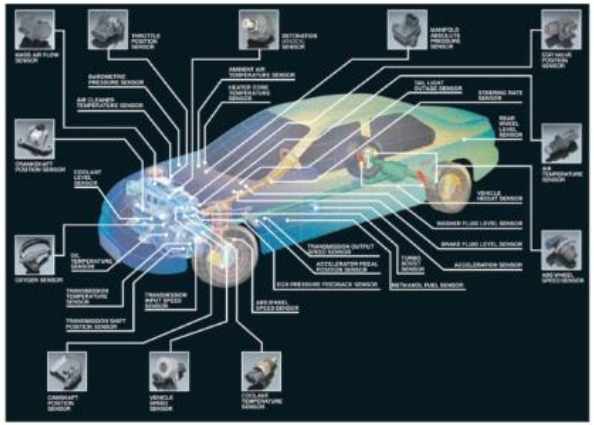
Randomized Adversarial Imitation Learning (IJCAI'19)

KICS AI Conference

Prof. Joongheon Kim
Korea University, School of Electrical Engineering
Artificial Intelligence and Mobility Laboratory
https://joongheon.github.io
joongheon@korea.ac.kr

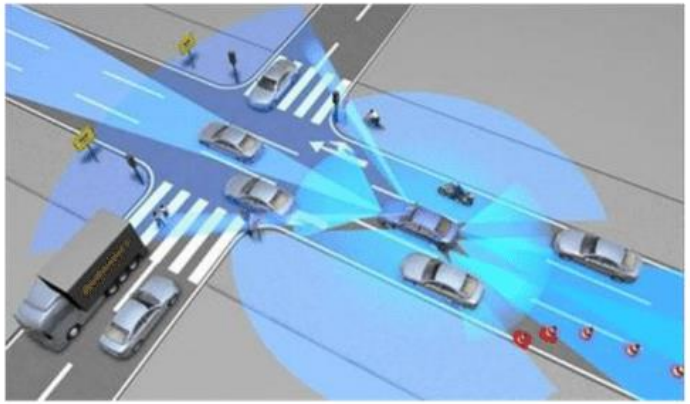
- **Vehicle Sensors**
- **Planning Algorithms**
- **Control Algorithms**
- **Deep Learning**

Vehicle Sensors



- Various sensors for vehicle (e.g. LIDAR, RADAR, ···)
- High performance
- Sensor fusion techniques

ADAS Algorithms

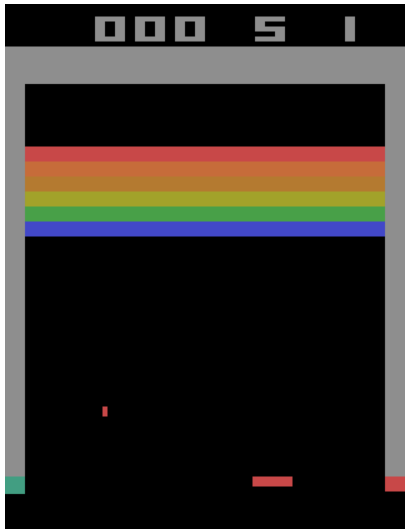


- Various ADAS (e.g. AEB, LKAS, BSD, ESC, ···)
- Already commercialized
- Essential function for safety

(ref) Deep Q Learning Based High Level Driving Policy Determination by Kyusik Min (IV Symposium 2018)

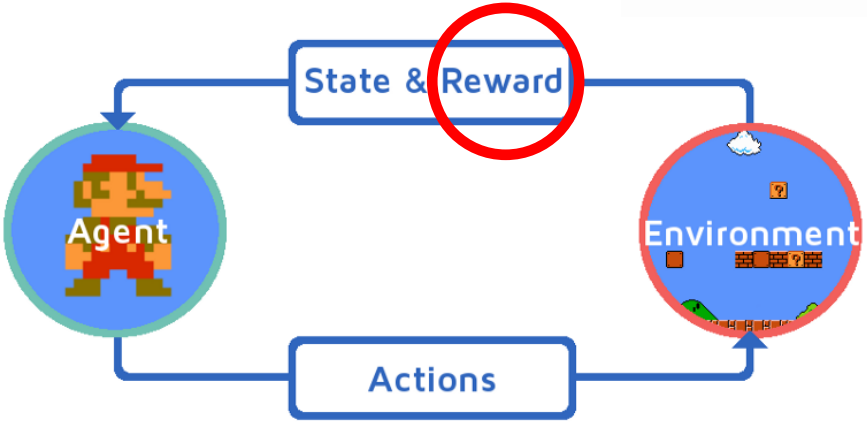
- Motivation
  - Many driver assistance systems are already commercialized and applied to lots of vehicles.
  - Many driver assistance systems perform partial function of self-driving car (i.e., lane keeping, cruising, lane changing, …)
  - Autonomous driving can be achieved by properly choosing driver assistant system in simple highway scenarios.
    - The problem is how to design a supervisor for selecting a proper action among driver assistance systems.

Goal : Learn policies
High-dimensional & raw observations



Super Human Performance

Challenge : Provide appropriate cost or reward signal.



