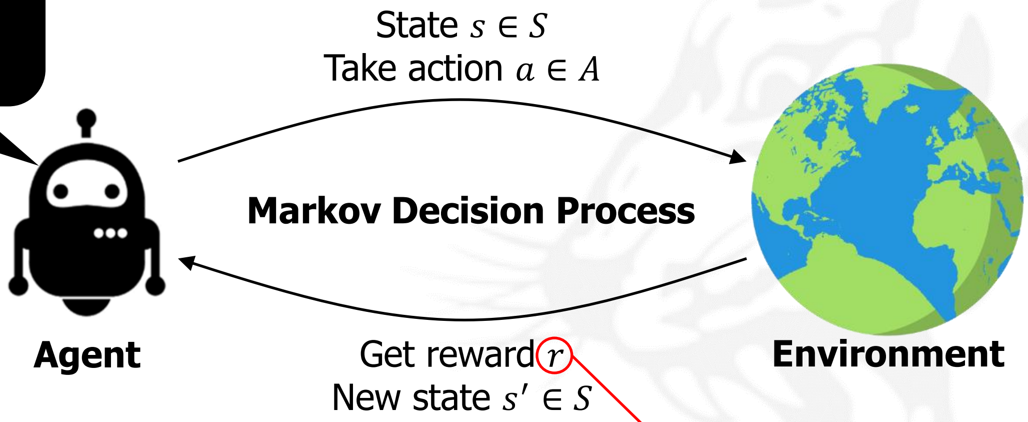
Artificial Intelligence and Mobility Lab

Meta Reinforcement Learning and Imitation Learning

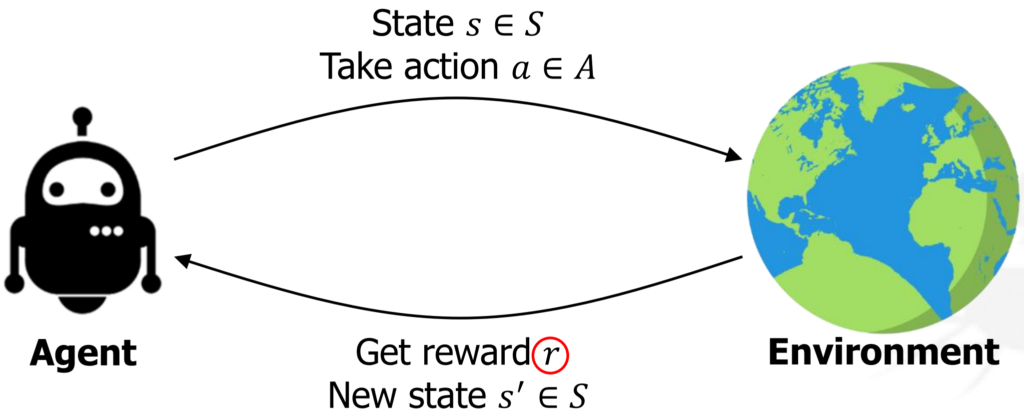
Prof. Joongheon Kim
Korea University, School of Electrical Engineering
Artificial Intelligence and Mobility Laboratory
https://joongheon.github.io
joongheon@korea.ac.kr

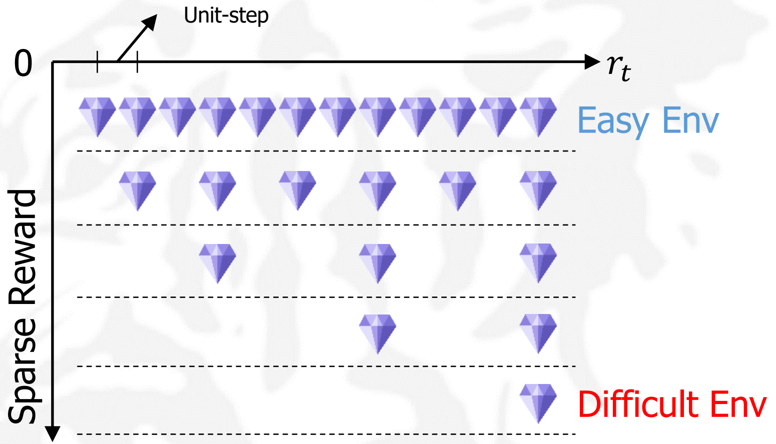
I decide actions in the direction of maximizing the cumulative reward.



