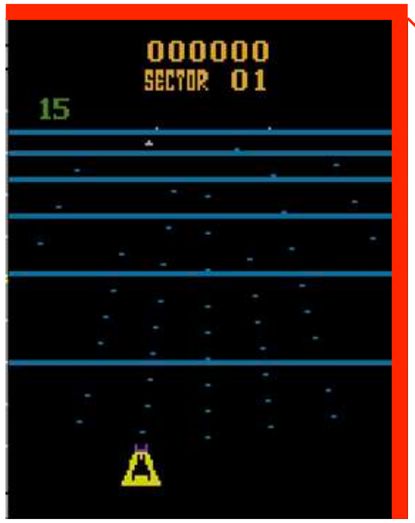
- Reinforcement Learning Backgrounds
- DRL Techniques
- Exploration
- Robotic Applications
- Summary

MLSS 2021 TAIPEI

ELSA LAB
http://elsalab.ai

# Why DRL?

- The traditional RL employs simple models (e.g. single layer neural networks)
- They fail on the high-dimensional state space (the curse of dimensionality)

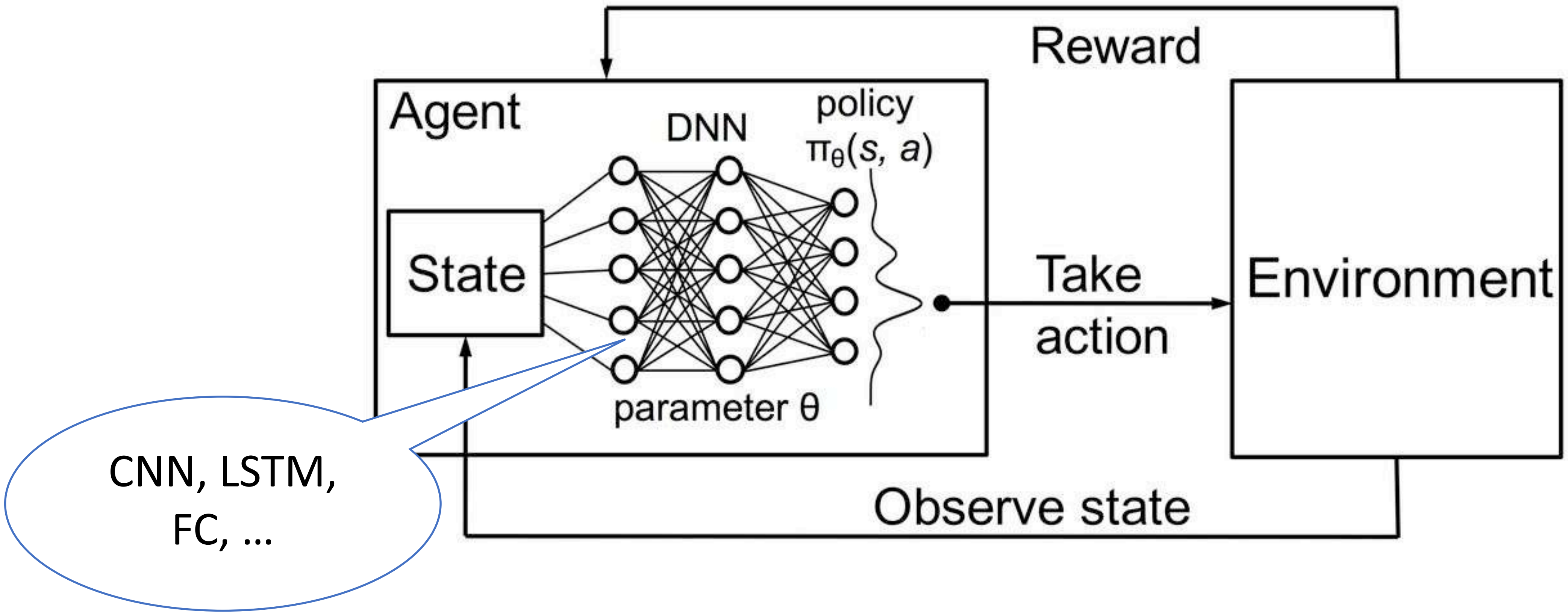



$$(210 \times 160 \times 3)$$

Quite a large size

# The Basic Concept of DRL

- Embrace the expressive power of deep neural network (DNN)



# The Contemporary DRL Algorithms

- Deep Q-Network (DQN)
- Deep Deterministic Policy Gradient (DDPG)
- Asynchronous Advantage Actor-Critic (A3C)
- Trust Region Policy Optimization (TRPO)
- Maximum Entropy RL

# The Main Concepts of DQN

- Parameterize the Q-function with a DNN
- Enhance the data-efficiency by
- Enhance the performance by tra
  functions

Make it more stable

$$J(\theta) = \mathbb{E}_{s_t,a_t,s_{t+1}\sim\mathcal{Z}}\left[\left(r(s_t,a_t)+\gamma \max_a Q_{\theta^-}(s_{t+1},a)-Q_\theta(s_t,a_t)\right)^2\right]$$

$Q_{\theta^-}$ is a Q-function of the target network

$\mathcal{Z}$ is experience replay (buffer) consisting of many $(s_t, a_t, s_{t+1})$

# The Main Concepts of DDPG

- Combine **DQN** (experience replay, target network) and **Actor-Critic**
- Reformulate the objective as (according to DPG theorem):

Actor $J(\theta^\pi) = \mathbb{E}_{s_t \sim \mathcal{Z}}\left[Q_{\theta^Q}(s_t, \pi_{\theta^\pi}(s_t))\right]$

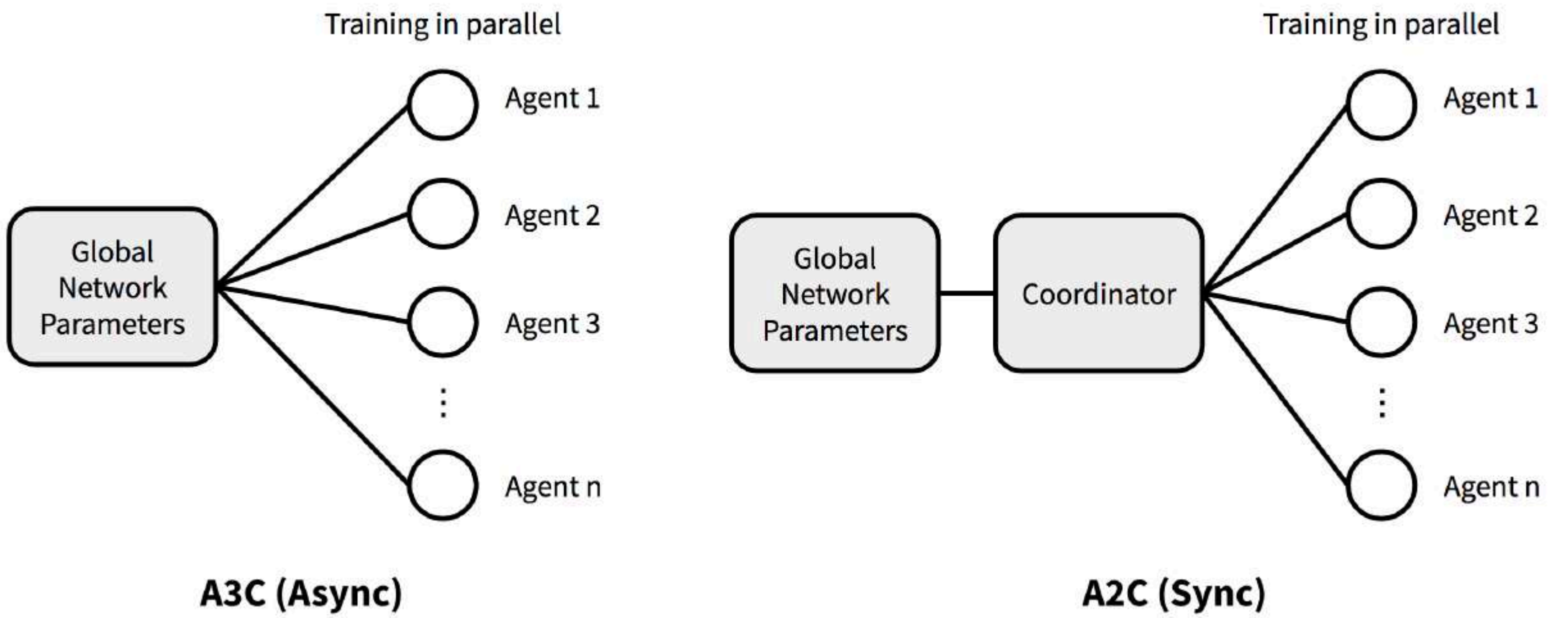
Critic $J(\theta^Q) = \mathbb{E}_{s_t, a_t, s_{t+1} \sim \mathcal{Z}}\left[(r(s_t, a_t) + \gamma Q_{\theta^{Q^-}}(s_{t+1}, \pi_{\theta^\pi}(s_{t+1})) - Q_{\theta^Q}(s_t, a_t))^2\right]$

$\theta^\pi, \theta^Q$ are the parameters for actor and critic

$\mathcal{Z}$ is experience replay consisting of many $(s_t, a_t, s_{t+1})$

# The Main Concepts of A3C

- The formulation remains the same as **actor-critic** method
- Parameterize the actor and critic with DNN
- Concurrently collect data from multiple workers, update a single policy.
- However, the recent works found that the synchronous version of A3C is more efficient in GPU, which is called **A2C**
- 



# Trust Region Policy Optimization

- Abbreviated as TRPO

- The objective and constraint function can be expressed as :

$$\max_{\theta} \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}[\frac{\pi_\theta(a\,|\,s)}{q(a\,|\,s)} Q_{\theta_{old}}(s, a)] \ ,$$

$$\text{subject to } \ \mathbb{E}_{s\sim\rho_{\theta_{old}}}[\bar{D}_{KL}^{\rho_{\theta_{old}}}(\pi_{\theta_{old}}(\,.\,|\,s)\,||\,\pi_\theta(\,.\,|\,s))] \leq \delta$$

- TRPO guarantees monotonic improvement for policy updates

# Maximum Entropy Reinforcement Learning

- Standard reinforcement learning

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s_t,a_t)\sim\pi}[\sum_t R(s_t, a_t)]$$

- Maximum entropy reinforcement learning
    - An entropy term $\alpha\mathcal{H}(\pi(\cdot\,|\,s_t))$ is introduced to encourage exploration
        - $\alpha$ is a hyper-parameter for controlling how important the entropy term is
        - $\mathcal{H}(\pi(\cdot\,|\,s_t))$ is the entropy function
        - The policy is trained to maximize **(1) the expected return and (2) entropy of the actions**

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s_t,a_t)\sim\pi}[\underbrace{\sum_t R(s_t, a_t)}_{(1)} + \underbrace{\alpha\mathcal{H}(\pi(\cdot\,|\,s_t))}_{(2)}]$$

# Agenda

- Reinforcement Learning Backgrounds
- DRL Techniques
- Exploration
- Robotic Applications
- Summary

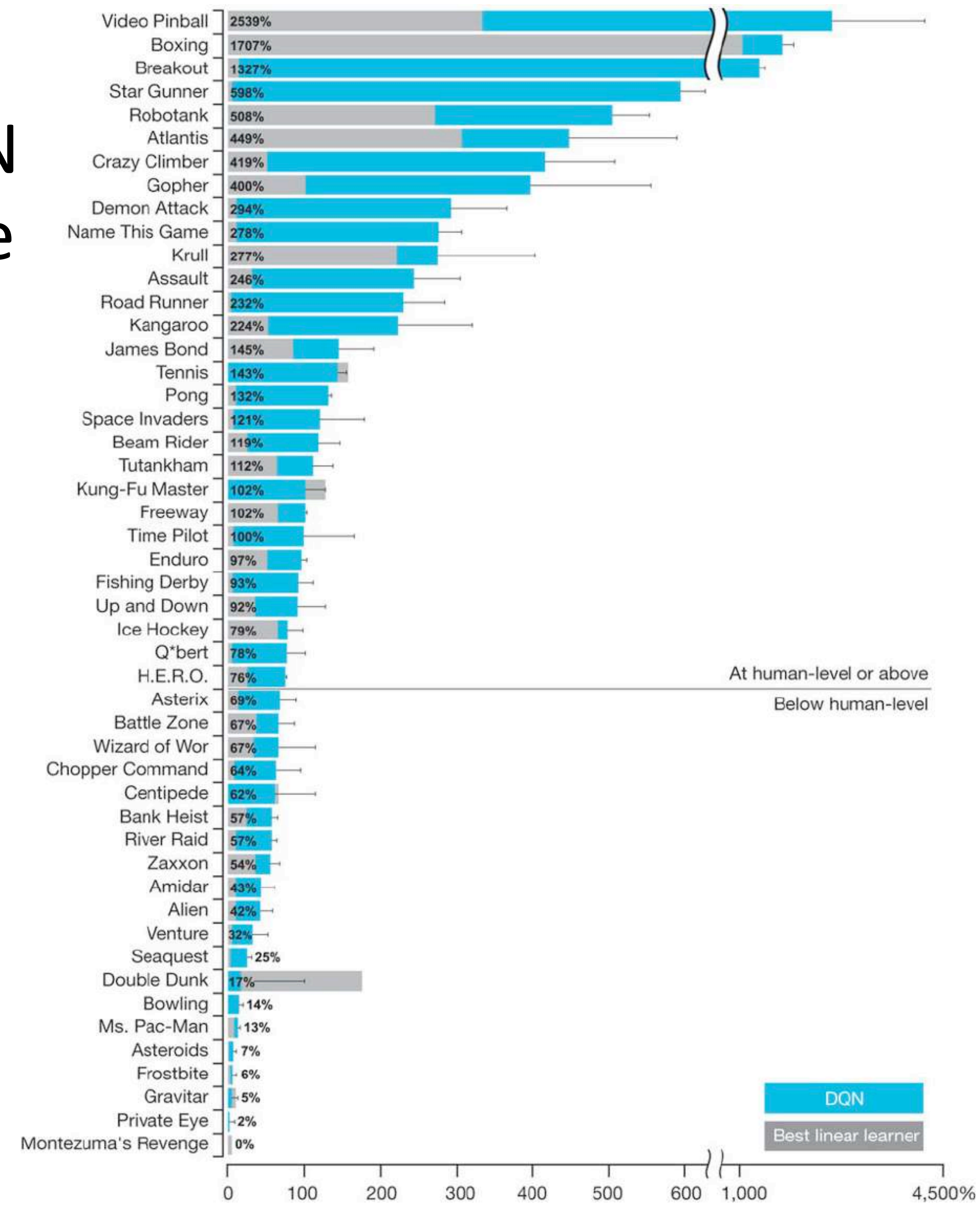
MLSS 2021 TAIPEI

ELSA LAB
http://elsalab.ai

# DRL is Extremely Data-Inefficient

- For example, in the best case, DQN consumes **200M** frames to achieve the human-level performance!

- Moreover, DQN mostly fails to human even consuming **> 200M** frames.



# If you are a Supervise Learning Practitioner

- You might wonder that why supervised learning (SL) can train DNN efficiently on several dataset (e.g. MNIST, CIFAR-10, Imagenet), but DRL cannot do that.

It is because the ”training data”!

Bad training data contribute nothing on RL
Just like unclean training data undermines SL

# Training Data for DRL

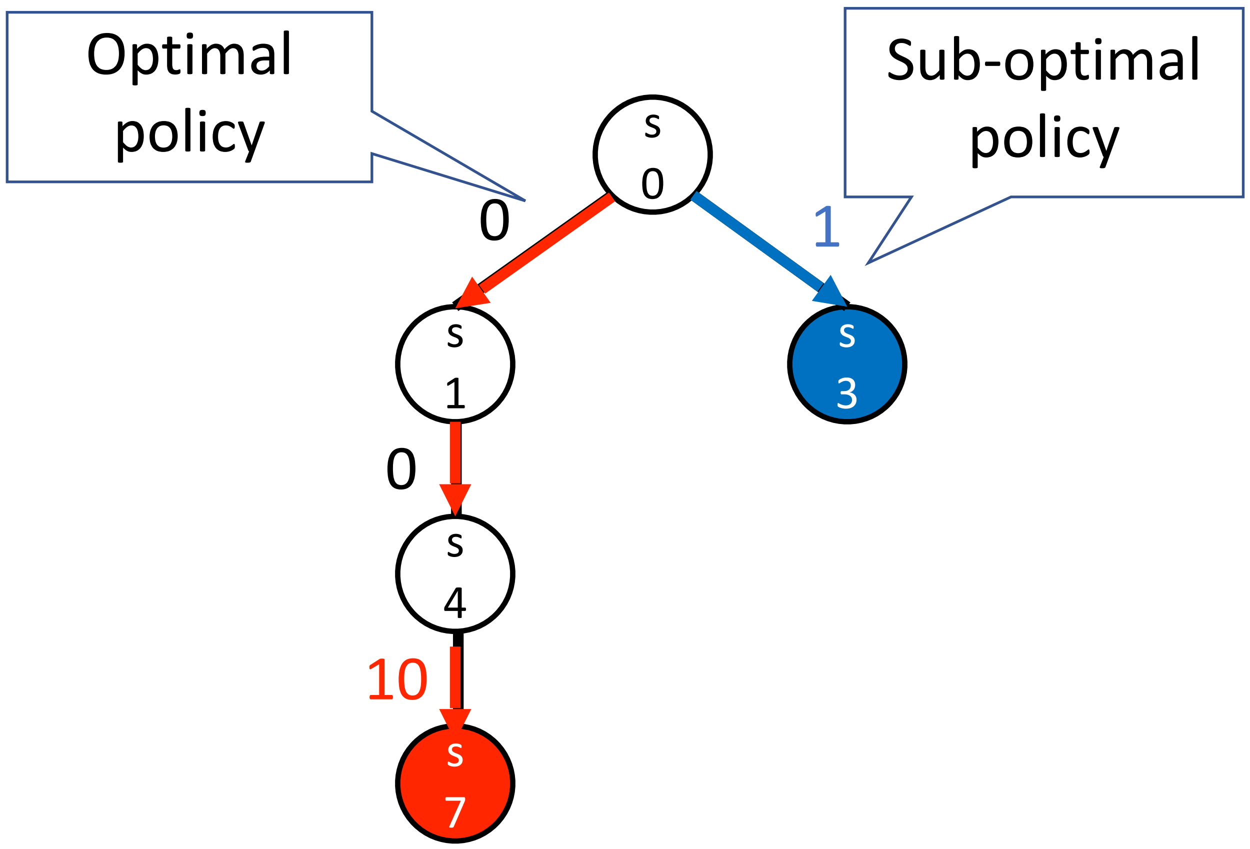
- Recap: the objective function of RL

Usually filled by the agent itself. Filling by human takes too much time.