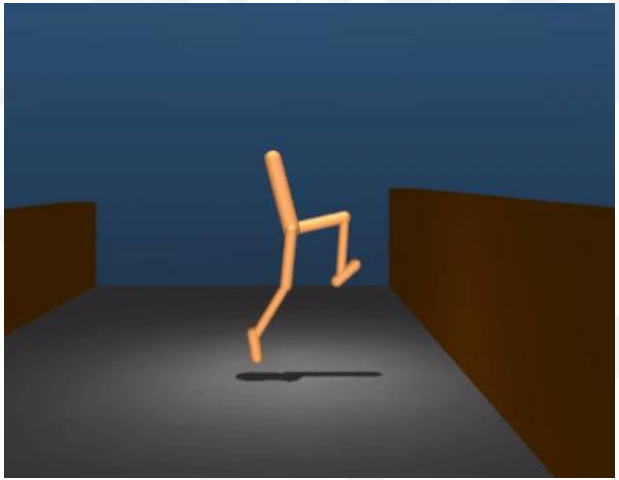
- Sparse reward
- Mathematical definition

Jump is desired behavior or not?

Input : expert behavior generated by expert π_E

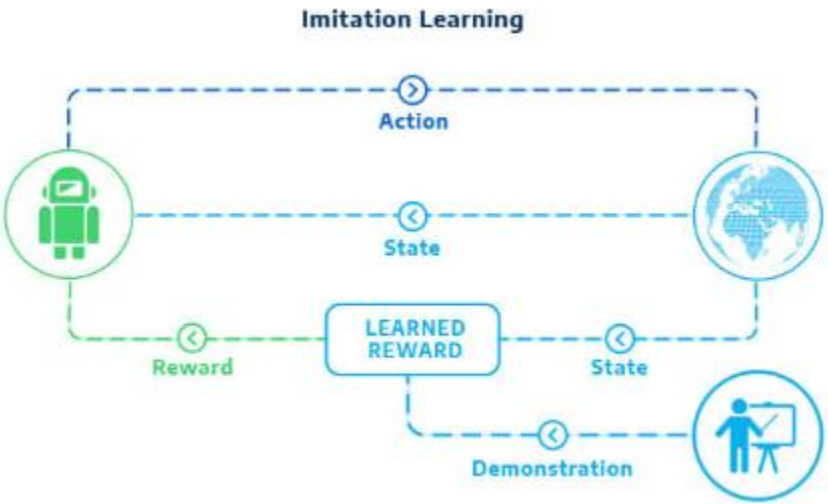
$$\{(s_0^i, a_0^i, s_1^i, a_1^i, \dots)\}_{i=1}^{N} \sim \pi_E$$

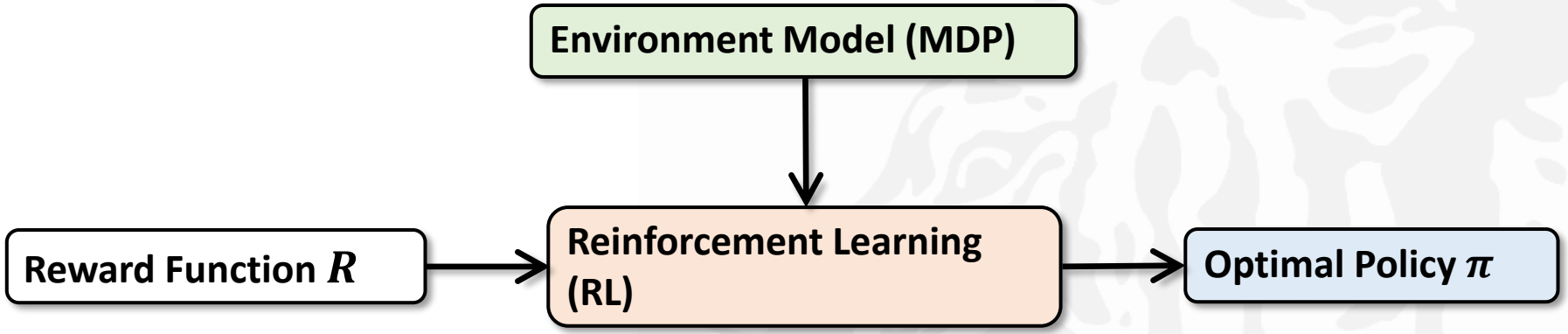
Goal : learn **cost function** or **policy**