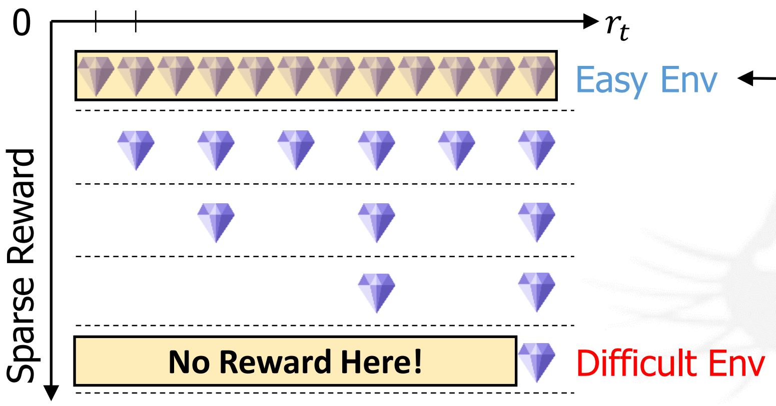
Reward Shaping is the only way to achieve goal

Reinforcement Learning





$$Q^*(s_t, a_t) \leftarrow Q(s_t, a_t) + a(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$



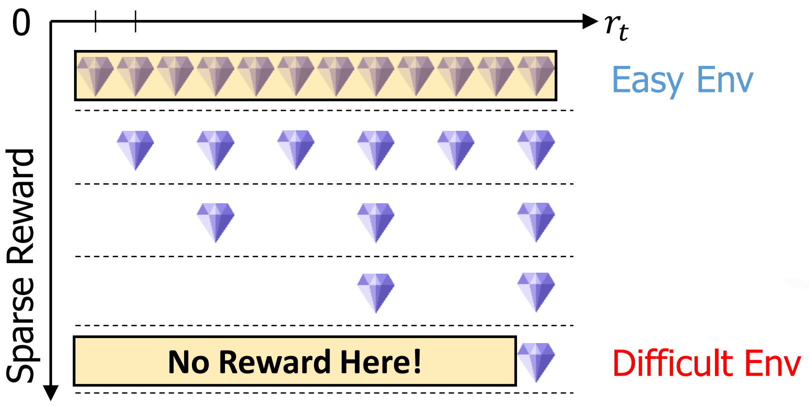
Agent recognizes "Every action is meaningful"

Easy to train

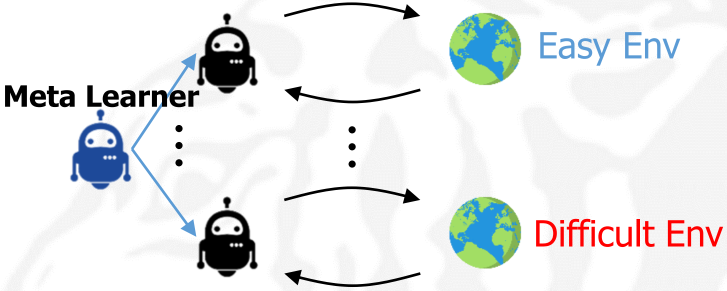
Agent recognizes "Every action is meaningless", because the frequency of reward is very low."