$$J(.) = \mathbb{E}_{s_t, a_t, s_{t+1} \sim \mathcal{Z}}[\cdot]$$

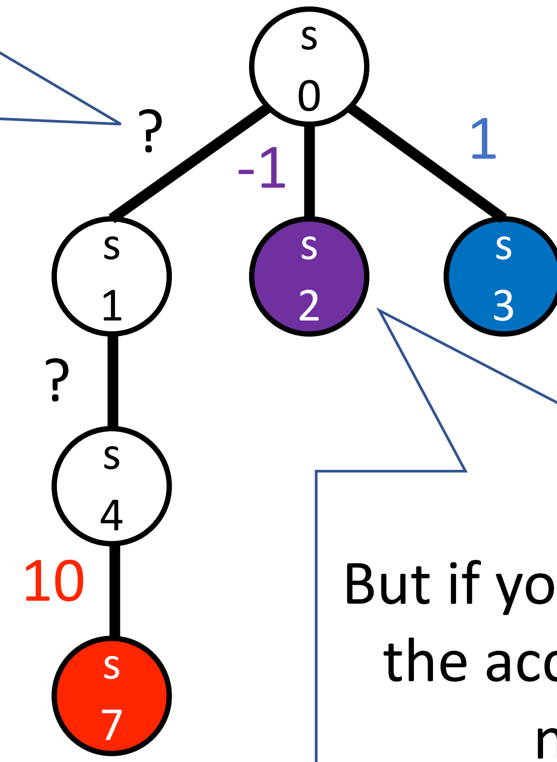
$$J(.) = \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta}[\cdot]$$

# Sub-Optimal Policy



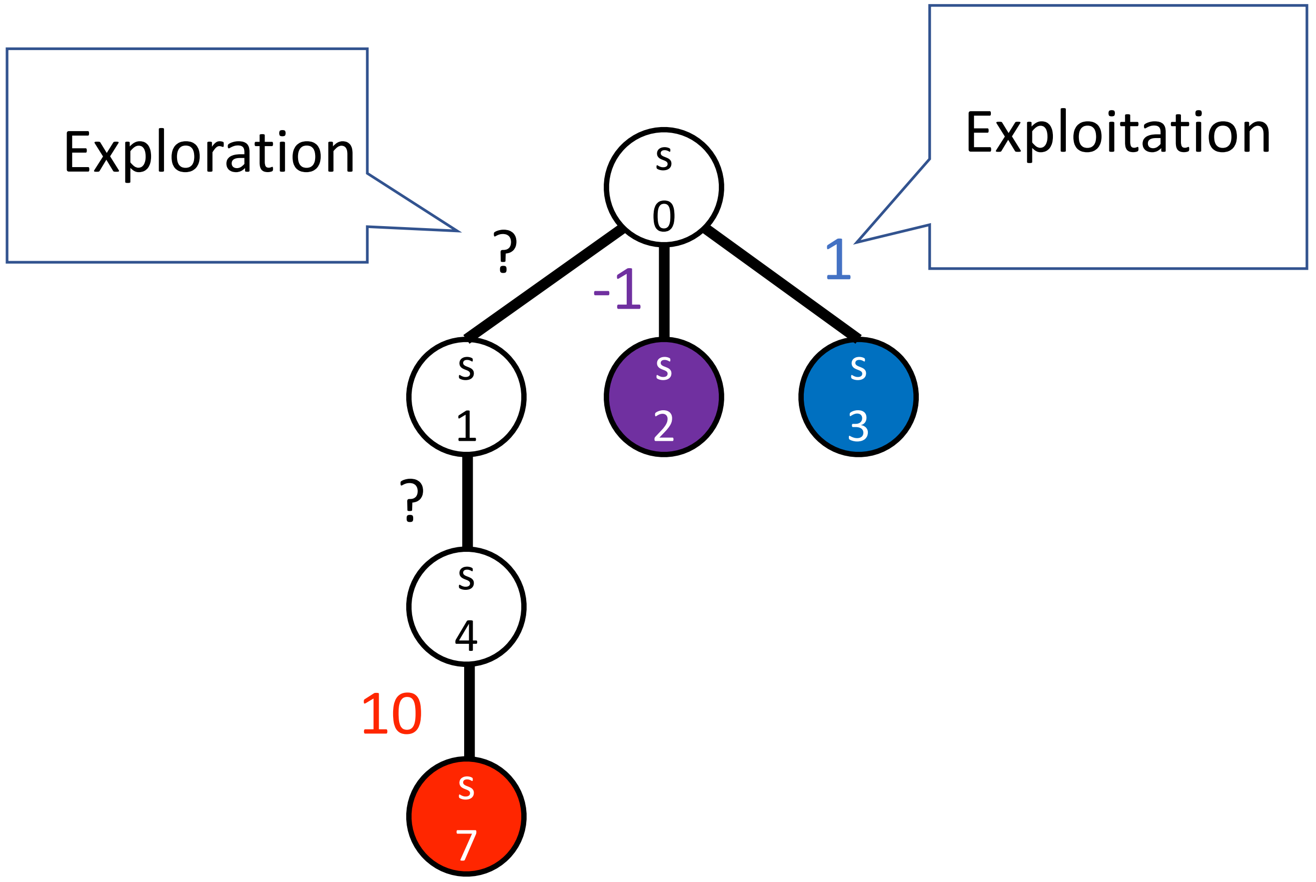
# The Cause of Sub-Optimal Policy

If the agent don’t try the unfamiliar action, it will not find the higher reward




But if you keep trying all actions, the accumulated rewards will not be maximized

# Exploitation and Exploration

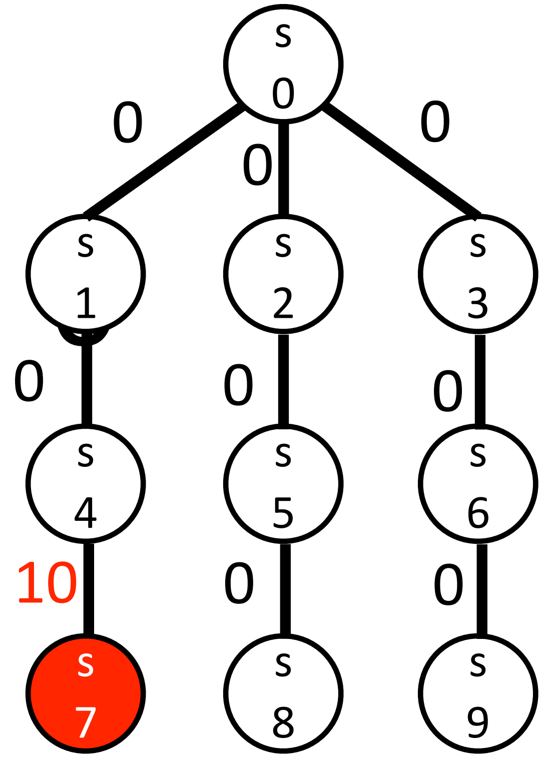



**Balancing the exploitation and exploration is necessary for RL, otherwise no useful training data can be obtained**

# Sparse Rewards and Deceptive Rewards

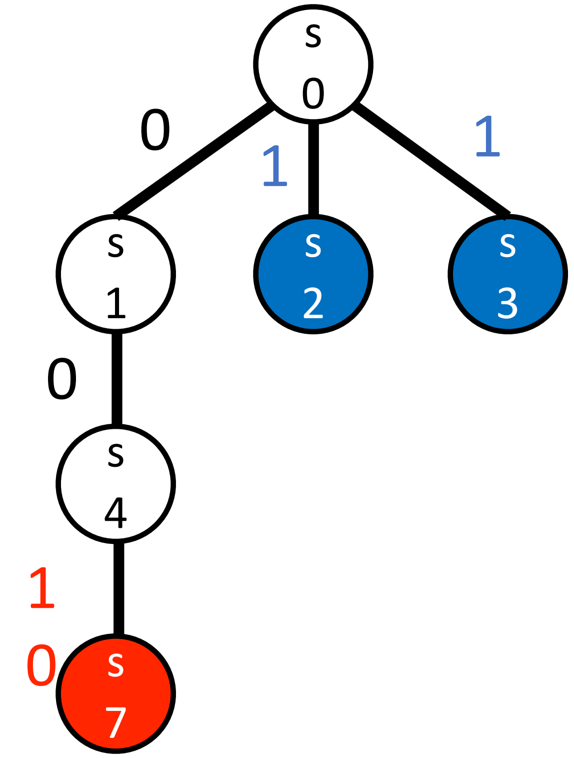
**Sparse reward**

Difficult to find the reward




**Deceptive rewards**

Prone to learn a sub-optimal policy



# Exploitation and Exploration

How do RL agents exploit? Optimization.

$$J(\theta) = \mathbb{E}_{s_t, a_t, s_{t+1} \sim \mathcal{Z}}\left[\left(r(s_t, a_t) + \gamma \max_a Q_{\theta^-}(s_{t+1}, a) - Q_\theta(s_t, a_t)\right)^2\right]$$

$$J(\theta^\pi) = \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta}\left[V_{\theta^V}(s, a)\right]$$

$$J(\theta^\pi) = \mathbb{E}_{s_t \sim \mathcal{Z}}\left[Q_{\theta^Q}(s_t, \pi_{\theta^\pi}(s_t))\right]$$

How do RL agents explore?

?

# RL Exploration Strategies

- **Exploration in deep reinforcement learning**
  - To visit novel states as many as possible
  - To obtain better estimation of the value function

- **Contemporary exploration strategies**
  - **ε-greedy exploration and noise based exploration**
  - **Entropy-based exploration**
  - **Curiosity-based exploration**
  - **Diversity-driven exploration**
  - **Never give up**
  - **Go and explore**

- **Challenges**
  - Latency (the timesteps required for convergence)
  - Efficiency (the states visited by the agent should be as fewer as possible)
  - Easy implementation

# ε-Greedy, Entropy-based, Noise-based Exploration

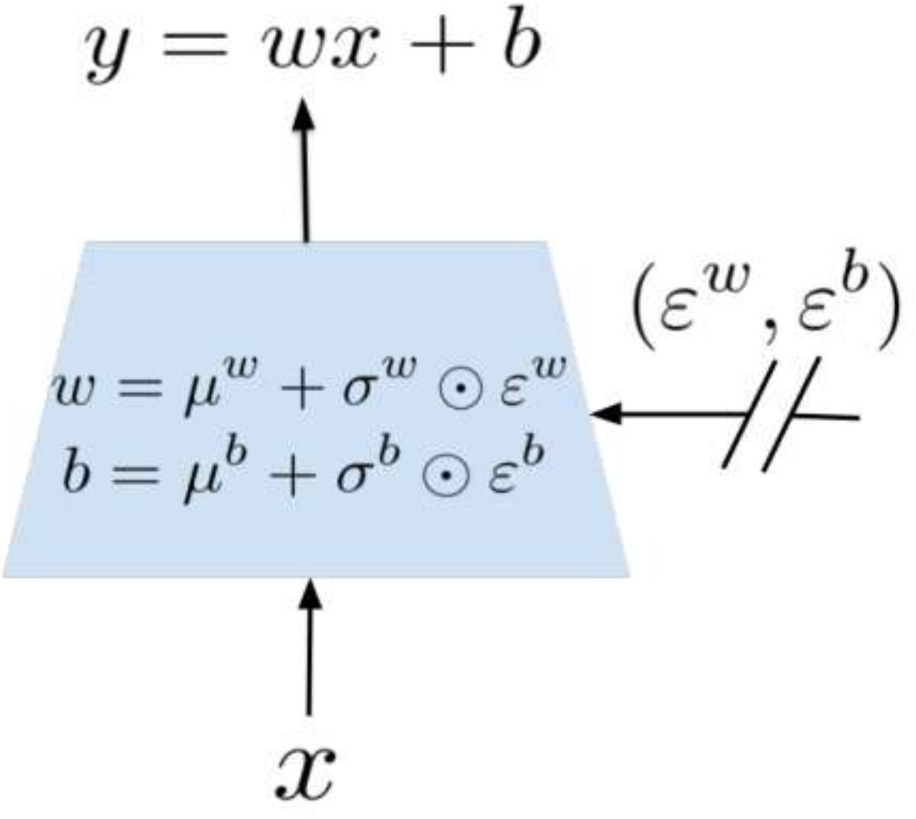
> ε value is linearly decayed

> The mean and variance are fixed

| Stochasticity | Stochasticity | Stochasticity |
|---|---|---|
| **DQN** | **A3C** | **DDPG** |
| $a_t = \begin{cases} \text{argmax}_a Q_{\theta^Q}(s_t, a), & p < \epsilon \\ U(\mathcal{A}), & \text{otherwise} \end{cases}$ <br> $p \sim U([0,1])$ | $a_t \sim \pi_{\theta^\pi}(a \mid s_t)$ <br> $J(\theta^\pi) = \mathbb{E}_{s,a\sim\pi}[V_{\theta^V}(s,a) + H(\pi_{\theta^\pi}(.\mid s))]$ <br> $\mathbb{E}_{s,a\sim\pi}[.] = \mathbb{E}_{s\sim d^{\pi_{\theta^\pi}}, a\sim\pi_{\theta^\pi}}[.]$ | Gaussian noise <br> $a_t = \pi_{\theta^\pi}(s_t) + \mathcal{N}(\mu, \sigma)$ <br> $a_t = \pi_{\theta^\pi}(s_t) + \text{OU}(\mu, \sigma)$ <br> OU noise |
| ε–greedy <br> Sometimes take the best action, sometimes randomly take an action | Add the entropy regularization term to encourage the stochasticity in the policy | Perturb the deterministic action with random noise |

# Noisy Network DQN / A2C

- Parameterizes the weights of neural networks as **probability distributions**



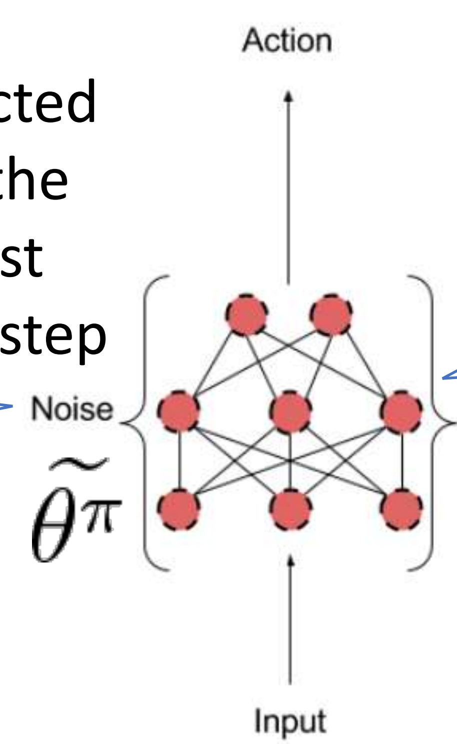
# Parameter Noise DDPG

- They inject noise in the parameters at the beginning of an episode and collect a whole rollout using this perturbed parameters

Take an action at each timestep

Injected at the first timestep

Injected at the beginning of an episode and fixed during that episode




Compute action with the perturbed network, during that episode

$$a_t = \pi_{\widetilde{\theta^\pi}}(s_t)$$

# Diversity-Driven Exploration

The modified loss function encourages the agent **act optimally while differentiating from prior policies**.