Goal

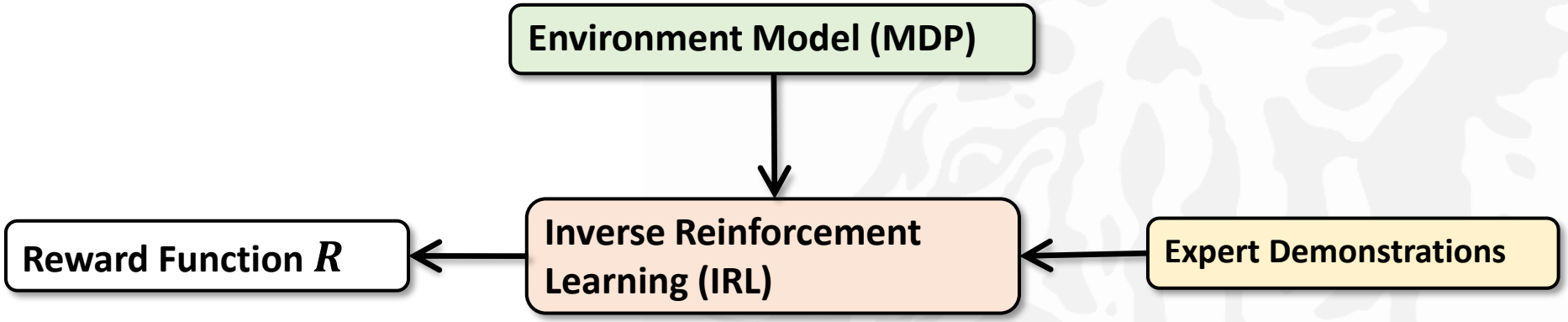
- Apply imitation learning to train supervisor of self driving car
- Enhancing safety of the self driving agent during training and testing.





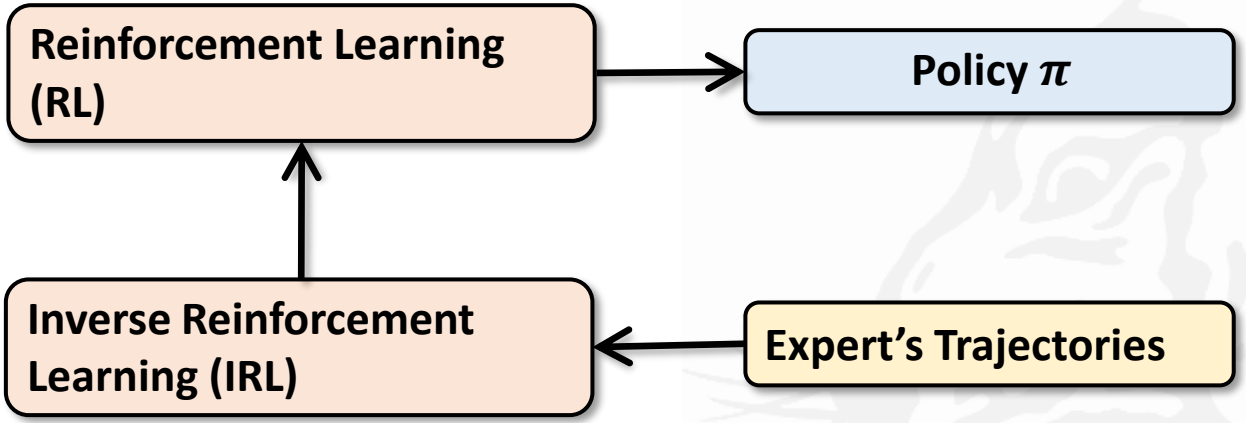
Takes as input a cost function

$$RL(R) = arg\min_{\pi} \mathbb{E}_{\pi}\left[R(s,a)\right] - H(\pi)$$



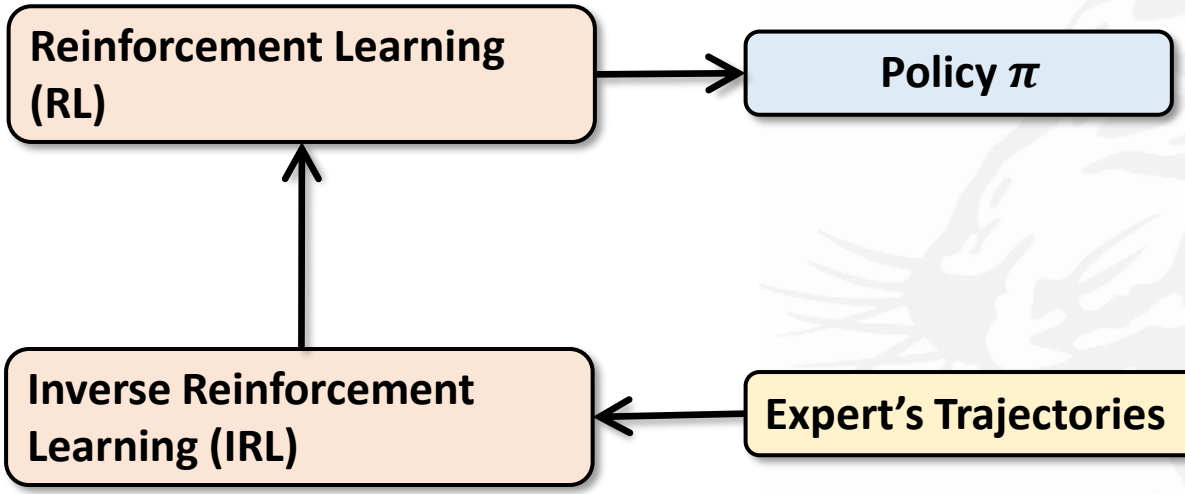
Make the expert policy uniquely optimal with respect to that cost function.