Hard to train

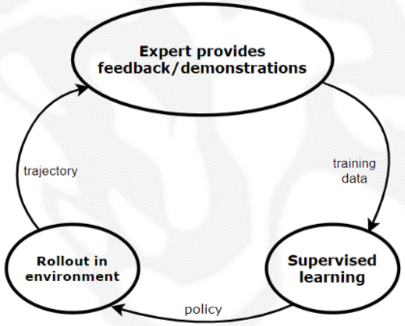
Two Approach to Solve



Meta Reinforcement Learning



Imitation Learning



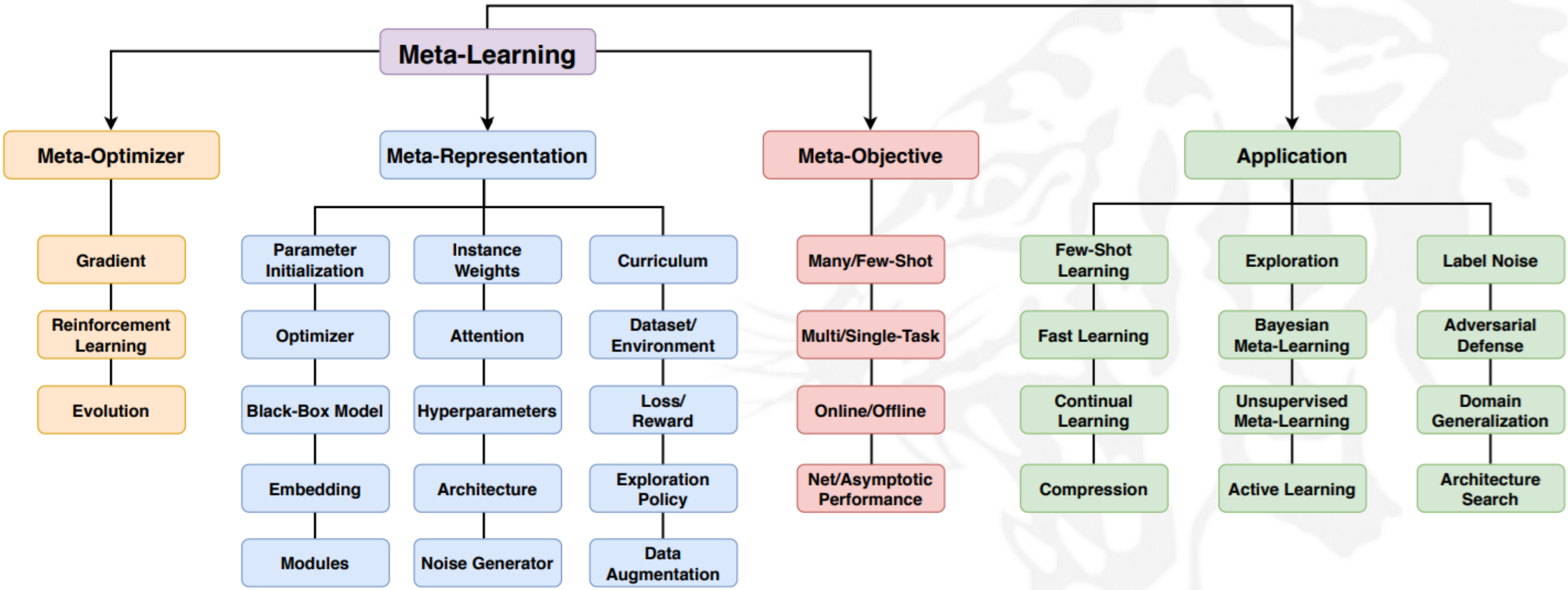
Contents

Meta Reinforcement Learning

Imitation Learning

Meta Reinforcement Learning

Meta Reinforcement Learning

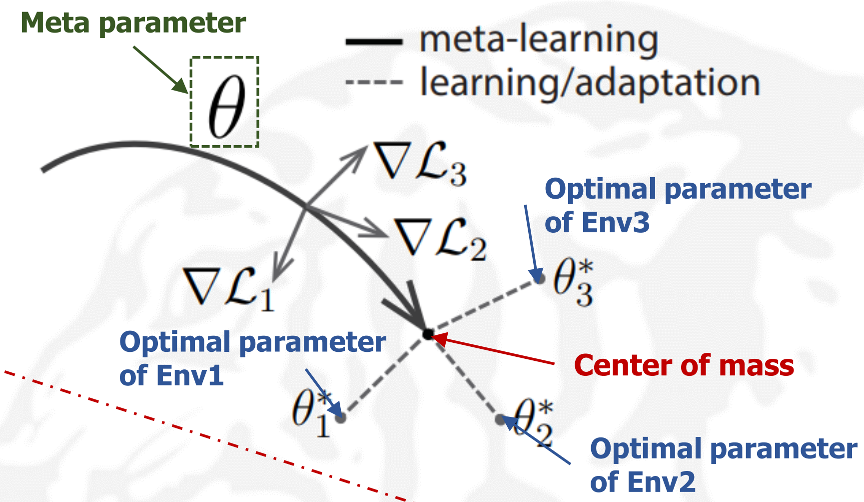


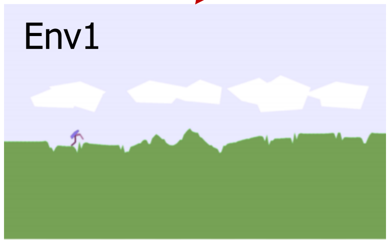
[1] Timothy Hospedales, Antreas Antoniou, Paul Micaelli and Amos Storkey, "Meta-Learning in Neural Networks: A Survey," *arxiv*, 2020