Optimality

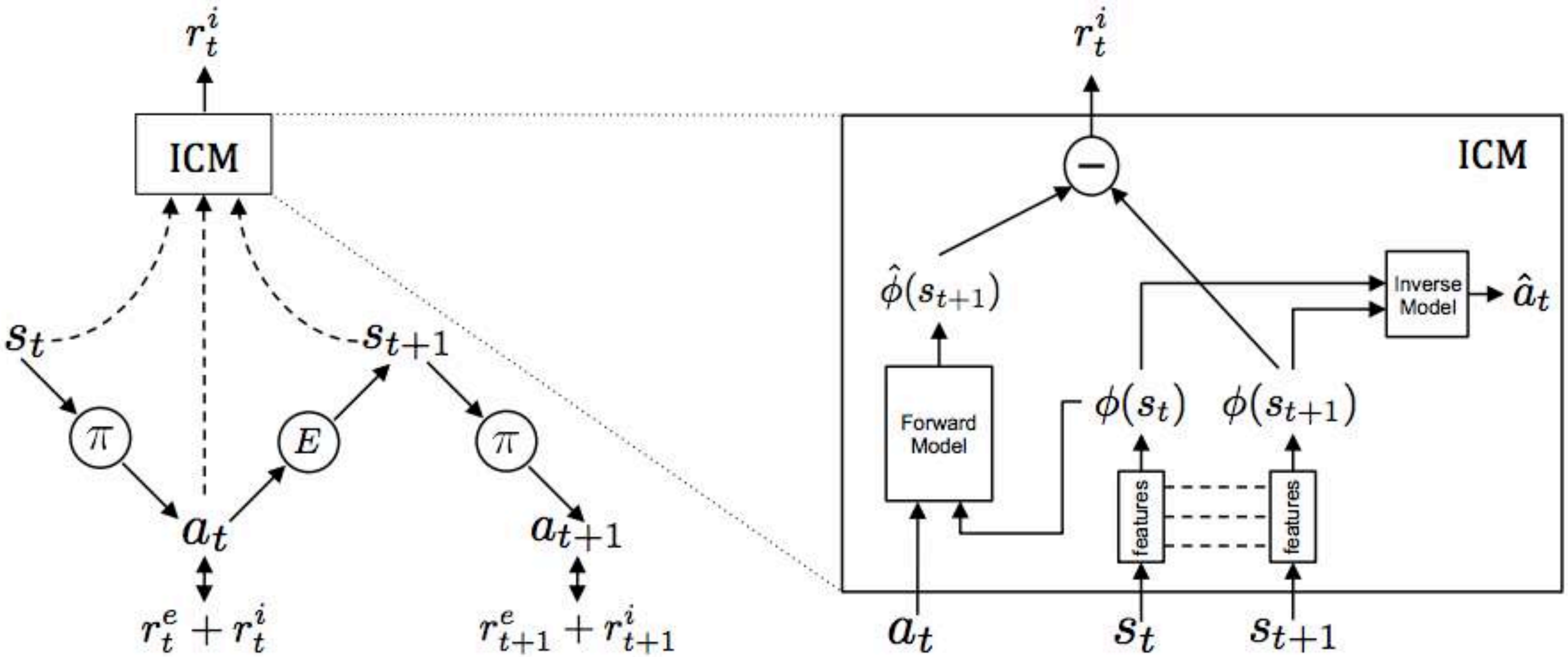
$$L_D = L - \mathbb{E}_{\pi' \in \Pi}[\alpha D(\pi, \pi'))]$$

Diversity

"-" in the loss function indicates "maximization"

# Curiosity-Driven Exploration

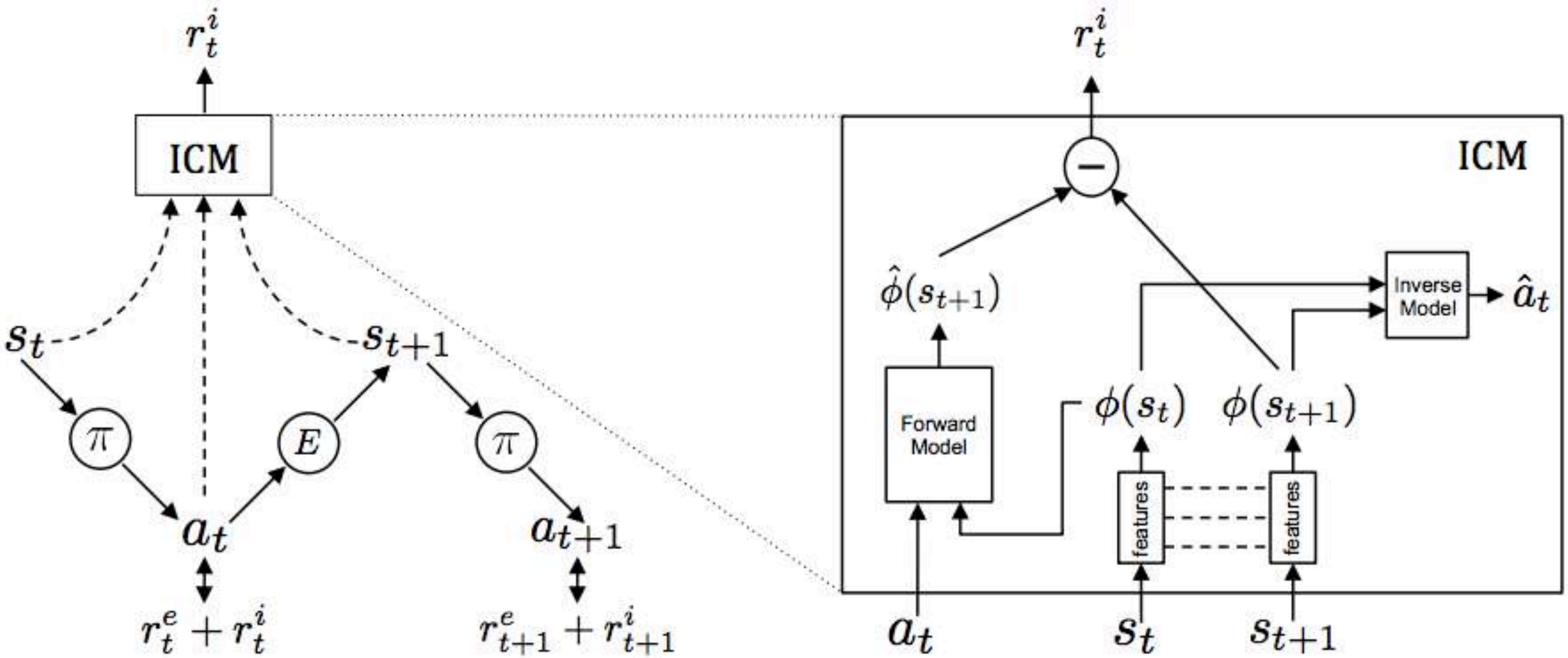
## Architecture




- **Two separate modules: An RL agent and an intrinsic curiosity module (ICM)**
  - ICM contains a forward dynamics model for estimating the novel of states
  - ICM also incorporates an inverse dynamics model for regulating the embeddings
  - The losses of the forward dynamics model serve as the intrinsic rewards for the agent

# Curiosity-Driven Exploration

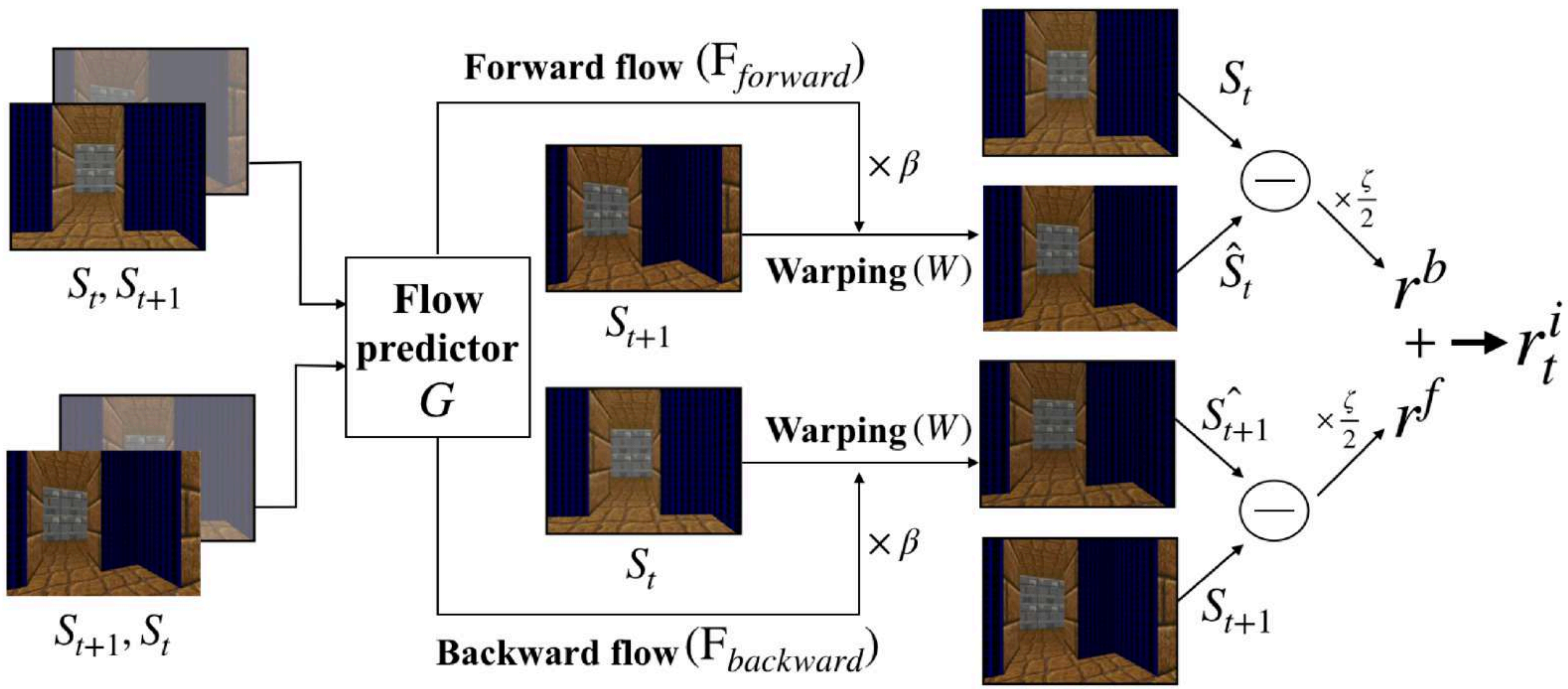
## Challenges




- **Two separate modules: An RL agent and an intrinsic curiosity module (ICM)**
  - ICM contains a forward dynamics model for estimating the novel of states
  - ICM also incorporates an inverse dynamics model for regulating the embeddings
  - The losses of the forward dynamics model serve as the intrinsic rewards for the agent

# Flow-Based Intrinsic Curiosity Module

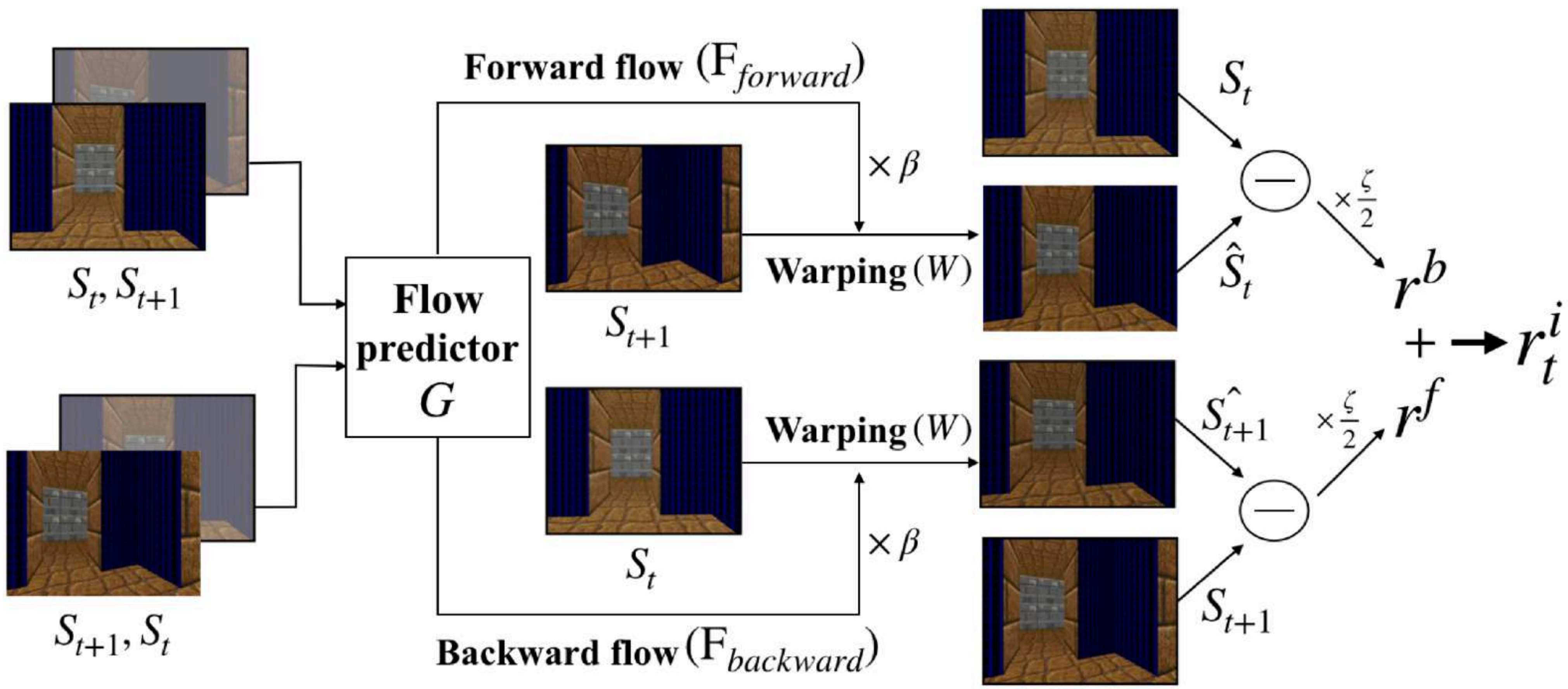
## Architecture




- **The properties of flow-baed intrinsic curiosity model (FICM)**
  - FICM uses optical flow prediction errors as the novelty of states
  - Optical flows are estimated in dual directions: Forward and backward
  - The differences between the warped frames and the actual frames serve as the intrinsic reward

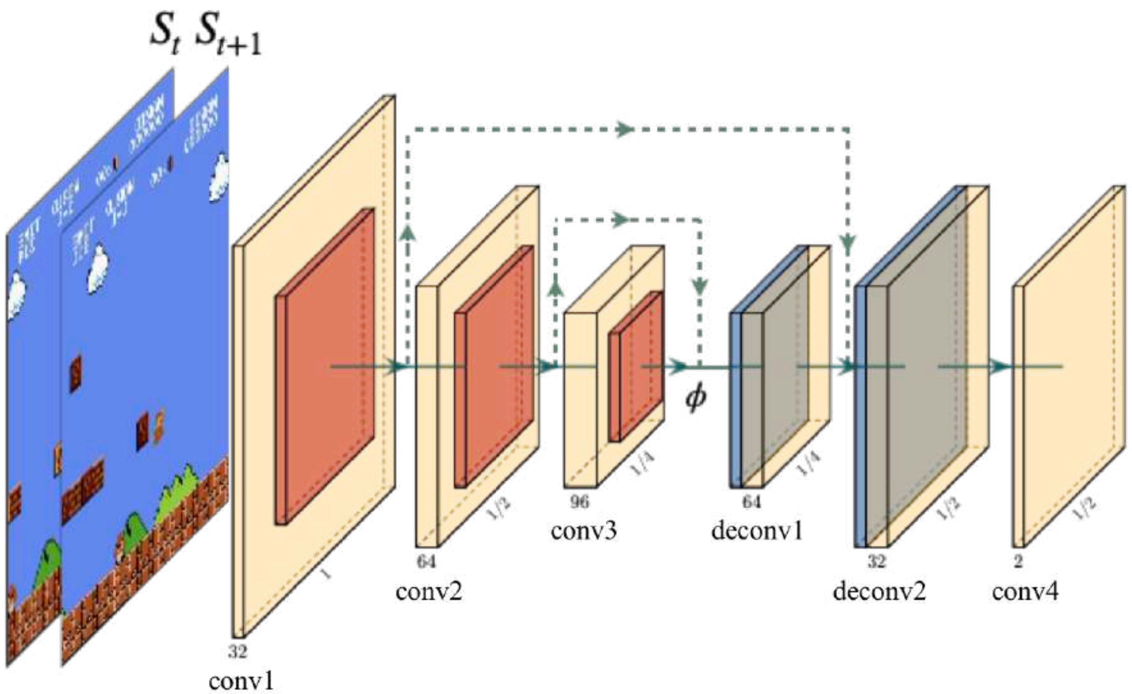
# Flow-Based Intrinsic Curiosity Module

## Architecture

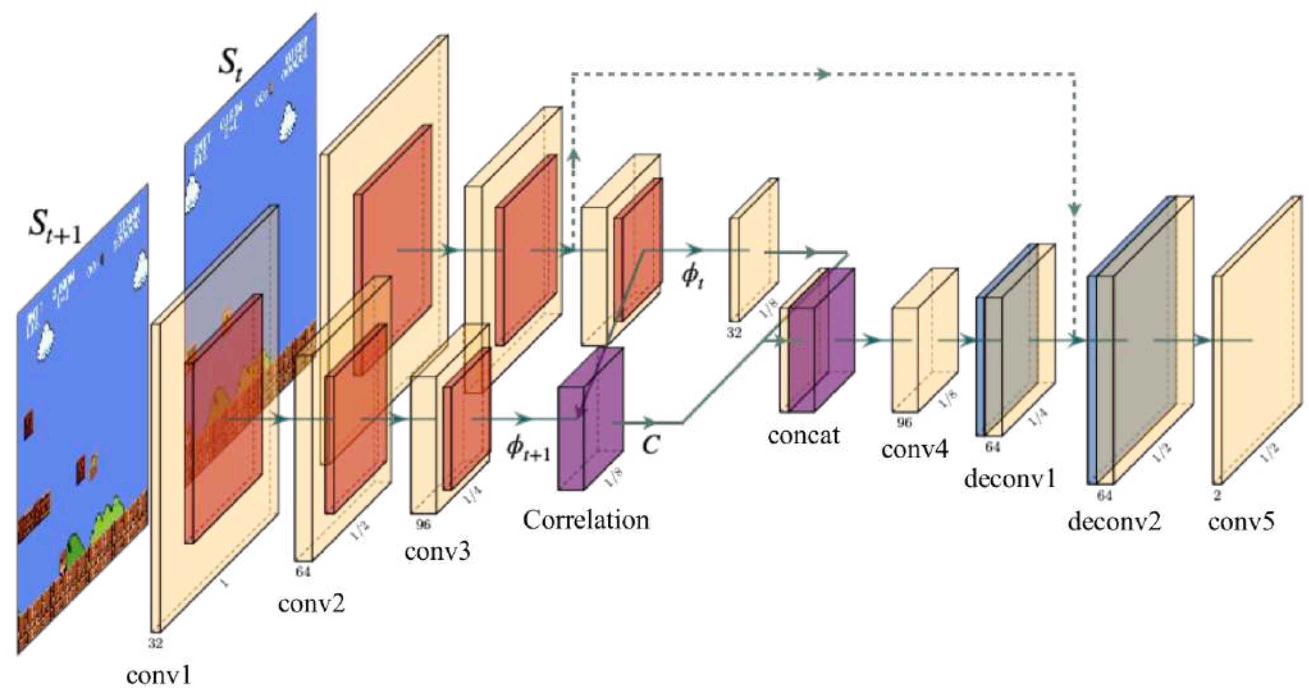



- **The properties of flow-baed intrinsic curiosity model (FICM)**
  - FICM uses optical flow prediction errors as the novelty of states
  - Optical flows are estimated in dual directions: Forward and backward
  - The differences between the warped frames and the actual frames serve as the intrinsic reward

# Flow-Based Intrinsic Module
# Implementations




(a) The flow predictor of FICM-S




(b) The flow predictor of FICM-C

- **Two different implementations of flow predictors are provided**
  - Different implementations validate the generalizability of FICM
  - FICM only requires two states as its input, instead of eight as in ICM
  - The two implementations are based on FlowNet 2.0 modules

# Random Network Distillation (RND)

## Source of prediction errors

1. **Amount of training data** — Prediction error is high where few similar examples were seen by the predictor (epistemic uncertainty)
2. **Stochasticity** — Prediction error is high because the target function is stochastic (aleatoric uncertainty). Stochastic transitions are a source of such error for forward dynamics prediction
3. **Model misspecification** — Prediction error is high because necessary information is missing, or the model class is too limited to fit the complexity of the target function
4. **Learning dynamics** — Prediction error is high because the optimization process fails to find a predictor in the model class that best approximates the target function

# Random Network Distillation (RND)

Factor (2) ***Stochasticity*** can result in the Noisy TV problem

RND aims to address the stochasticity issue, because in the original ICM, **the target network can be chosen to be deterministic**.