$$\max_R \left(\min_\pi \mathbb{E}_\pi [R(s,a)] - H(\pi) \right) - \mathbb{E}_{\pi_E}[R(s,a)]$$



$$\max_{R} -\psi(R) + \left(\min_{\pi} \mathbb{E}_{\pi}\left[R(s,a)\right] - H(\pi)\right) - \mathbb{E}_{\pi_E}\left[R(s,a)\right]$$

Maximum Entropy Inverse Reinforcement Learning, *AAAI 2008*



ρ_π = occupancy measure

➔ Distribution of state-action pairs encountered when navigating the environment with the policy

ρ_{E_π} = expert's occupancy measure

[Theorem]

ψ regularized inverse reinforcement learning implicitly, seeks a policy whose occupancy measure is close to the expert's, as measured by ψ^*

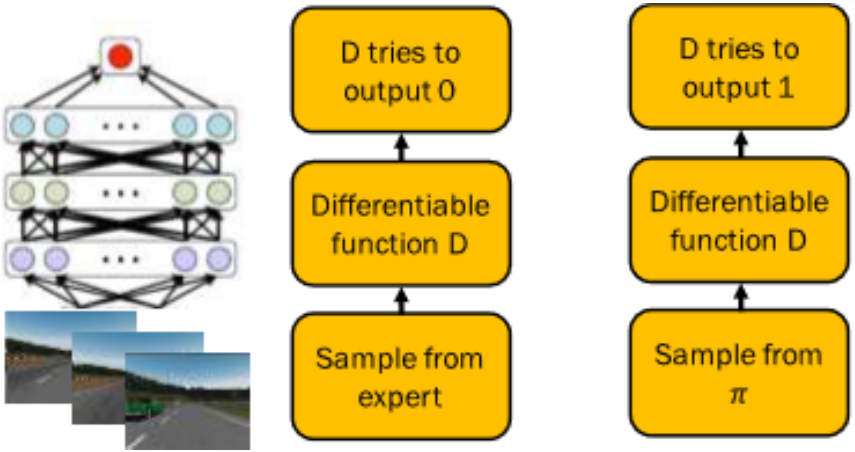
- Typical IRL finds a cost function such that the expert policy is uniquely optimal
- IRL as a procedure that tries to induce a policy that matches the expert's occupancy measure (generative model)

Generative Adversarial Imitation Learning (GAIL), *NIPS 2016*

Use this regularizer $\psi_{GA}(R) = \begin{cases} \mathbb{E}_{\pi_E}[g(R(s,a))] & if\ R < 0 \\ +\infty & otherwise \end{cases}$

$$\min_{\pi} \psi^*(\boldsymbol{\rho_\pi} - \boldsymbol{\rho_{E_\pi}}) - H(\pi)$$

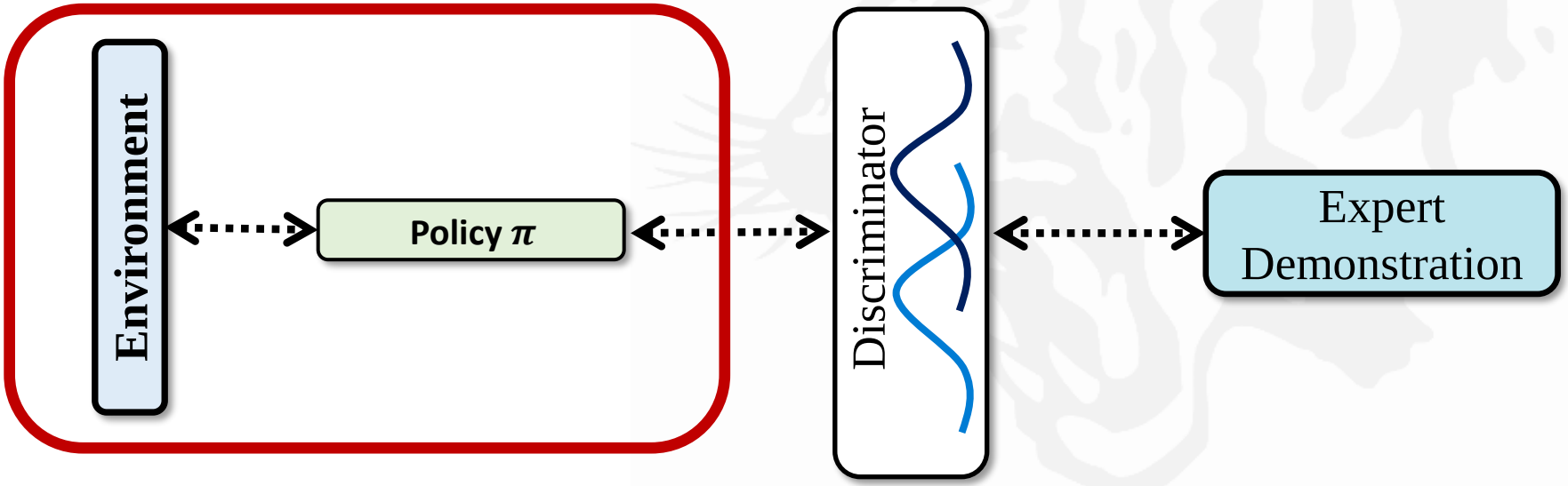
Generative Adversarial Imitation Learning (GAIL), *NIPS 2016*