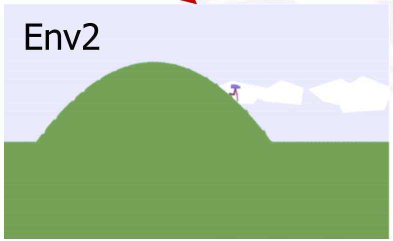
Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

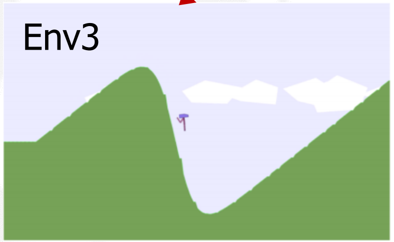
Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $\boxed{\mathcal{T}_i} \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
end while

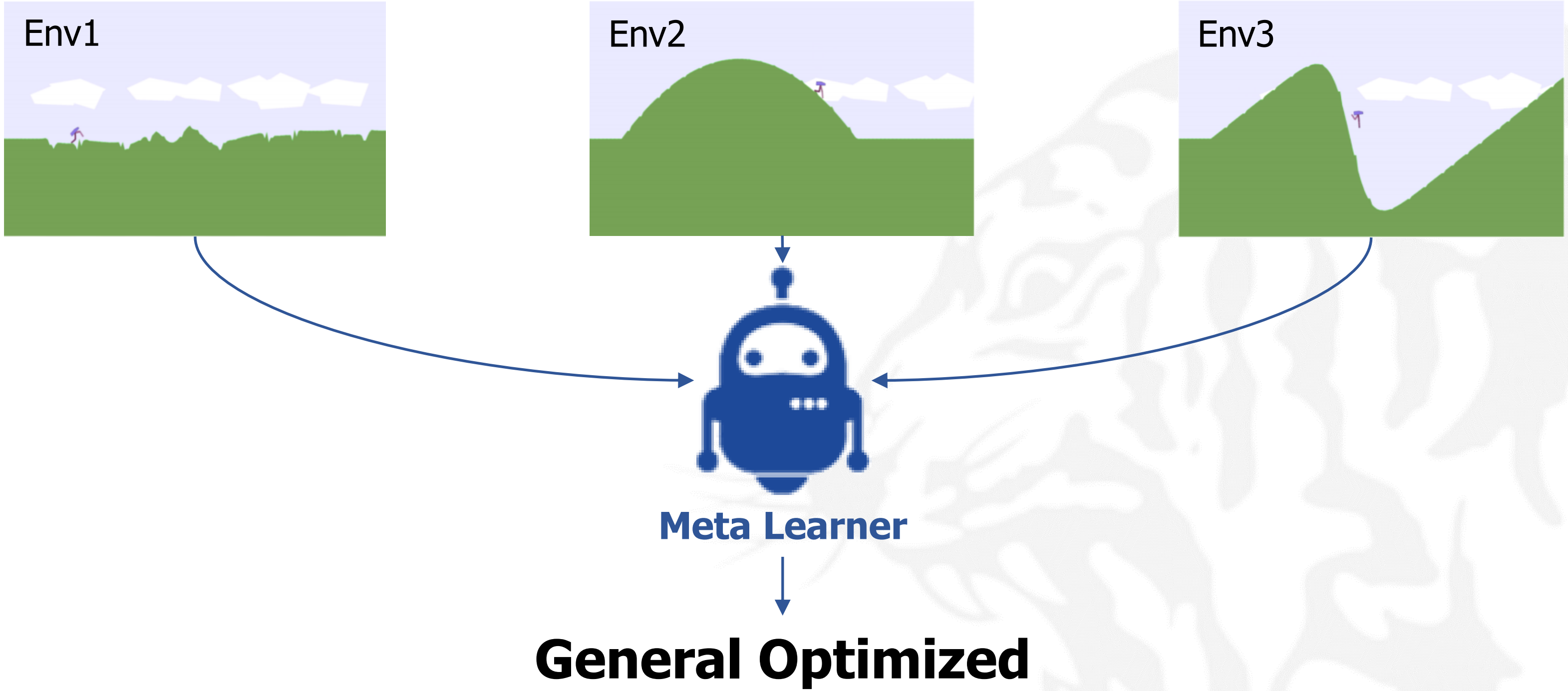


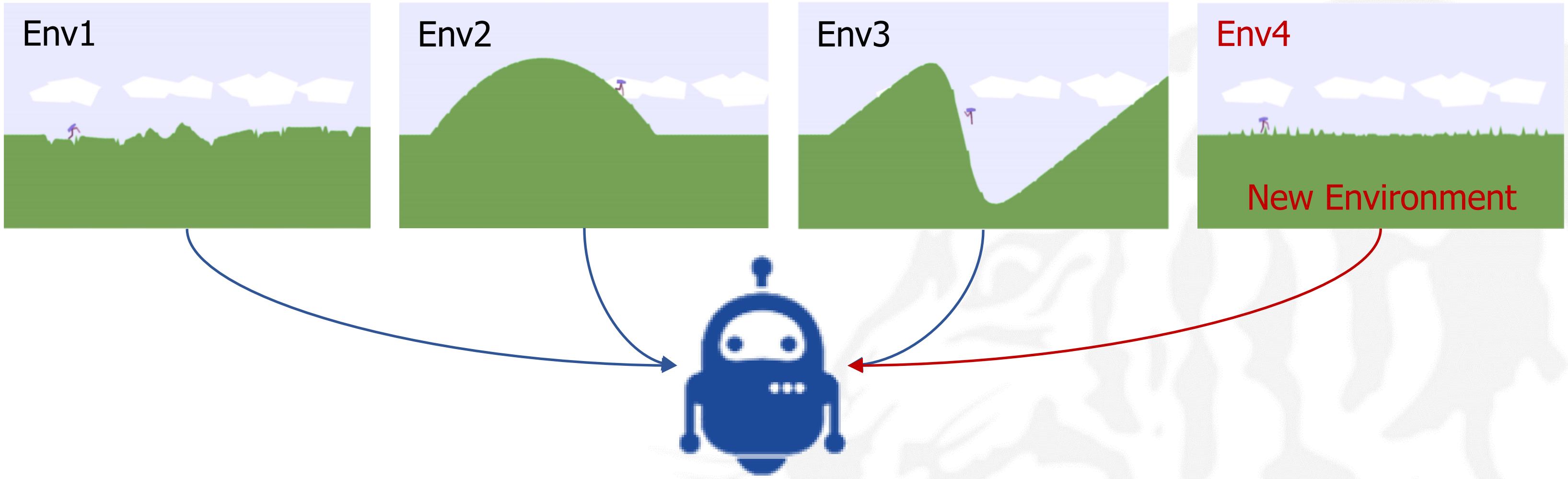






[2] Chelsea Finn, Pieter Abbeel and Sergey Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," *ICML*, 2017