# Random Network Distillation (RND)

A different approach where **the prediction problem is randomly generated**

- This involves two neural networks:
  - A fixed and randomly initialized ***target*** network which sets the prediction problem
  - A ***predictor*** network trained on the data collected by the agent.

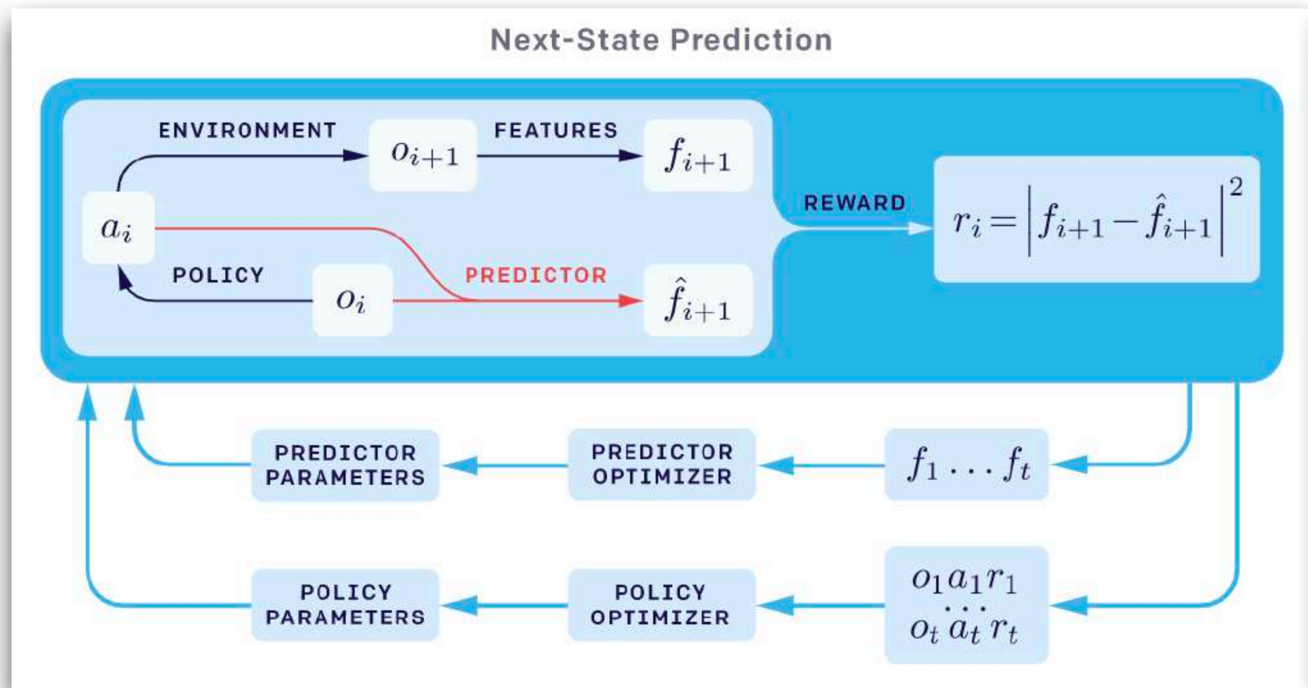
The ***target*** network takes an observation transforms it to an embedding

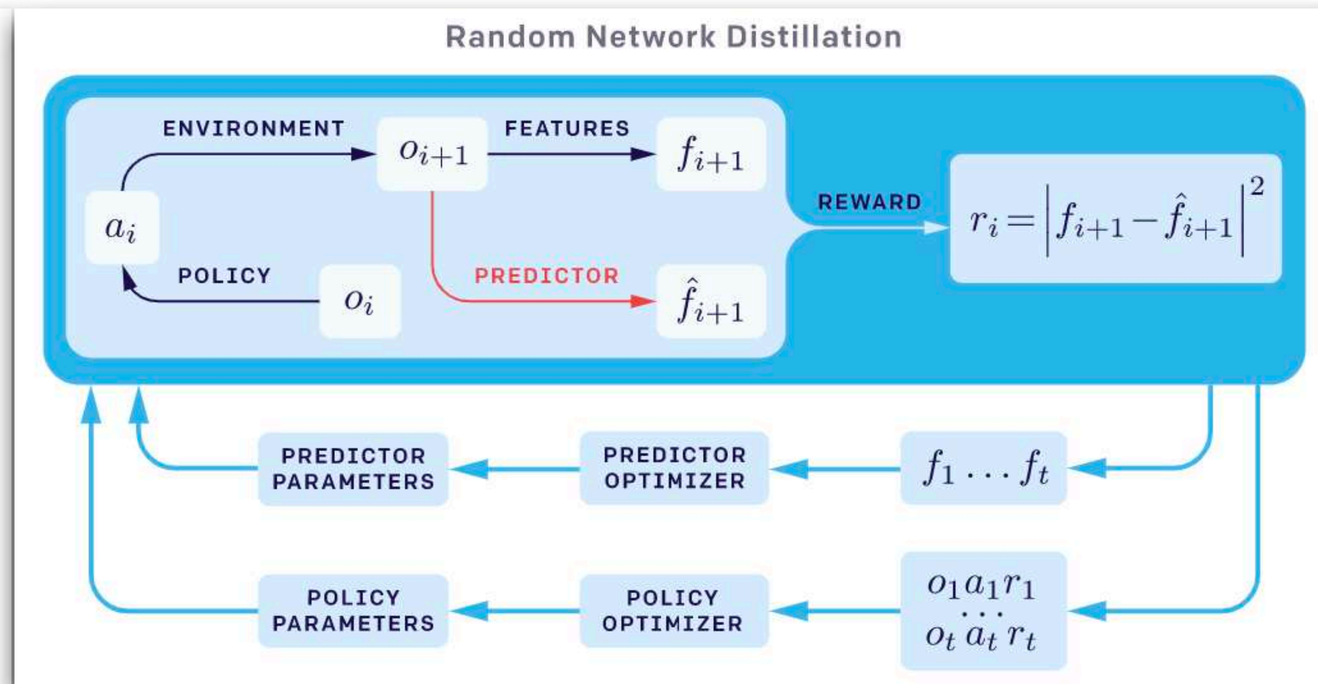
$$f : O \to \mathbb{R}^k$$

The ***predictor*** network $\hat{f} : O \to \mathbb{R}^k$ is trained by gradient descent to minimize the expected MSE $||\hat{f}(x;\theta) - f(x)||^2$ with respect to its parameter $\theta_{\hat{f}}$.

# Random Network Distillation (RND)

## Comparison of next-state prediction with RND



# Never Give Up: Learning Directed Exploration Strategies

- First reinforcement learning agent able to solve hard exploration games by **learning a range of directed exploratory policies**
- First algorithm to achieve non-zero rewards (with a mean score of 8,400) in the game of **Pitfall!** without using demonstrations or hand-crafted features
- **NGU** o jointly learns a family of policies, with various degrees of exploratory behavior
    - The learning of the exploratory policies can be thought of as a **set of auxiliary tasks** that can help build a shared architecture that continues to develop even in the absence of extrinsic rewards

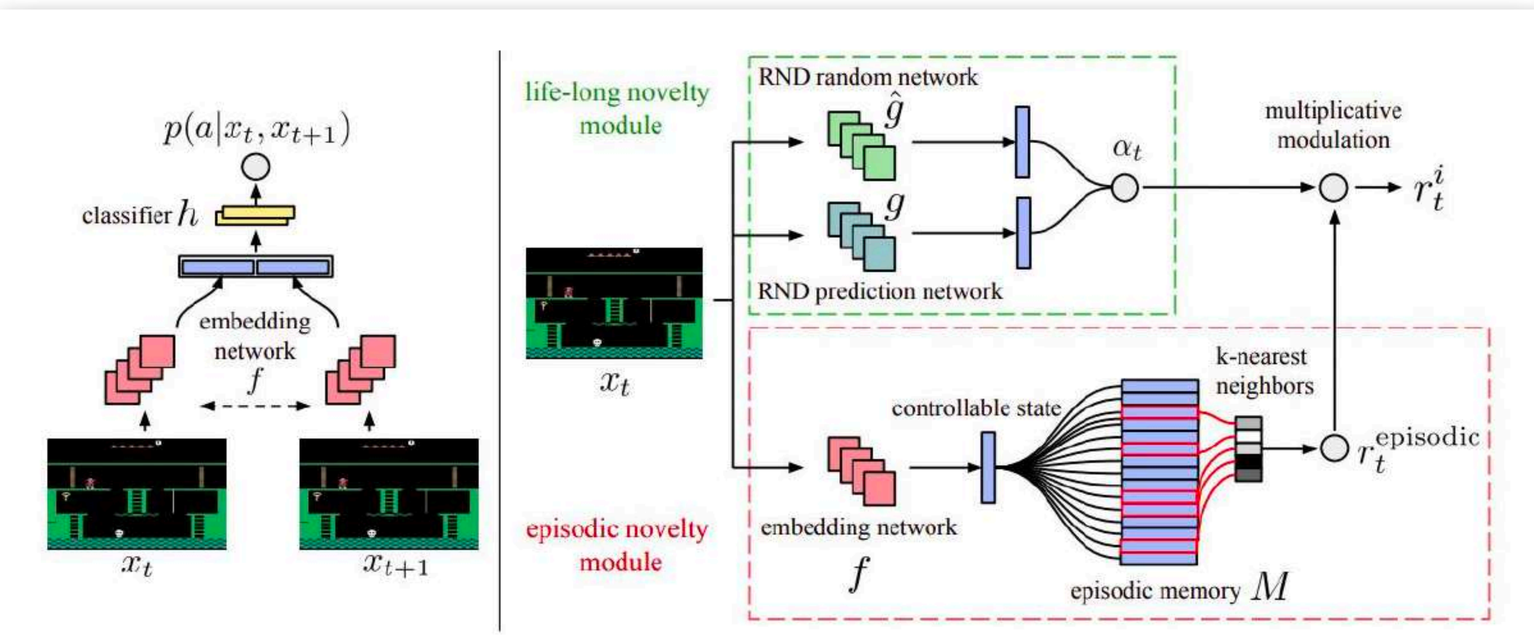
# Never Give Up: Learning Directed Exploration Strategies

## Contributions

- The main contributions of **NGU** consists of the following items:
- An exploration bonus combining ***life-long*** and ***episodic*** novelty to learn exploratory strategies that can **maintain exploration** throughout the agent's training process (to **never give up**)
    - Developing intrinsic motivation rewards that encourage an agent to explore and visit as many states as possible by providing more dense "internal" rewards for novelty-seeking behaviors
    - Long-term ***life-long*** novelty rewards encourage visiting many states throughout training, across many episodes
    - Short-term ***episodic*** novelty rewards encourage visiting many states over a short span of time (e.g., within a single episode of a game)

# Never Give Up: Learning Directed Exploration Strategies

## The never-give-up intrinsic reward generation architecture




- The network is trained based on the augmented reward

$$r_t = r_t^e + \beta r_t^i$$

- The intrinsic reward $r_t^i$ satisfies three properties:
    - It rapidly discourages revisiting the same state within the same episode
    - It slowly discourages visits to states visited many times across episodes

# Go-Explore: A New Approach for Hard-Exploration Problems

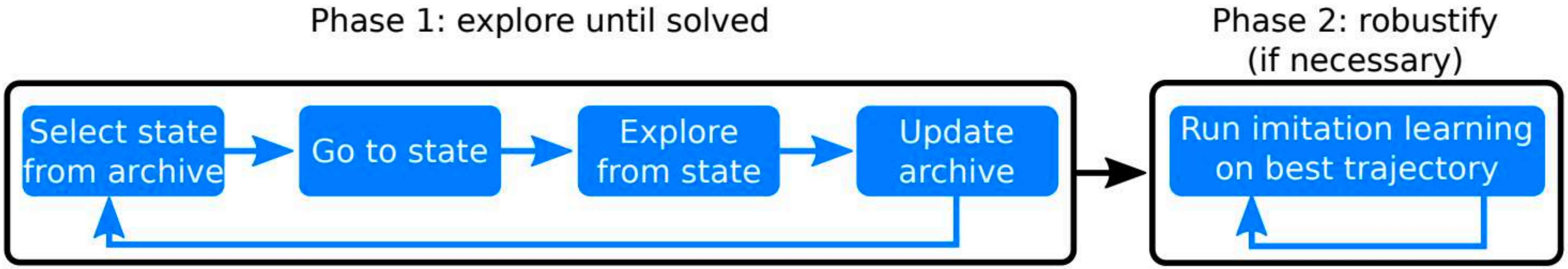
1. **Phase 1**

   **Go** — Go to the selected state

   **Explore** — Start explore from that state

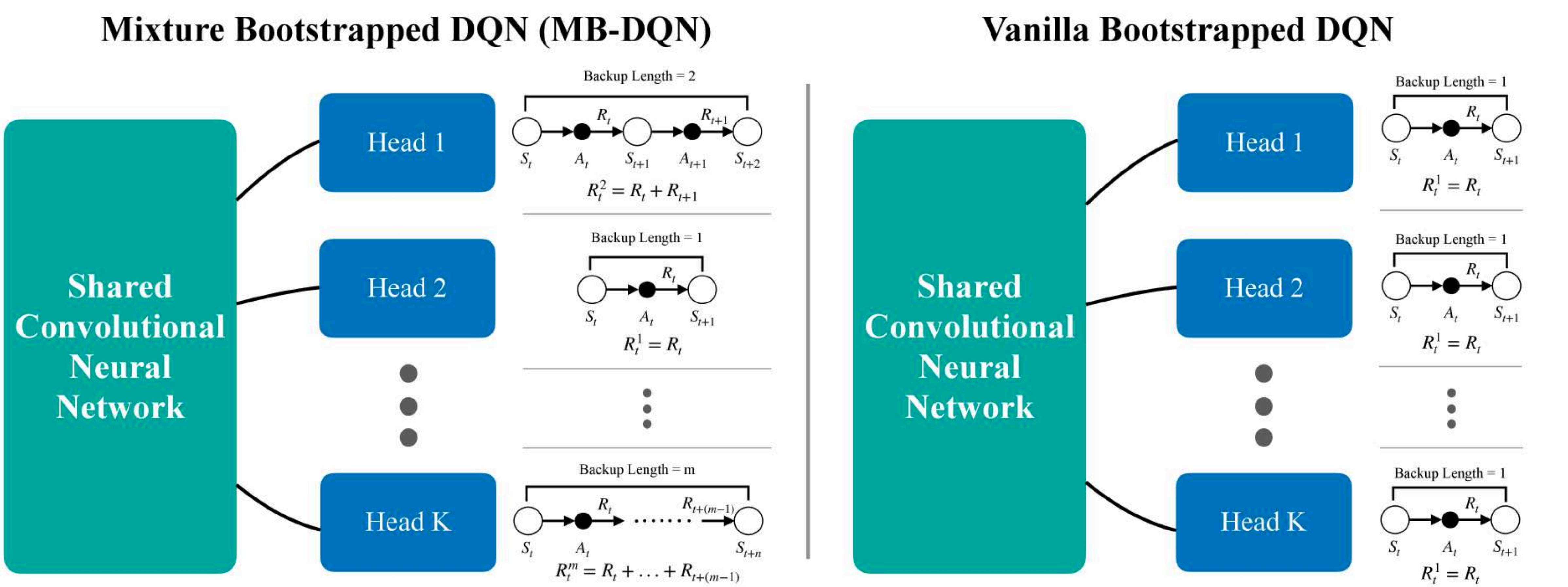
2. **Phase 2**

   **Robustify** — Use the trajectory collect from Phase 1.



# Bootstrapped DQN

- Multiple bootstrapped heads are used for evaluating the value function
- Each bootstrapped head is trained independently
- MB-DQN extends the concept of Bootstrapped DQN for multiple backup lengths



# Agenda

- Reinforcement Learning Backgrounds
- DRL Techniques
- Exploration
- Robotic Applications
- Summary

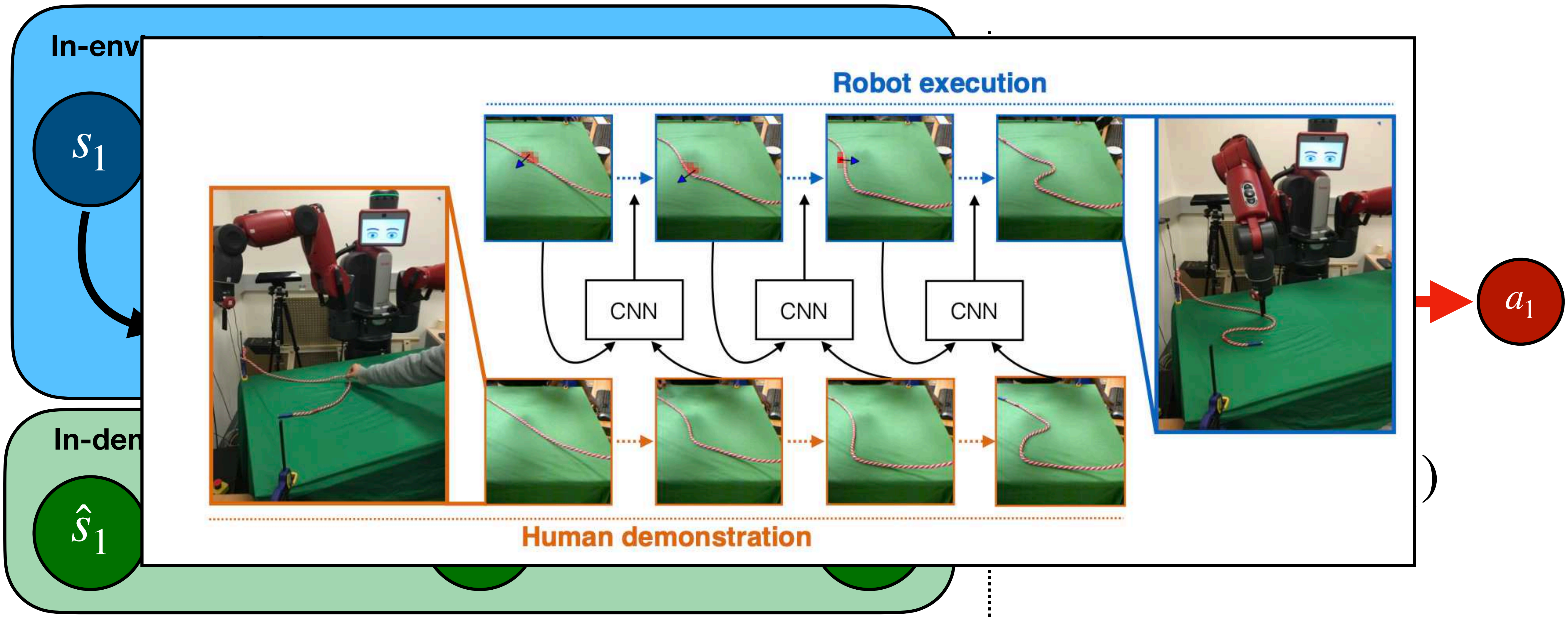
MLSS 2021 TAIPEI

ELSA LAB

http://elsalab.ai

# Inverse Dynamics Model (IDM) for Robotic Applications

Given a desired motion (i.e. a list of states, $\tau = [\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_T]$), a robot can accomplish this motion by inferring the actions (i.e. a list of torques, $[a_1, a_2, \cdots, a_T]$) by IDM