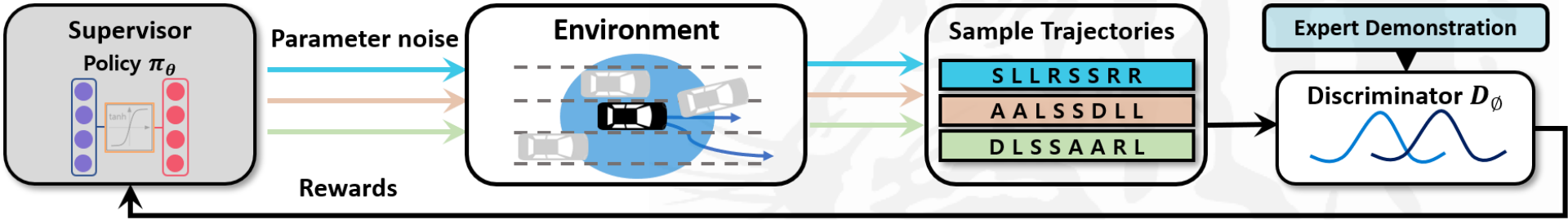
- The quality of the samples we obtain is measured by training the discriminator D.
- The policy is trained to produce behaviors that are difficult to distinguish from expert.

Challenge

- A lot of interaction with the environment is required to optimize the policy through GAIL framework
- Complex structure RL algorithm to train the policy.



$$minimize\ \ \mathbb{E}_{\pi}[\log(D(s,a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a)]$$



$D(s,a)$: Probability between 0 and 1

The probability that the input data sample is the expert data sample

Usually in AI:

$$f'(x) = \frac{df}{dx}$$

- To update the weights of policy the gradient descent method is used.

Proposed method

$$f'(a) = \frac{f(a+h) - f(a)}{h}$$

- To update the weights of policy **the method of finite differences.**

Policy

θ_t

$w_{1,1}$	$w_{1,2}$	$w_{1,3}$
$w_{2,1}$	$w_{2,2}$	$w_{2,3}$

Gaussian Noise

$\boldsymbol{\delta} = \{\delta_1, \delta_2, \cdots, \delta_N\}$

δ_i

0.14	0.37	0.49
0.96	0.25	0.94

$w_{1,1}$ + 0.14	$w_{1,2}$ + 0.37	$w_{1,3}$ + 0.49
$w_{2,1}$ + 0.96	$w_{2,2}$ + 0.25	$w_{2,3}$ + 0.94

$w_{1,1}$ - 0.14	$w_{1,2}$ - 0.37	$w_{1,3}$ - 0.49
$w_{2,1}$ - 0.96	$w_{2,2}$ - 0.25	$w_{2,3}$ - 0.94

random numbers or a matrix with random tiny values

Positive perturbative weights

R_{d-pos}

$w_{1,1} + d_{1,1}$	$w_{1,2} + d_{1,2}$	$w_{1,3} + d_{1,3}$
$w_{2,1} + d_{2,1}$	$w_{2,2} + d_{2,2}$	$w_{2,3} + d_{2,3}$

R_{e-pos}

$w_{1,1} + e_{1,1}$	$w_{1,2} + e_{1,2}$	$w_{1,3} + e_{1,3}$
$w_{2,1} + e_{2,1}$	$w_{2,2} + e_{2,2}$	$w_{2,3} + e_{2,3}$

R_{f-pos}

$w_{1,1} + f_{1,1}$	$w_{1,2} + f_{1,2}$	$w_{1,3} + f_{1,3}$
$w_{2,1} + f_{2,1}$	$w_{2,2} + f_{2,2}$	$w_{2,3} + f_{2,3}$

R_{g-pos}

$w_{1,1} + g_{1,1}$	$w_{1,2} + g_{1,2}$	$w_{1,3} + g_{1,3}$
$w_{2,1} + g_{2,1}$	$w_{2,2} + g_{2,2}$	$w_{2,3} + g_{2,3}$

Negative perturbative weights

R_{d-neg}

$w_{1,1} - d_{1,1}$	$w_{1,2} - d_{1,2}$	$w_{1,3} - d_{1,3}$
$w_{2,1} - d_{2,1}$	$w_{2,2} - d_{2,2}$	$w_{2,3} - d_{2,3}$

R_{e-neg}

$w_{1,1} - e_{1,1}$	$w_{1,2} - e_{1,2}$	$w_{1,3} - e_{1,3}$
$w_{2,1} - e_{2,1}$	$w_{2,2} - e_{2,2}$	$w_{2,3} - e_{2,3}$

R_{f-neg}

$w_{1,1} - f_{1,1}$	$w_{1,2} - f_{1,2}$	$w_{1,3} - f_{1,3}$
$w_{2,1} - f_{2,1}$	$w_{2,2} - f_{2,2}$	$w_{2,3} - f_{2,3}$

R_{g-neg}

$w_{1,1} - g_{1,1}$	$w_{1,2} - g_{1,2}$	$w_{1,3} - g_{1,3}$
$w_{2,1} - g_{2,1}$	$w_{2,2} - g_{2,2}$	$w_{2,3} - g_{2,3}$