[3] Rui Wang et al., "Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions ," *arxiv,* 2020


Env1
Env2
Env3
Meta Learner
General Optimized



Meta Learner

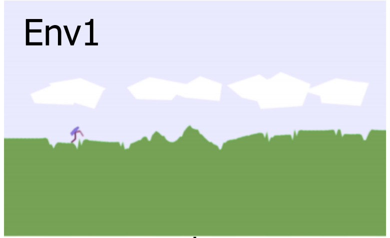
↓

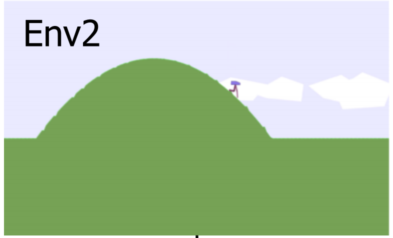
Fast Adaptation
&
Well Trained

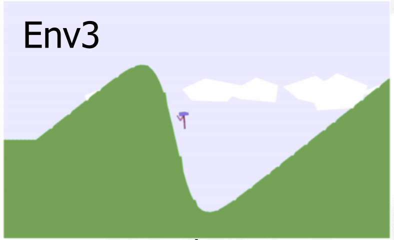
⇨ **Domain Adaptation**
Domain Generalization

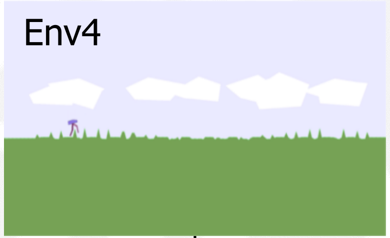
[4] Code Available: https://github.com/uber-research/poet, https://github.com/cbfinn/maml_rl