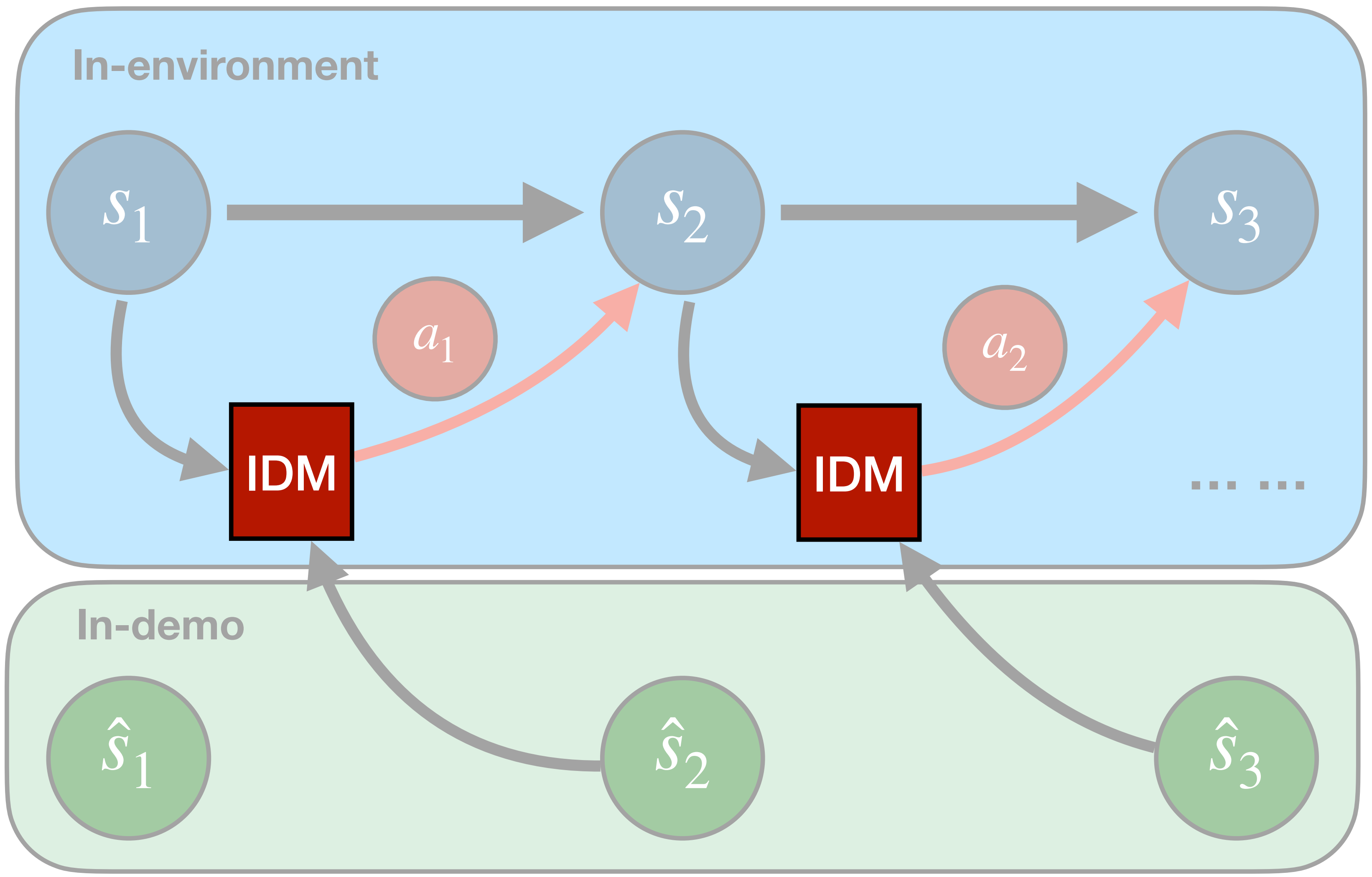
Nair, Ashvin, et al. "Combining self-supervised learning and imitation for vision-based rope manipulation." *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

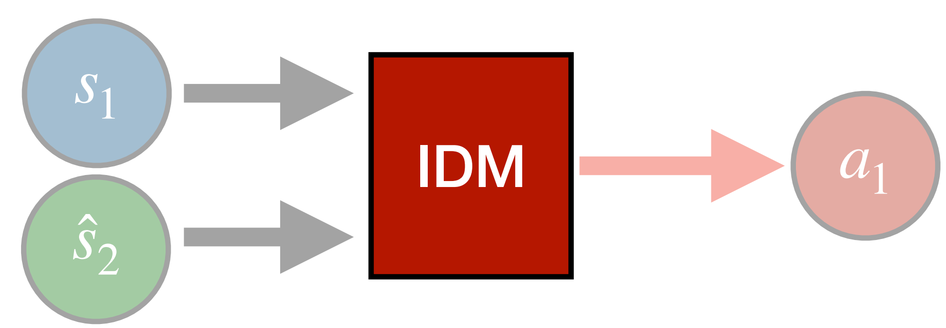
Agrawal, Pulkit, et al. "Learning to poke by poking: Experiential learning of intuitive physics." *Advances in Neural Information Processing Systems*. 2016.

Pathak, Deepak, et al. "Zero-Shot Visual Imitation." (2018) takes it as an image-based IDM

# How to Obtain an IDM?

Data-driven modeling (e.g., neural network)




$$a_t = I(s_t, \hat{s}_{t+1})$$

In this work, we take IDM as a Long Short Term Memory (LSTM)*

*Pathak, Deepak, et al. "Zero-Shot Visual Imitation." (2018) takes it as an image-based IDM

# Training Data Collection

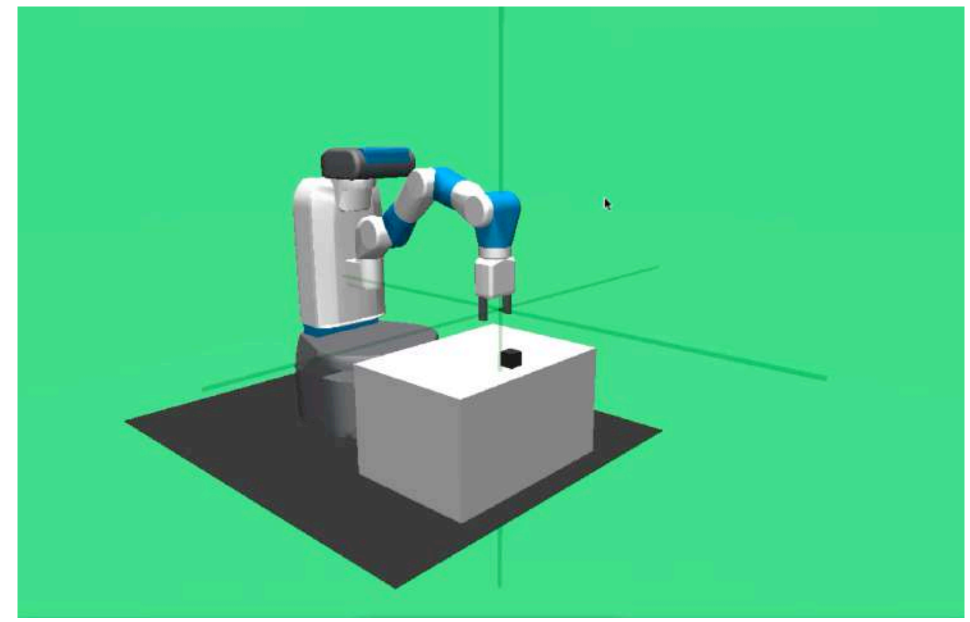
**Human demonstration**

(from Google AI blog)

**Pros: high-quality data (including complex motions)**

**Cons: too much human effort**

**Random exploration (by robots) [1, 2]**



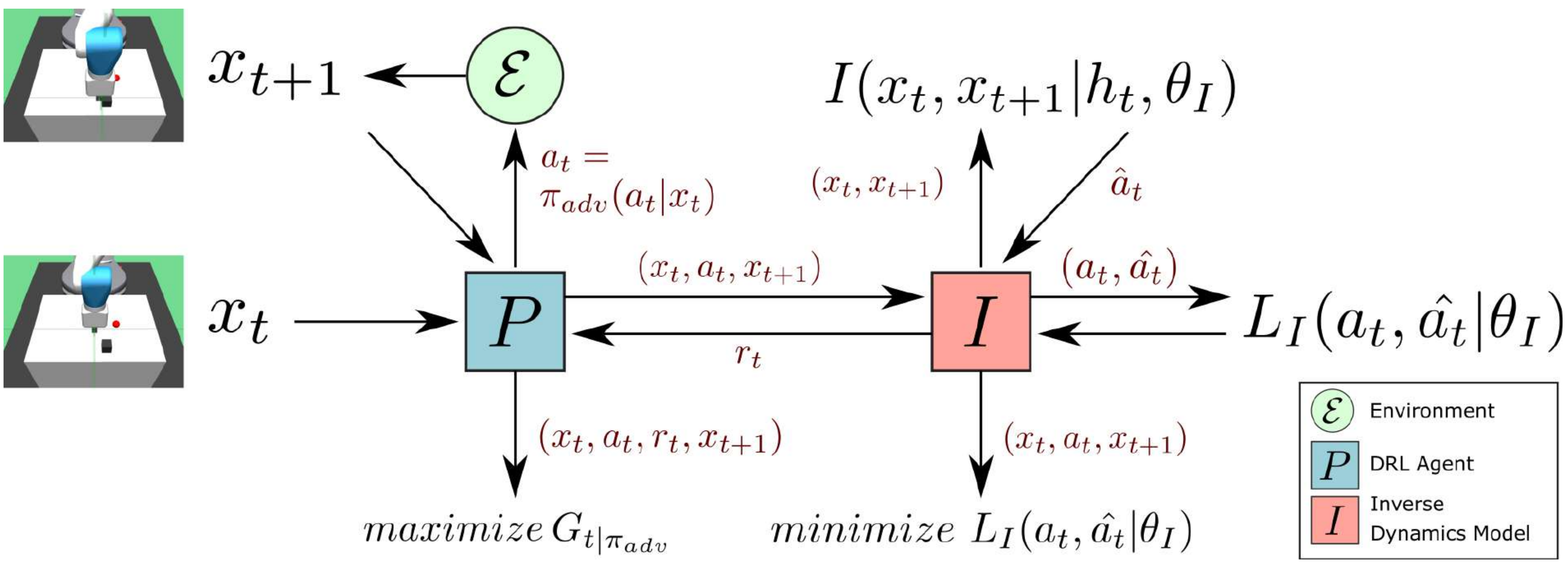
**Pros: zero human effort**

**Cons: low-quality data (no complex motions)**

[1] Nair, Ashvin, et al. "Combining self-supervised learning and imitation for vision-based rope manipulation." *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.
[2] Agrawal, Pulkit, et al. "Learning to poke by poking: Experiential learning of intuitive physics." *Advances in Neural Information Processing Systems*. 2016.

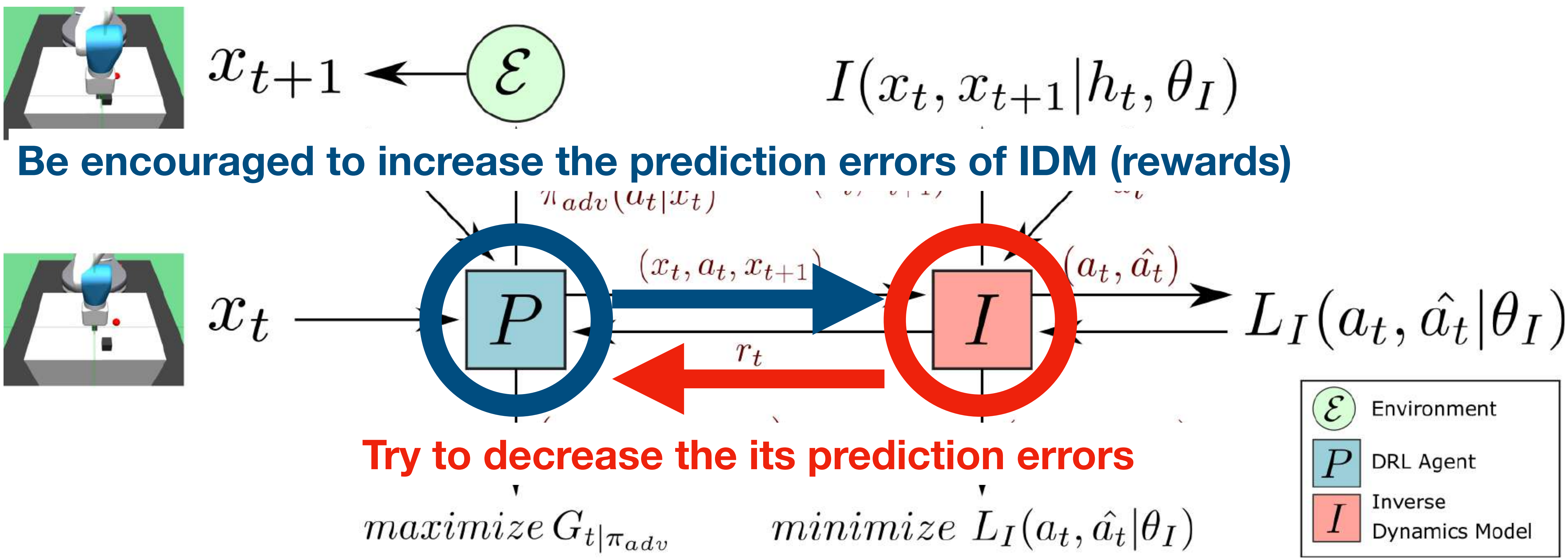
# Adversarial Active Exploration

We train a **reinforcement learning (RL)*** agent
to collect **non-trivial** training data (i.e. complex motions) for IDM



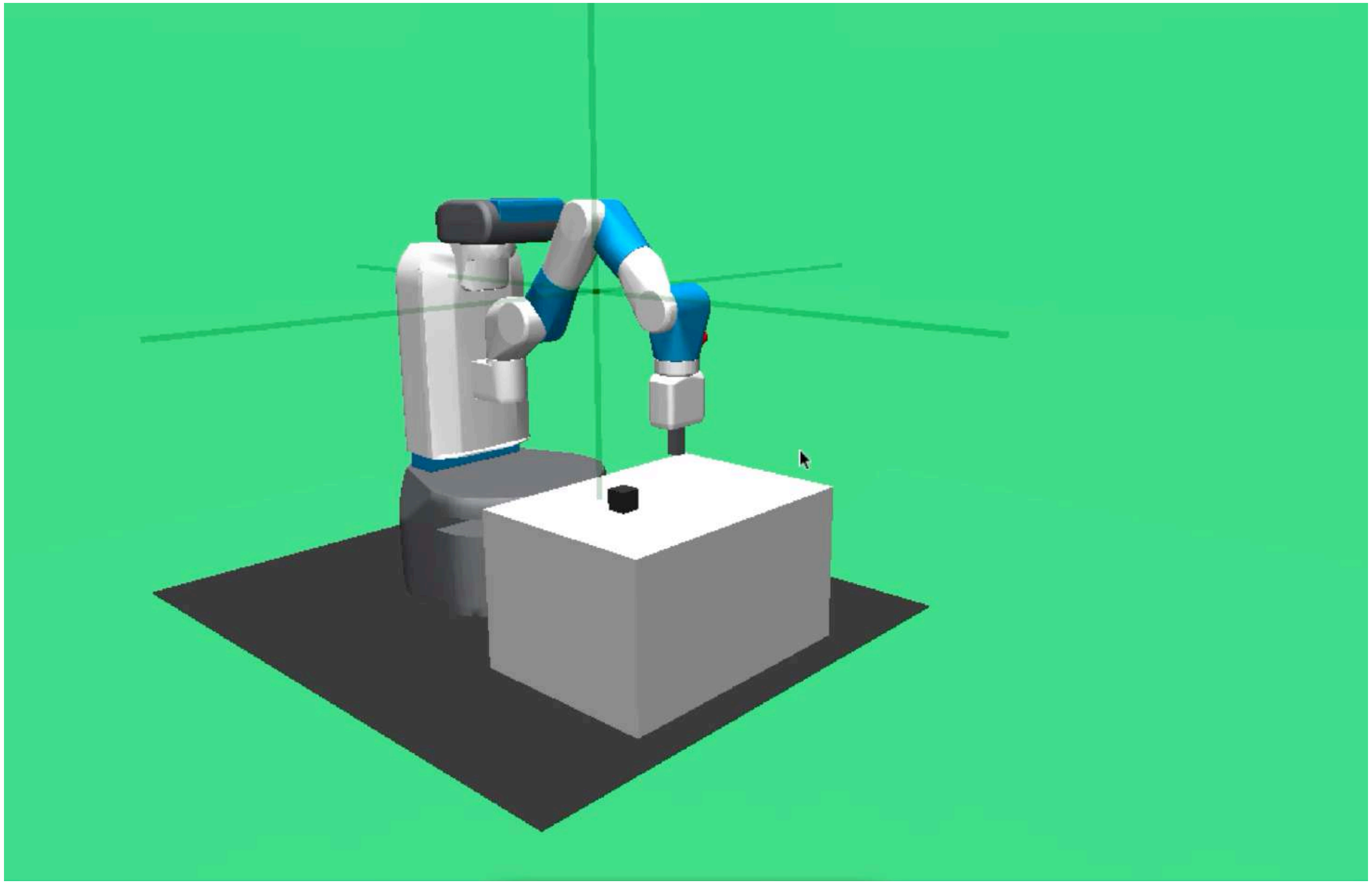

*Here we use Proximal Policy Optimization (PPO)