Randomized Adversarial Imitation Learning (RAIL)

R_{d-pos}

$w_{1,1} + d_{1,1}$	$w_{1,2} + d_{1,2}$	$w_{1,3} + d_{1,3}$
$w_{2,1} + d_{2,1}$	$w_{2,2} + d_{2,2}$	

R_{e-pos}

$w_{1,1} + e_{1,1}$	$w_{1,2} + e_{1,2}$	$w_{1,3} + e_{1,3}$
$w_{2,1} + e_{2,1}$	$w_{2,2} + e_{2,2}$	

R_{f-pos}

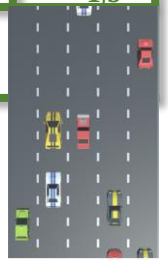
$w_{1,1} + f_{1,1}$	$w_{1,2} + f_{1,2}$	$w_{1,3} + f_{1,3}$
$w_{2,1} + f_{2,1}$	$w_{2,2} + f_{2,2}$	

R_{g-pos}

$w_{1,1} + g_{1,1}$	$w_{1,2} + g_{1,2}$	$w_{1,3} + g_{1,3}$
$w_{2,1} + g_{2,1}$	$w_{2,2} + g_{2,2}$	

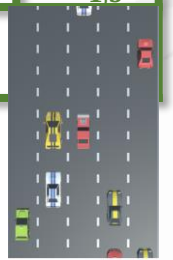
R_{d-neg}

$w_{1,1} - d_{1,1}$	$w_{1,2} - d_{1,2}$	$w_{1,3} - d_{1,3}$
$w_{2,1} - d_{2,1}$	$w_{2,2} - d_{2,2}$	



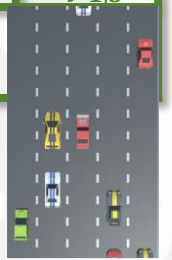
R_{e-neg}

$w_{1,1} - e_{1,1}$	$w_{1,2} - e_{1,2}$	$w_{1,3} - e_{1,3}$
$w_{2,1} - e_{2,1}$	$w_{2,2} - e_{2,2}$	



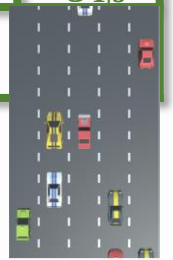
R_{f-neg}

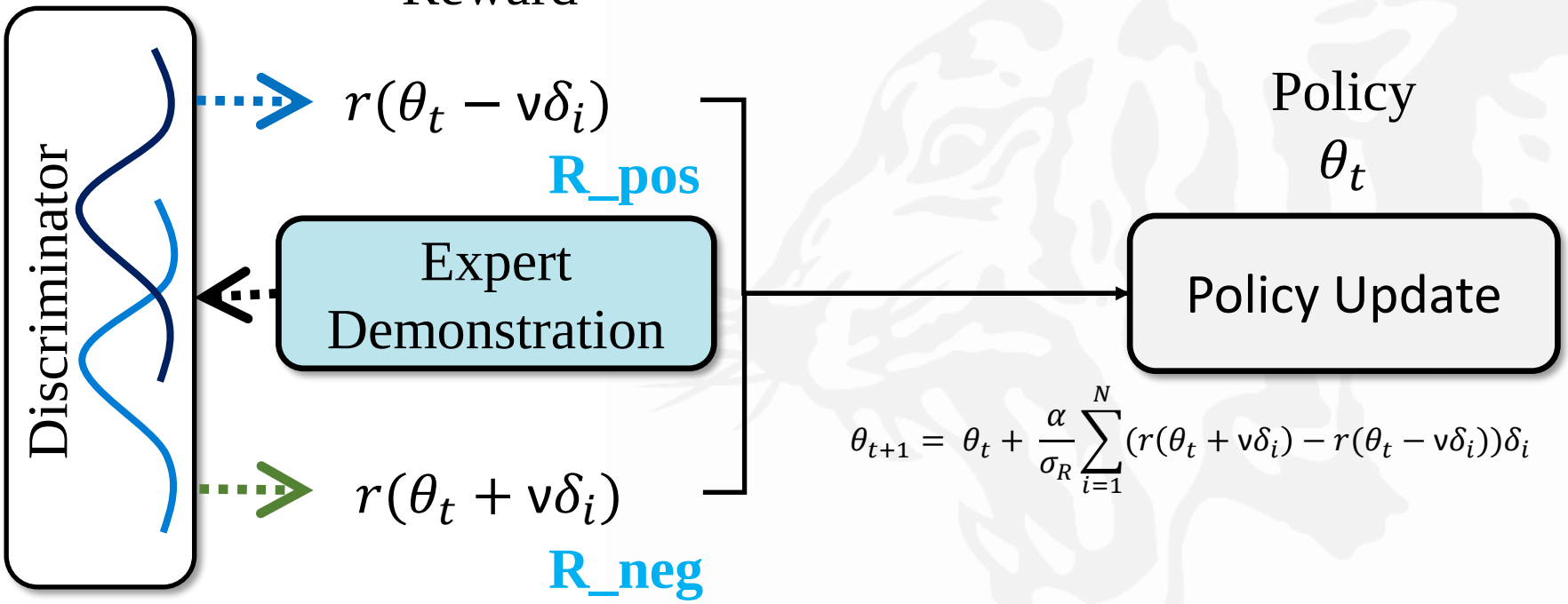
$w_{1,1} - f_{1,1}$	$w_{1,2} - f_{1,2}$	$w_{1,3} - f_{1,3}$
$w_{2,1} - f_{2,1}$	$w_{2,2} - f_{2,2}$	

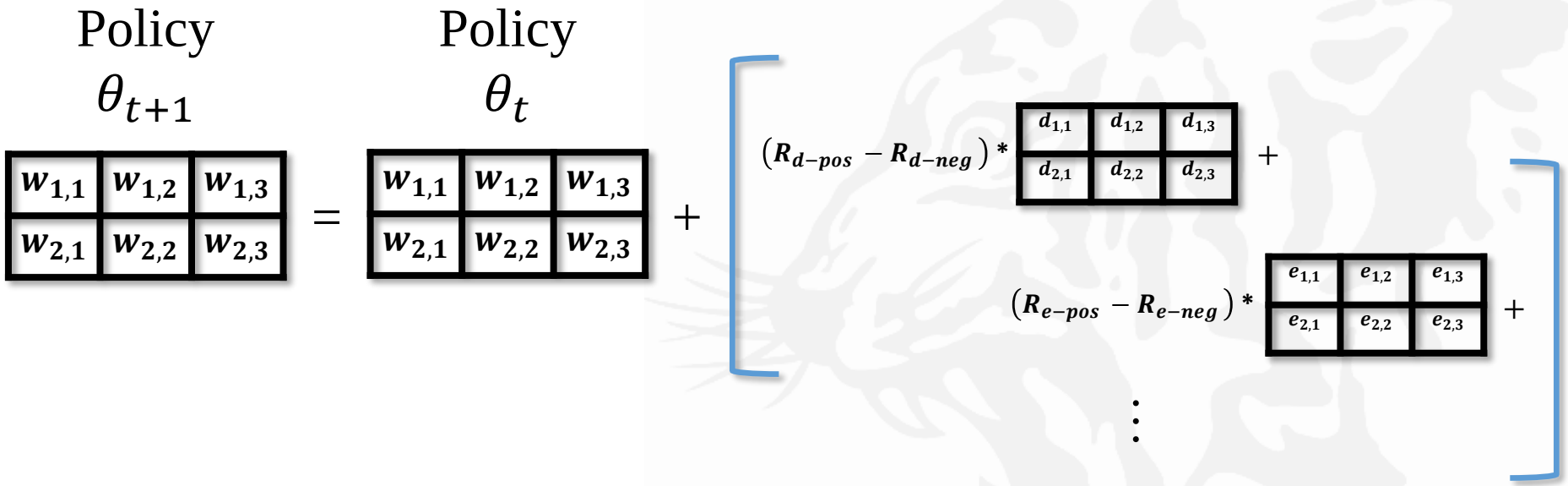


R_{g-neg}

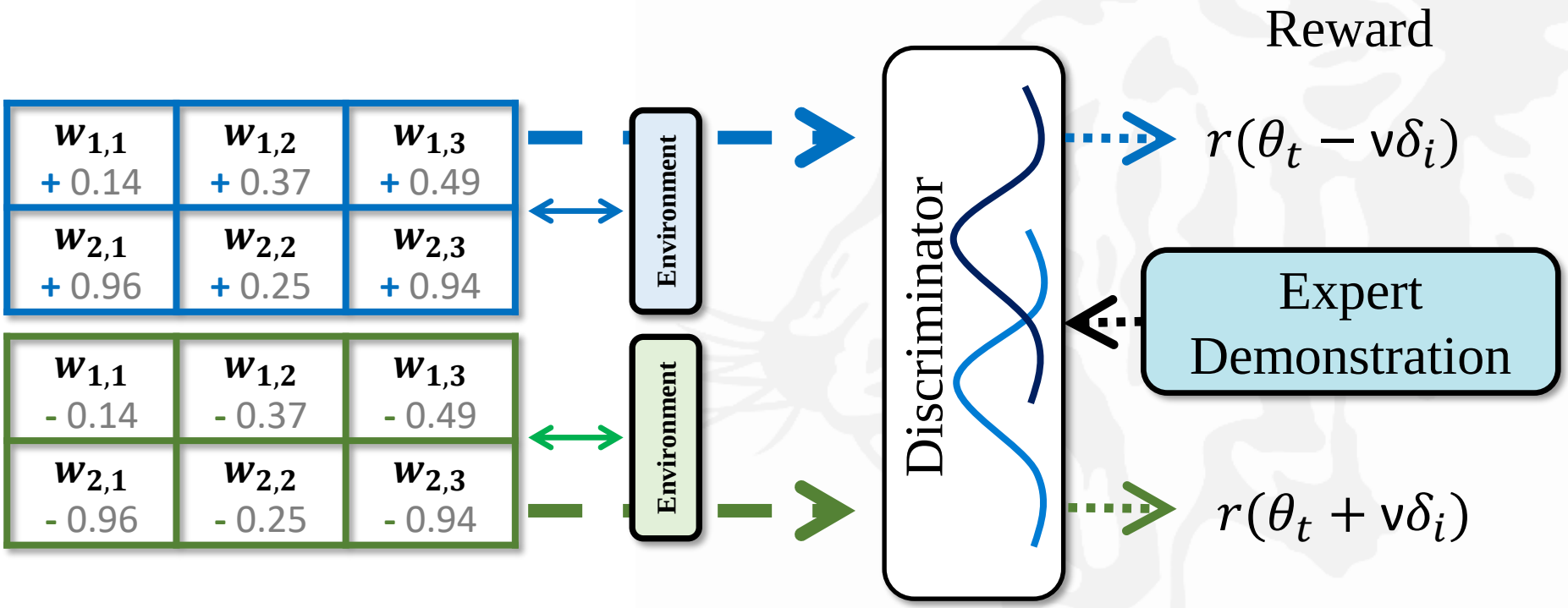
$w_{1,1} - g_{1,1}$	$w_{1,2} - g_{1,2}$	$w_{1,3} - g_{1,3}$
$w_{2,1} - g_{2,1}$	$w_{2,2} - g_{2,2}$	