# Adversarial Active Exploration




- The competitive relationship creates a **curriculum** to continually improve both sides
- As a result, there are a lot of **complex motions** in the collected dataset
- Also, our method **doesn't rely on human supervision**

# Adversarial Active Exploration



# TRAIN A ROBOT?

# Traditional Way — Google's Approach


**Time Consumption**
3000 robot-hours of practice

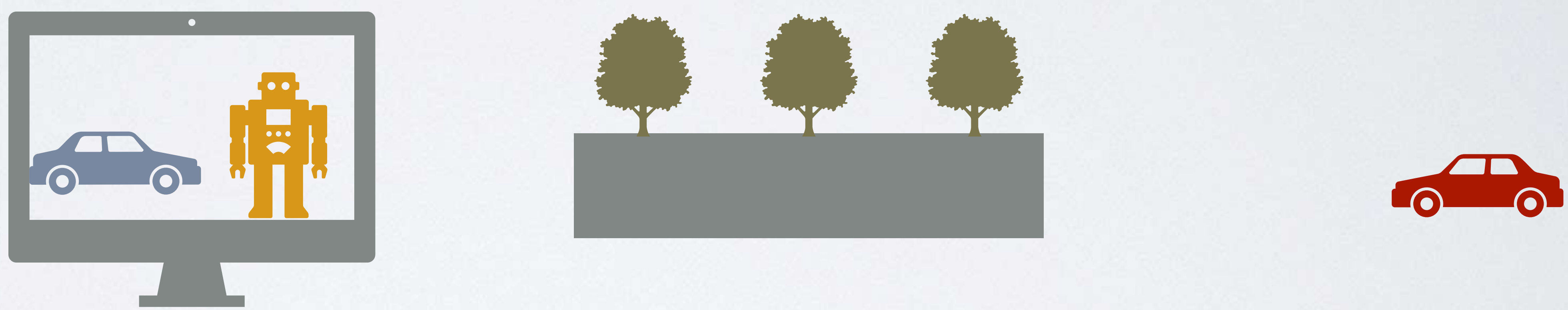
**Danger**
What if we train an autonomous car in real-world?

**Money**
We need to buy lots of robotic arms

**Potential Damage**
Bump into wall? Or fall into water!

Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection, Google, Mar. 2016

# Modular Architecture for Virtual-to-Real Deep Reinforcement Learning



# Advantage of Virtual World

**SAFETY**

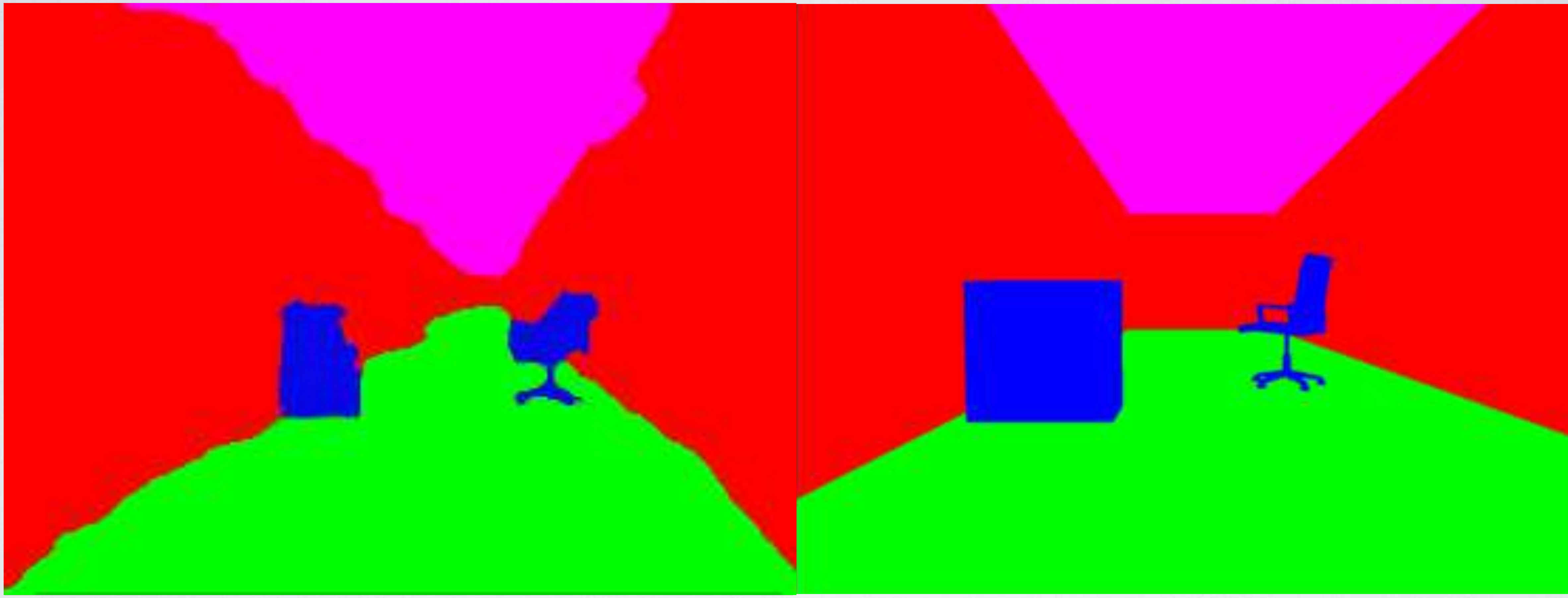
**SPEED**

**MONEY SAVING**

# Gap Between Virtual and Real

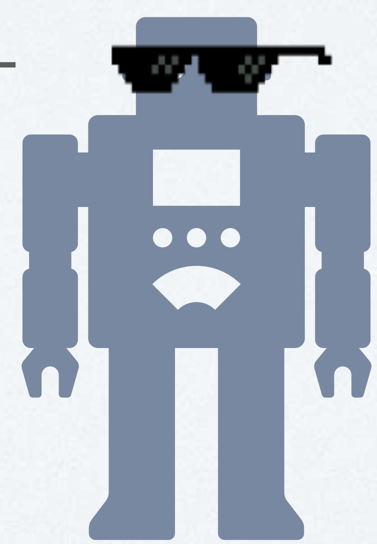
**Light Texture Shape Shadow**

**are totally different!**

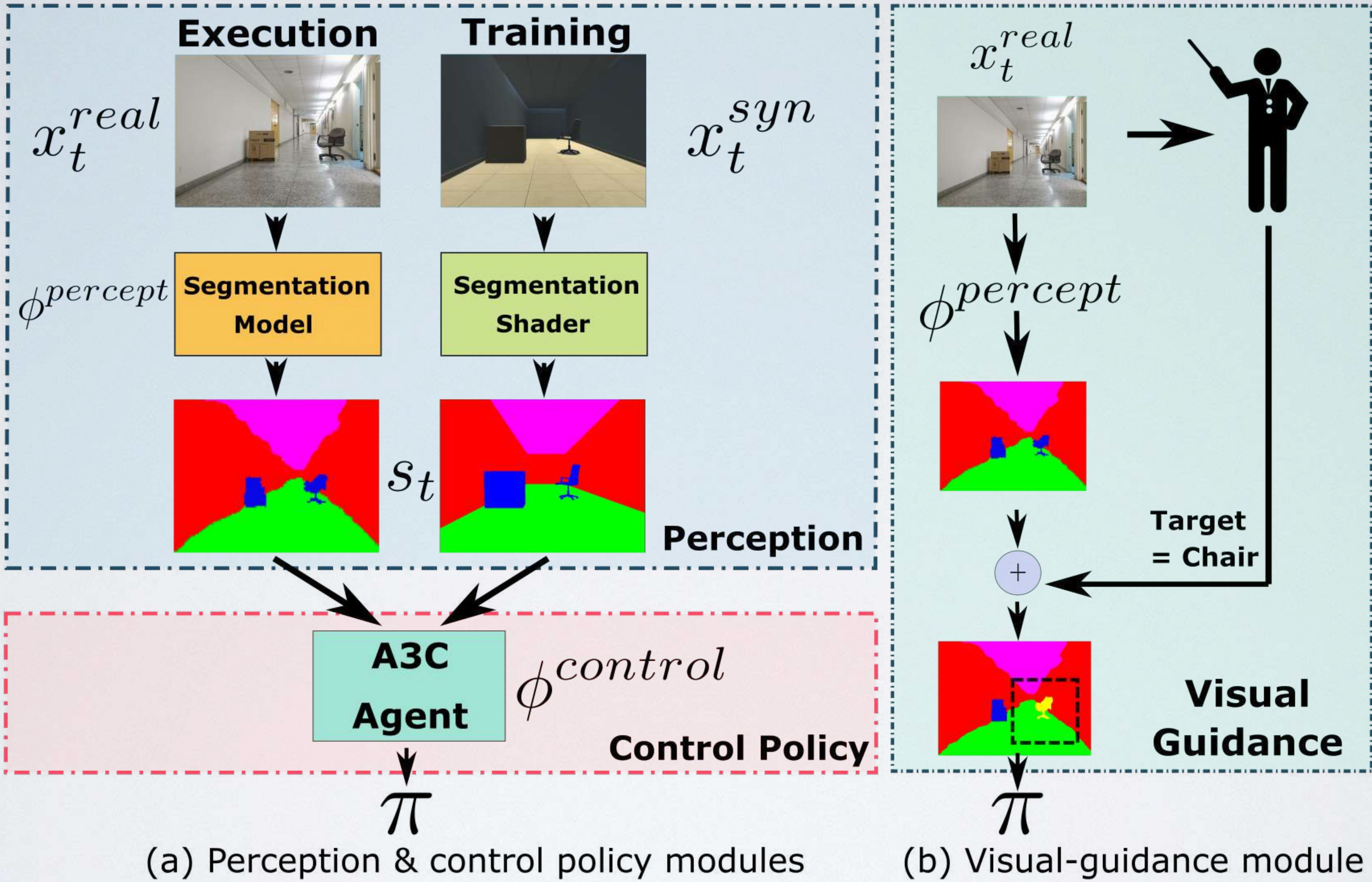


**not familiar** **familiar**

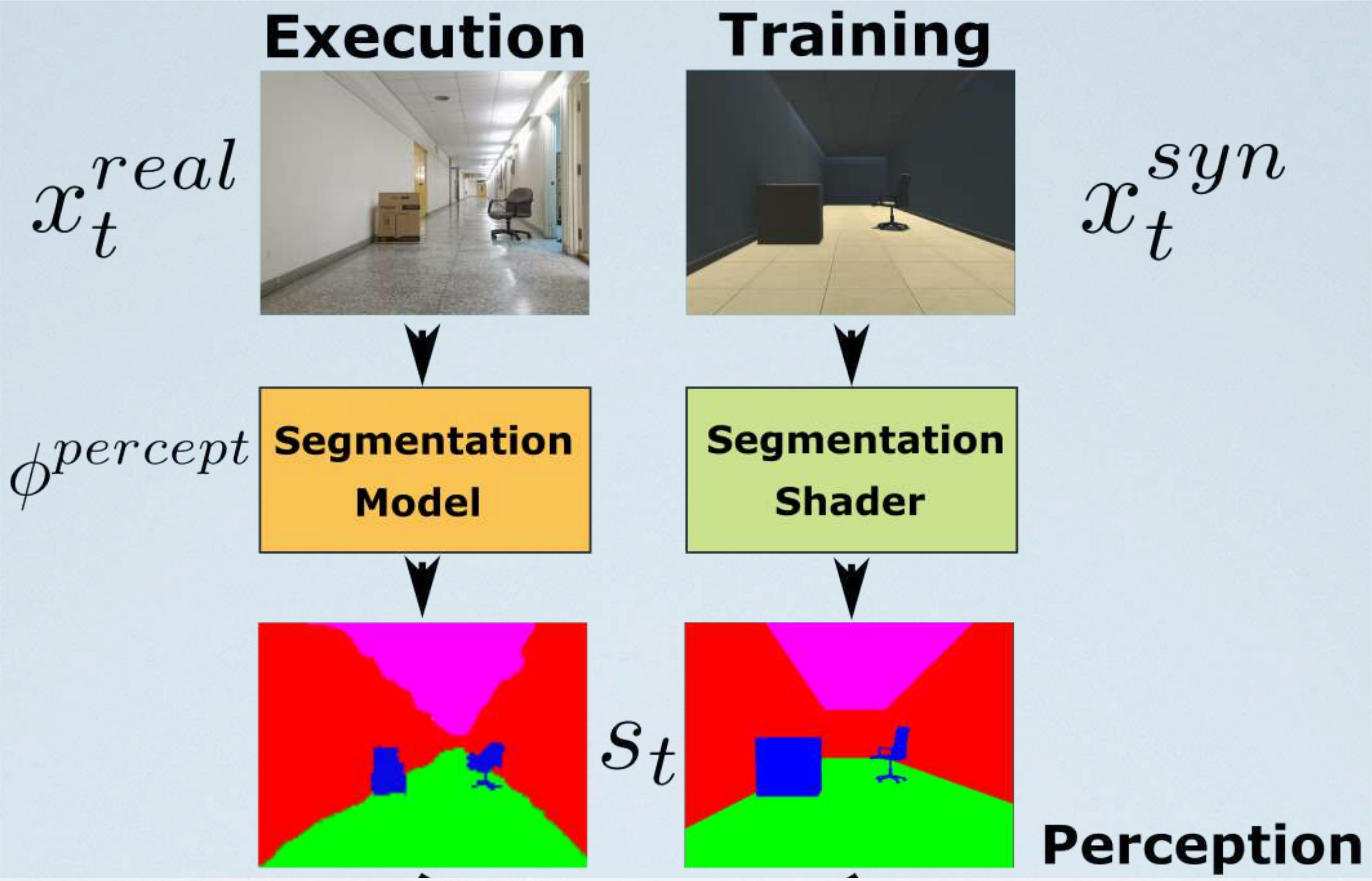
**Segmentation Model** ______



“Virtual-to-real: Learning to control in visual semantic segmentation”, IJCAI 2018




"Virtual-to-real: Learning to control in visual semantic segmentation", IJCAI 2018




# PERCEPTION MODEL

Image Semantic Segmentation

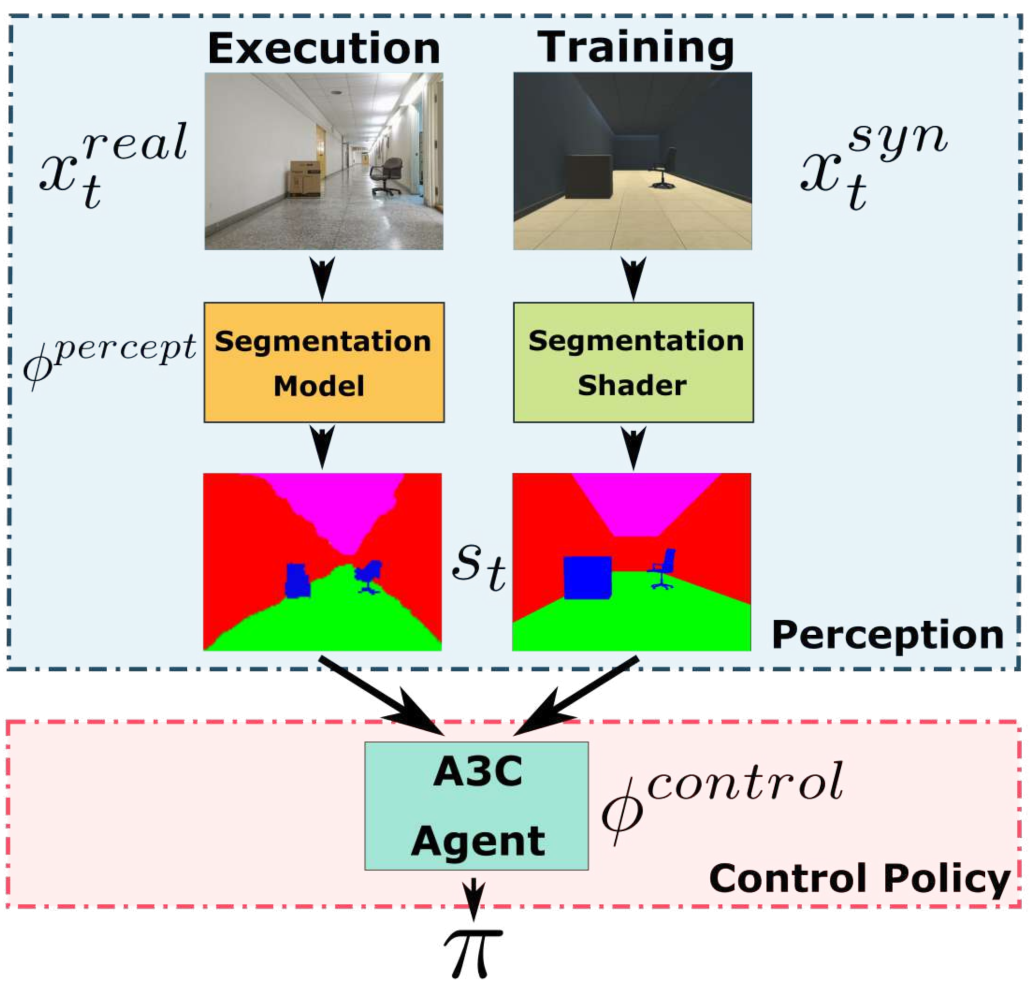
# Virtual-to-Real: Learning to Control in Visual Semantic Segmentation