Reward
Discriminator
$r(\theta_t - \nu\delta_i)$
R_pos
Expert Demonstration
$r(\theta_t + \nu\delta_i)$
R_neg
Policy
θ_t
Policy Update
$\theta_{t+1} = \theta_t + \frac{\alpha}{\sigma_R}\sum_{i=1}^{N}(r(\theta_t + \nu\delta_i) - r(\theta_t - \nu\delta_i))\delta_i$

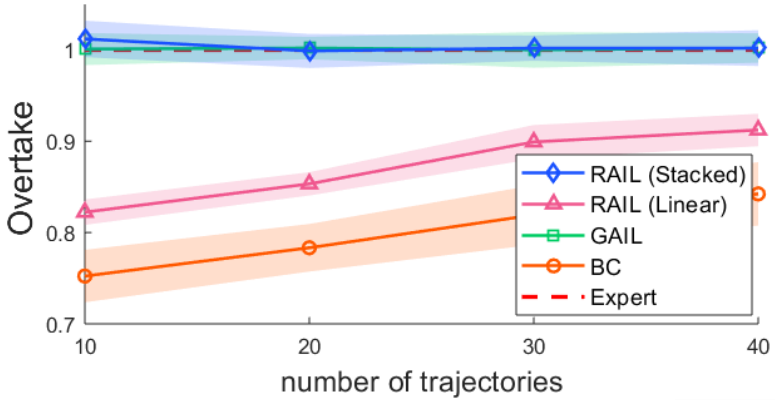

Policy
θ_{t+1}
Policy
θ_t
$w_{1,1}$ $w_{1,2}$ $w_{1,3}$
$w_{2,1}$ $w_{2,2}$ $w_{2,3}$
=
$w_{1,1}$ $w_{1,2}$ $w_{1,3}$
$w_{2,1}$ $w_{2,2}$ $w_{2,3}$
+
$(R_{d-pos} - R_{d-neg}) *$
$d_{1,1}$ $d_{1,2}$ $d_{1,3}$
$d_{2,1}$ $d_{2,2}$ $d_{2,3}$
+
$(R_{e-pos} - R_{e-neg}) *$
$e_{1,1}$ $e_{1,2}$ $e_{1,3}$
$e_{2,1}$ $e_{2,2}$ $e_{2,3}$
+
⋮

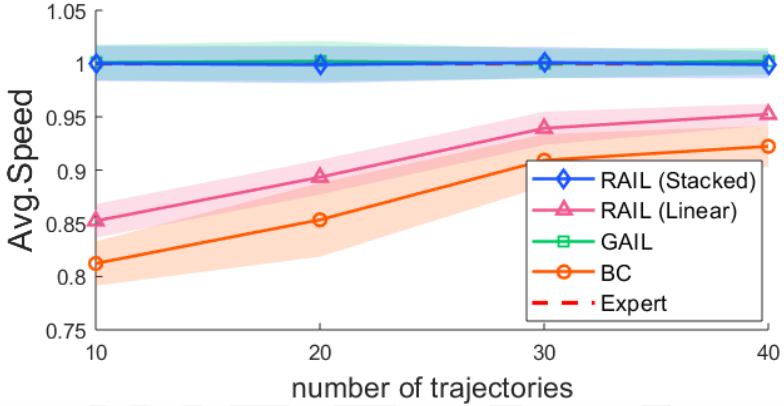
$$\text{Policy } \theta_{t+1} = \text{Policy } \theta_t + \left[(R_{d-pos} * \begin{bmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \end{bmatrix} - R_{d-neg} * \begin{bmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \end{bmatrix}) + \cdots \right]$$



Sample Trajectories

Randomized Adversarial Imitation Learning (RAIL)





Thank you for your attention!

- More questions?
  - joongheon@korea.ac.kr
- More details?
  - https://joongheon.github.io/