**NVIDIA Jetson Developer Challenge Champion Grand Prize, IJCAI 2018 Full Paper, and GTC 2018 Poster**

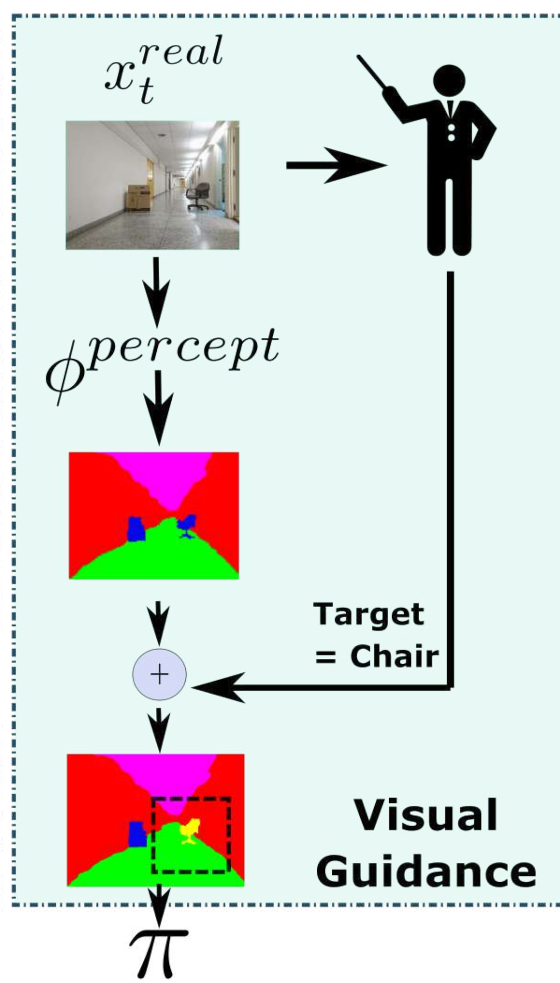
**Video Demonstration:** https://youtu.be/_OqdnG4AIl8

**Official Project Description:** https://tinyurl.com/y2dl7skl

**Paper Link:** https://www.ijcai.org/Proceedings/2018/0682.pdf




(a) Perception & control policy modules

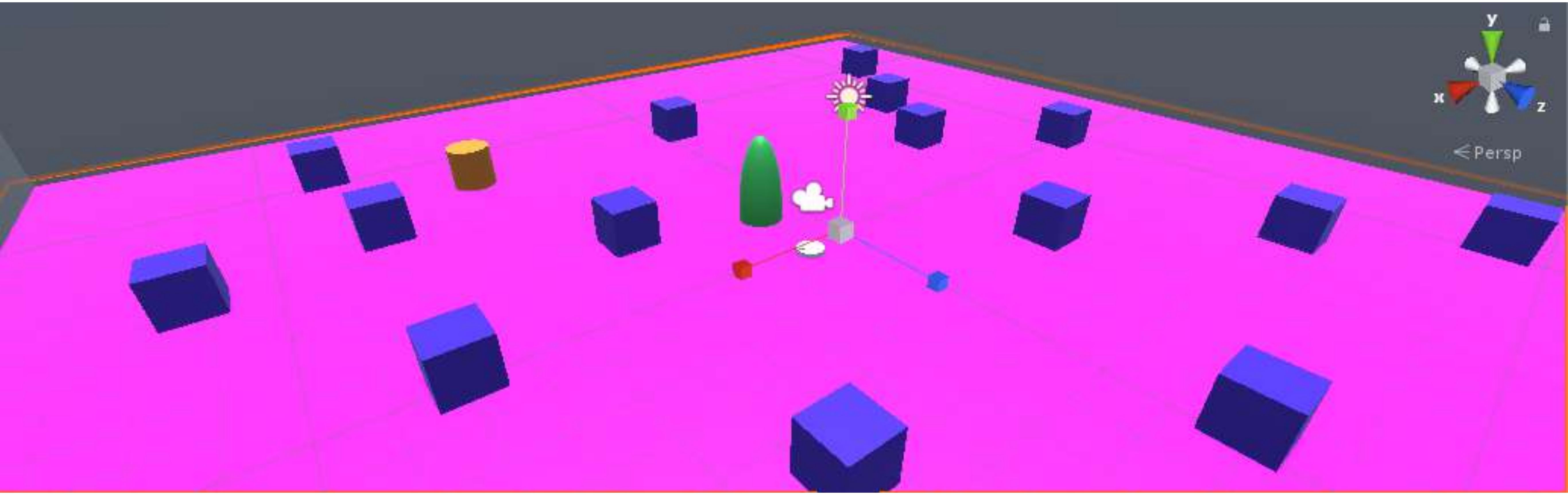


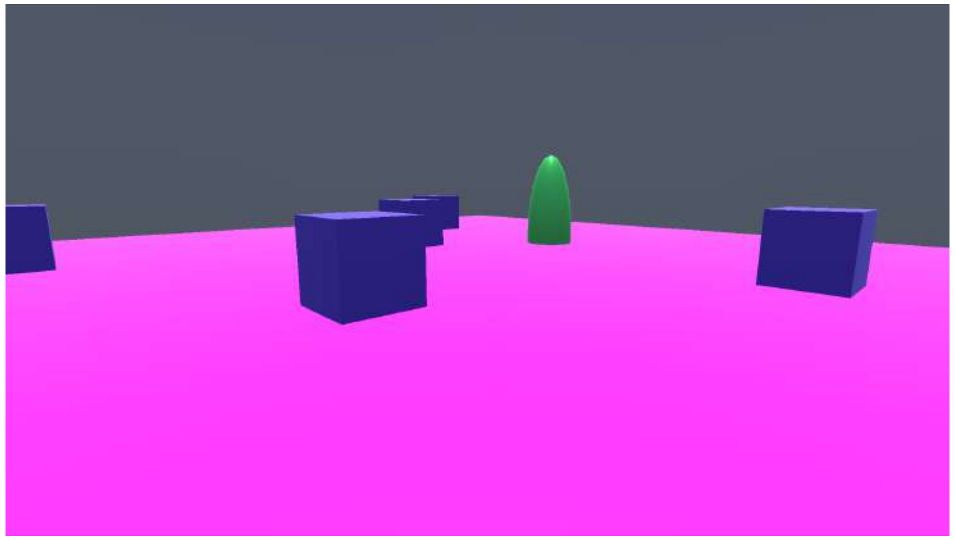

(b) Visual-guidance module

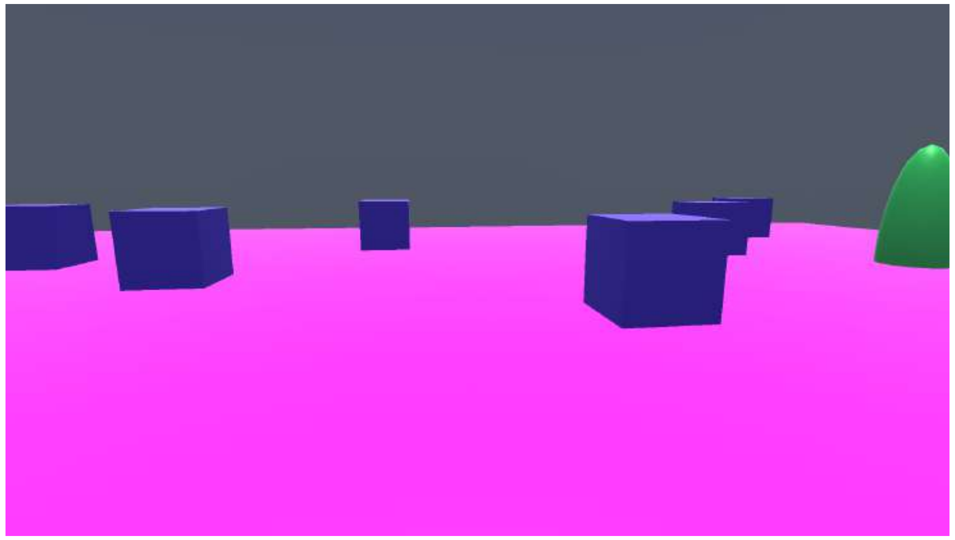
# Simulation Environments

unity ✗ ML-Agent

# Example Environments



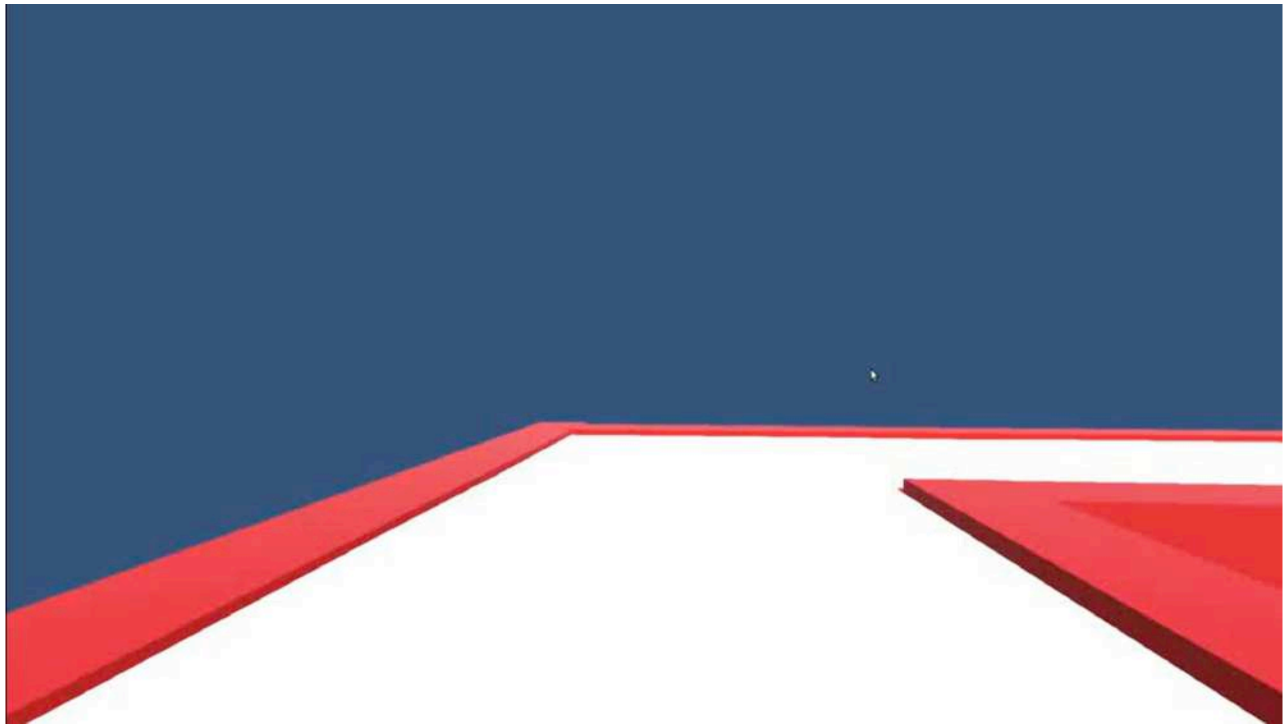







By Prof. Chun-Yi Lee at Dept. Computer Science, National Tsing Hua University

# A3C Agents in Virtual Environments




- A3C agent is able to move to its destination by itself
- The color palette has to be the same as those used for semantic segmentation

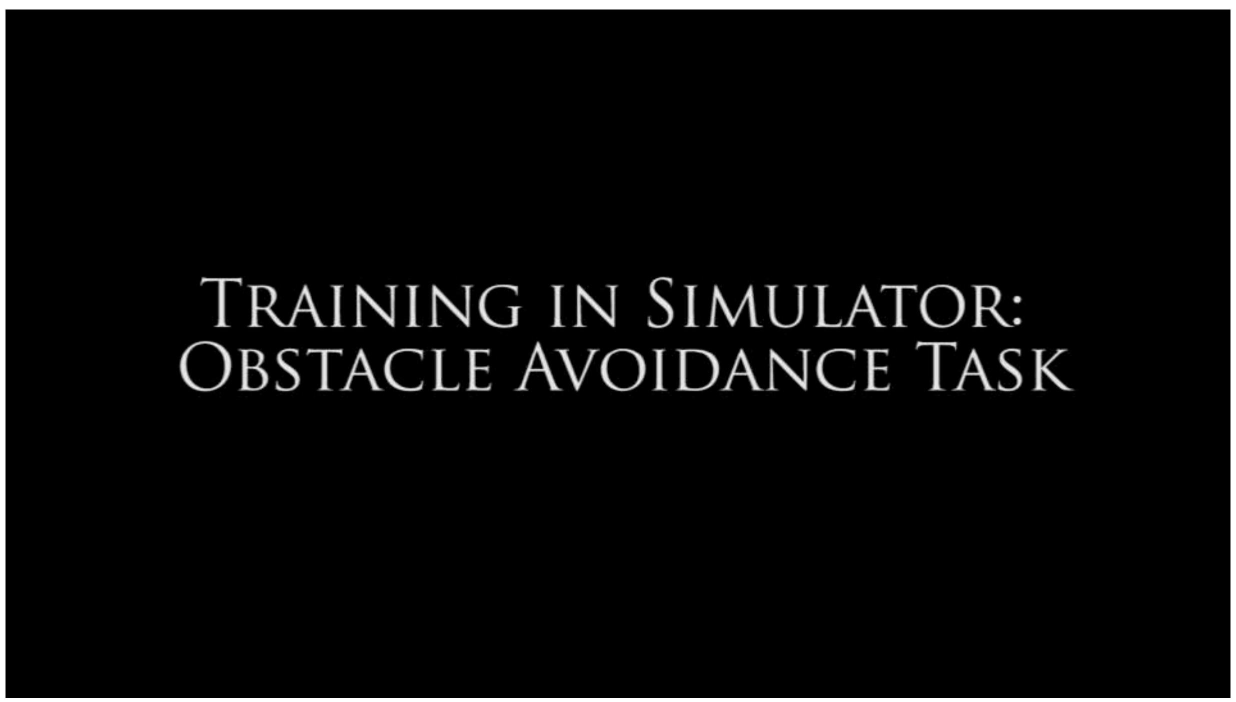


By Prof. Chun-Yi Lee at Dept. Computer Science, National Tsing Hua University

# Simulated environments

## Visualization

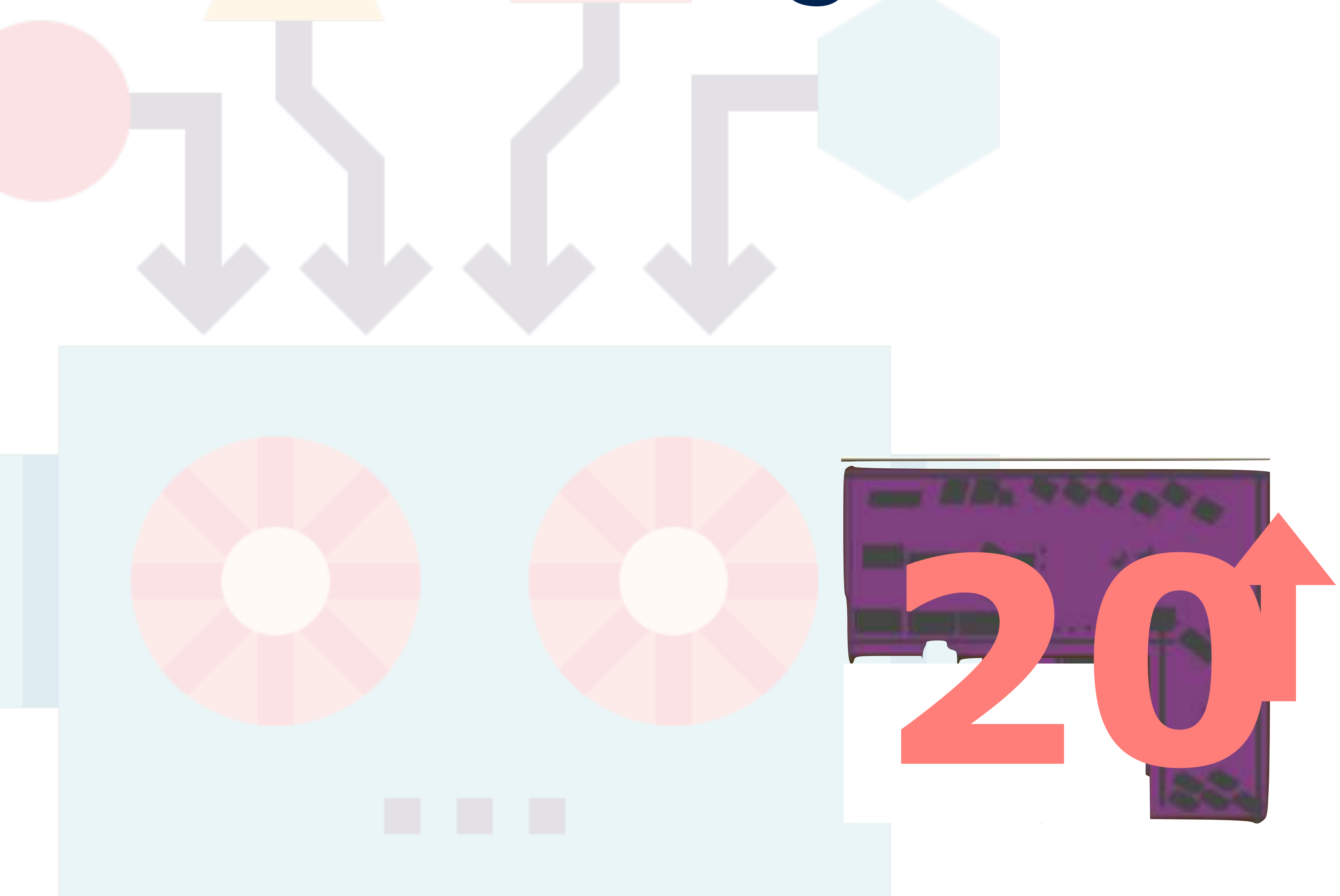
**Obstacle avoidance**



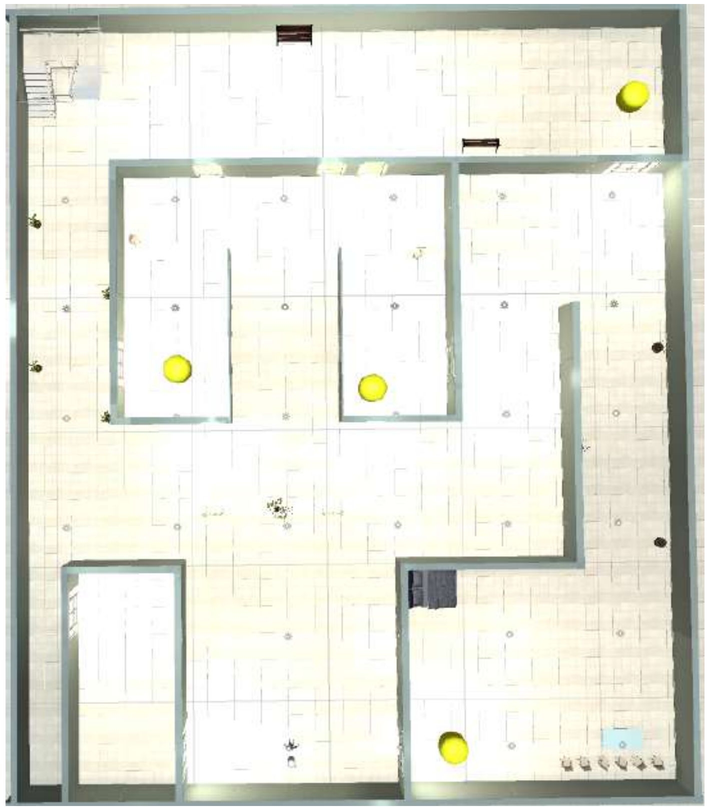

**Target following**


[1] Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World, Josh, et al.

# Training Environments



# EVALUATION ENVIRONMENTS



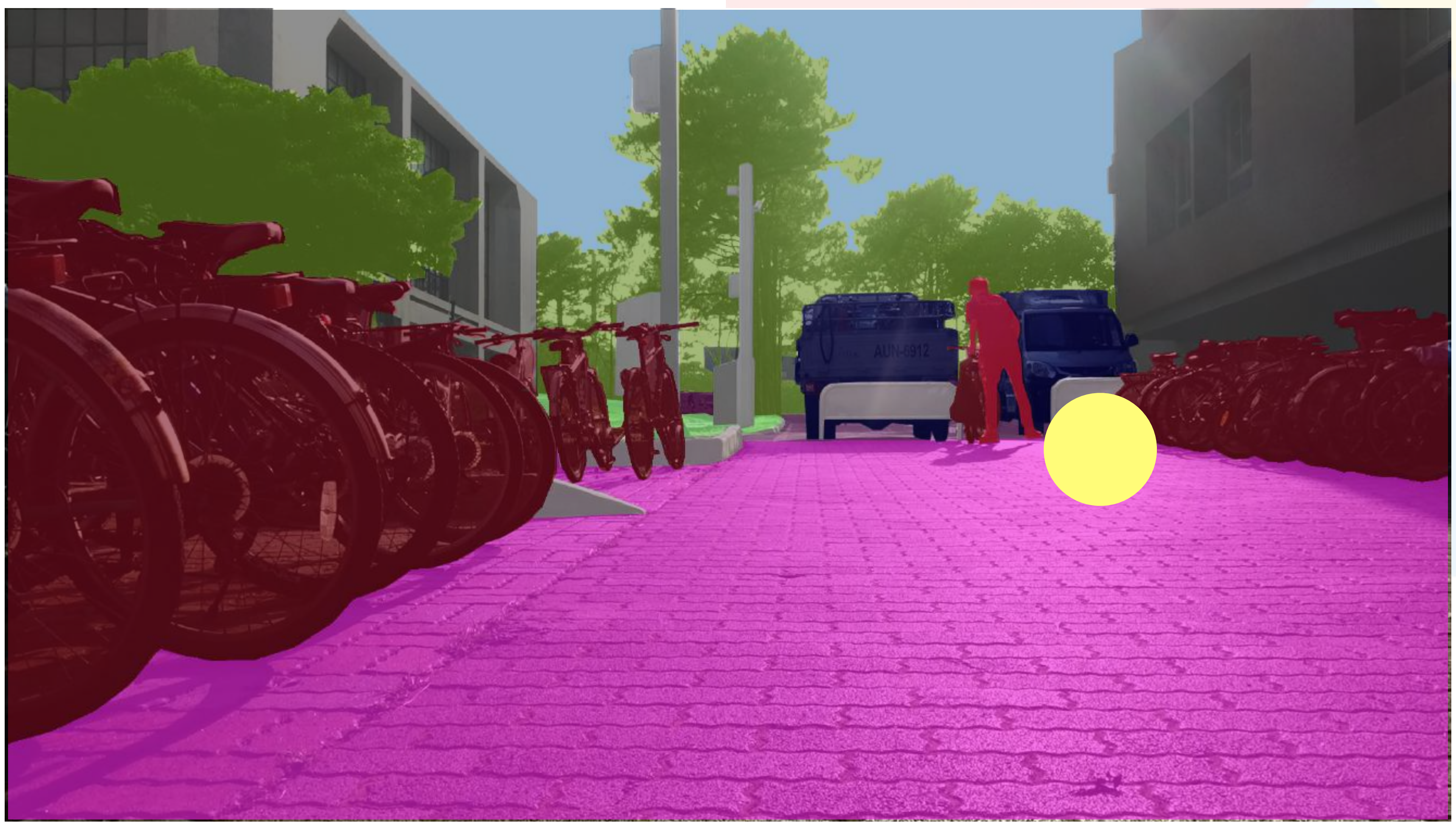
# EVALUATION ENVIRONMENTS

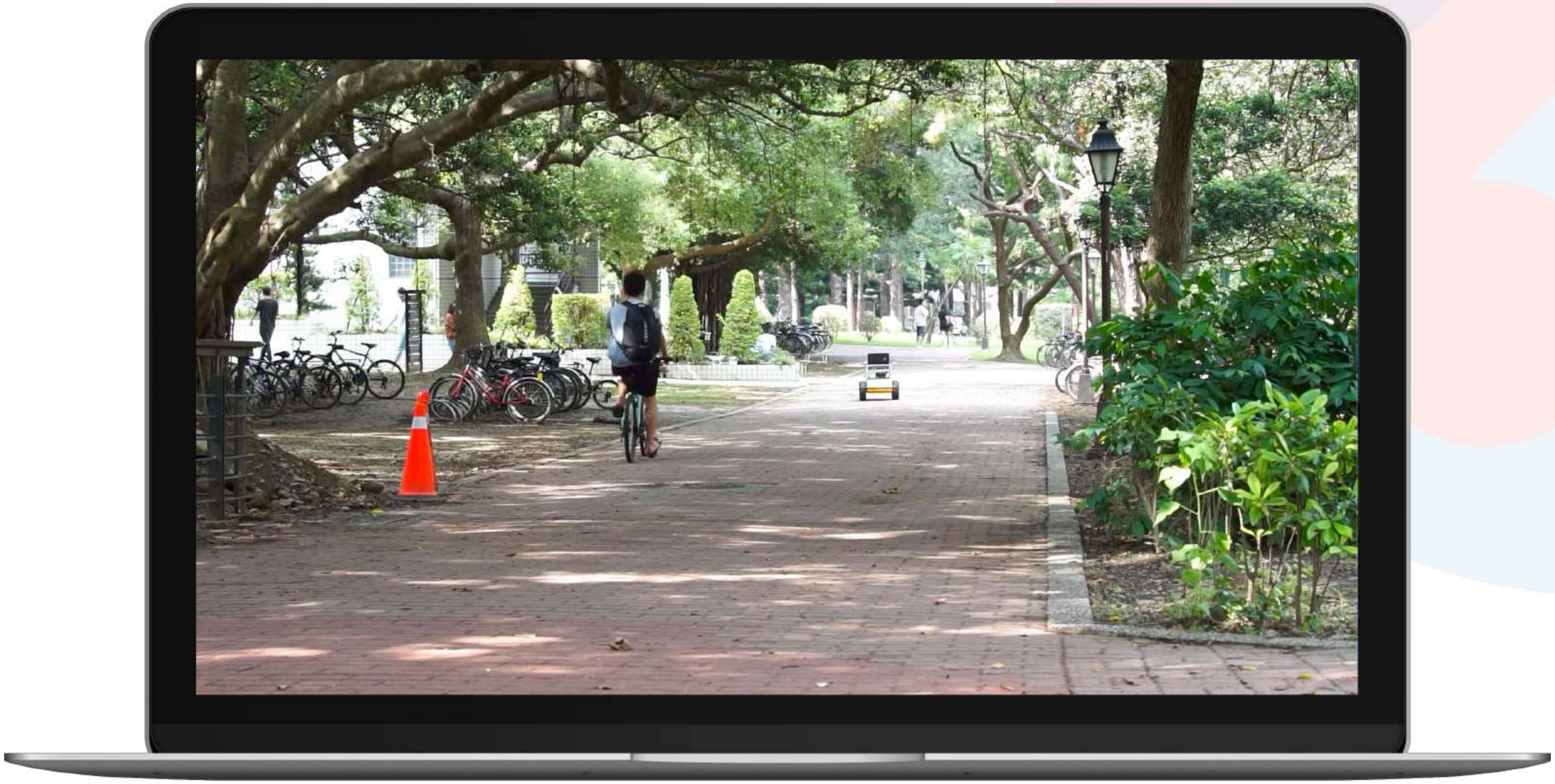
# Demonstration

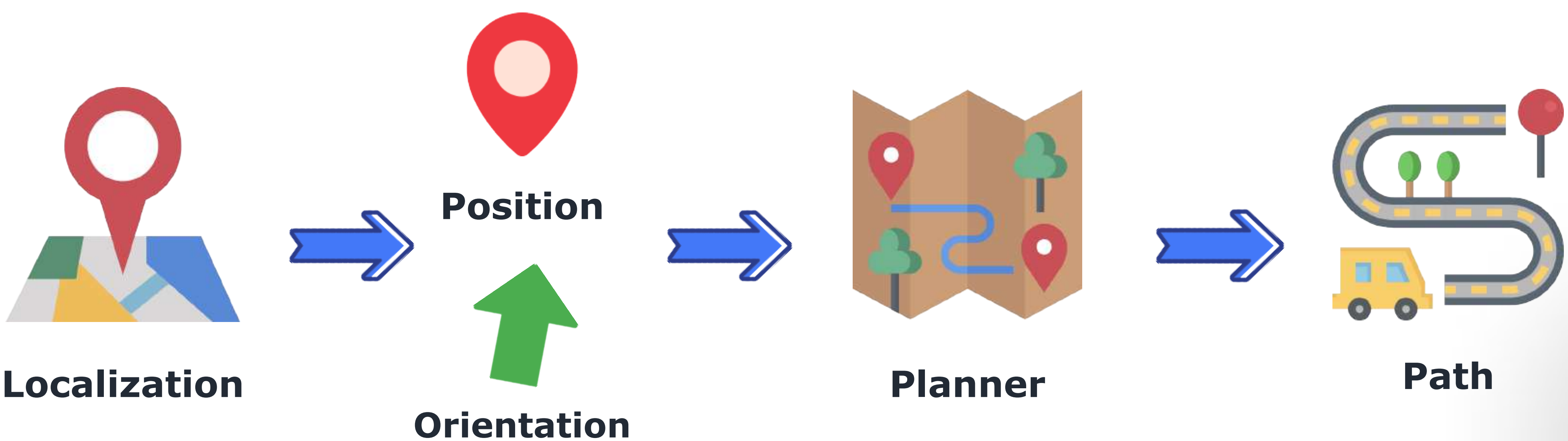
**Link: https://youtu.be/jz4lipO54Jg**

# Further Improvement of Virtual-to-Real — Virtual Guide

- Denoted as a 2-D Ball
- Dynamically Adjust the Size and Position






Localization
Position
Orientation
Planner
Path

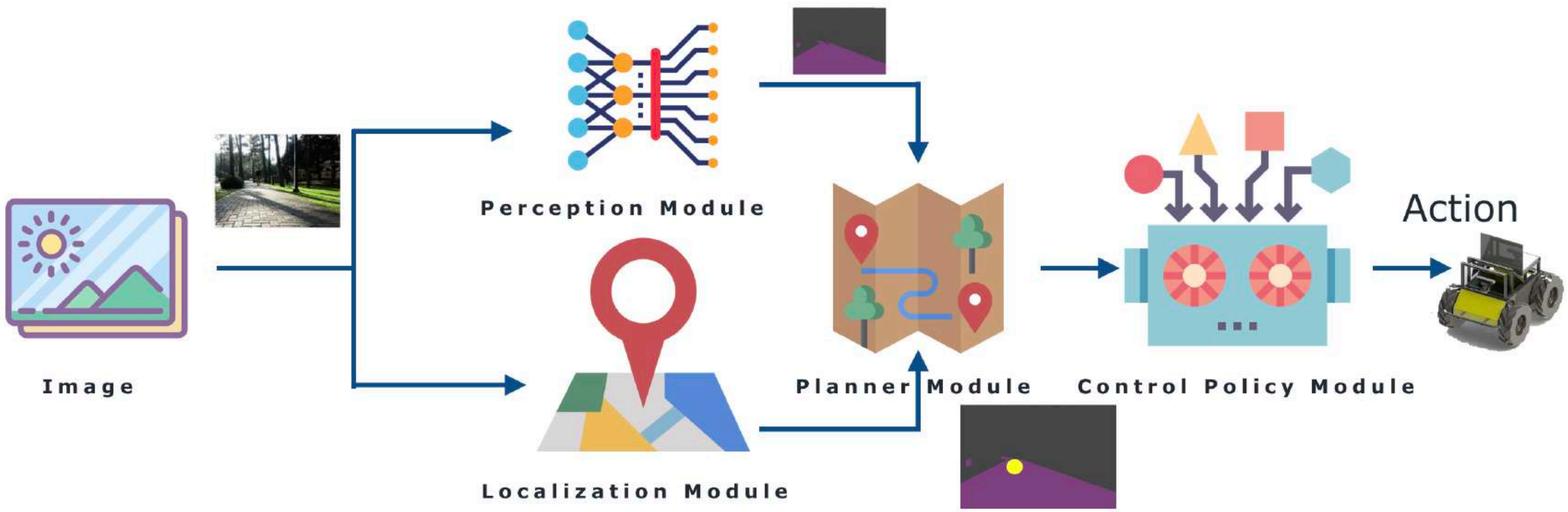
# Virtual Guidance for Robot Navigation

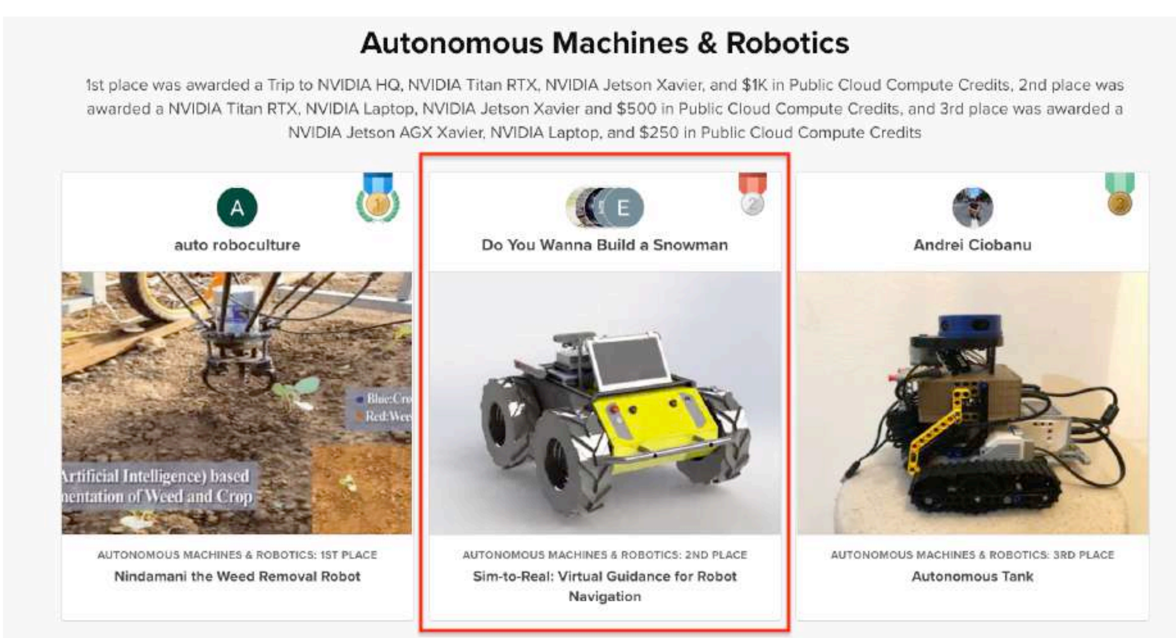
**NVIDIA AI the Edge Second Place Award, GTC 2020 Poster**

**Video Demonstration:** https://youtu.be/1Pq9YidaeBI

**Official Project Description:** https://www.hackster.io/contests/NVIDIA

(*including the report, source codes, as well as all the technical details)




# Demonstration

**Link: https://youtu.be/G9tcofUwFPw**

# Agenda

- Reinforcement Learning Backgrounds
- DRL Techniques
- Exploration
- Robotic Applications
- Summary

MLSS 2021 TAIPEI

ELSA LAB

# Summary

- In this talk, we discussed the fundamental concepts of reinforcement learning, and introduced the concepts of deep reinforcement learning
- We have explained the importance of exploration, and discussed several representative exploration techniques.
- We have discussed how reinforcement learning can be applied to robotic applications, and demonstrated how a policy can be trained in virtual worlds and transferred to the real world.

# Thank you for your attention!
# Q & A

## Fundamentals and Applications of Deep Reinforcement Learning

8/4/2021

**Prof. Chun-Yi Lee**

cylee@cs.nthu.edu.tw

ELSA LAB

http://elsalab.ai

**Elsa Lab, Department of Computer Science**

**National Tsing Hua University**