To optimize for all environments with conventional DRL method,









Reward Shaping for Env1

Reward Shaping for Env2

Reward Shaping for Env3

Reward Shaping for Env4

Conventional DRL Method is
"Task Specific Deep-Reinforcement Learning Model"

Algorithm 3 MAML for Reinforcement Learning

$\mathcal{T}_i$

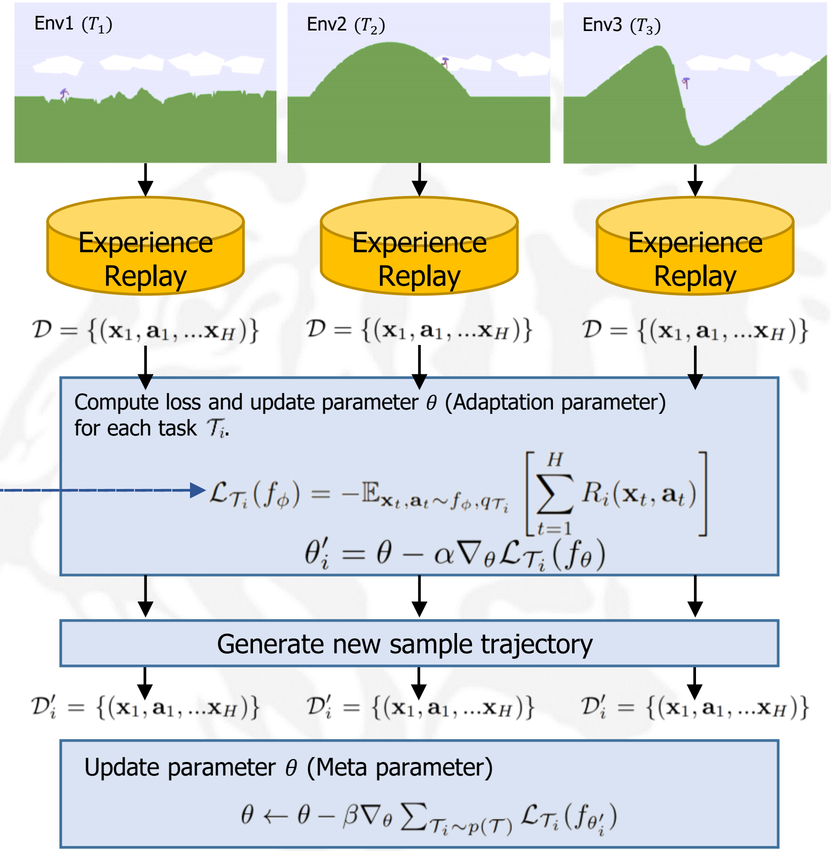
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

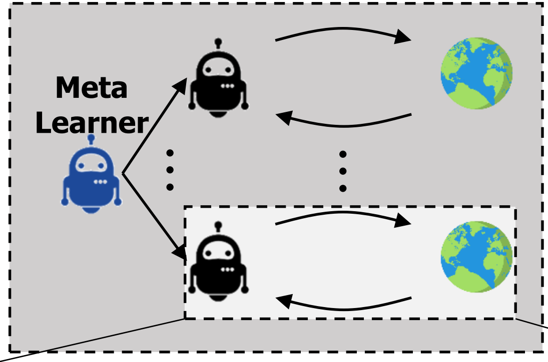
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**

Fine Tuning

    Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, \ldots \mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, \ldots \mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
end while

Update Meta-policy





Model-free Reinforcement Learning

Markov Decision Process: $\ll S, A, R, P, \gamma \gg$

With trajectory $D = \{\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{r}_1, \dots, \boldsymbol{s}_{T+1}\}$

min: $\mathcal{L}(\theta, D) = -\mathbb{E}_{(s_t, a_t)\sim\pi_\theta}\left[\sum_{t=1}^{H} R(s_t, a_t)\right] = -\mathbb{E}_{(s_t, a_t)\sim\pi_\theta}\left[\sum_{t=1}^{H} r_t\right]$

With Gradient Descent Method

$$\nabla_\theta \mathcal{L}(\theta, D) = -\mathbb{E}_{(s_t, a_t)\sim\pi_\theta}\left[A^\pi(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

$$s.t.\ A^\pi(s_t, a_t) = \sum_{t'=t}^{H} \gamma^{t'-t} r_{t'} - V^\pi(s_t)$$

Model-Agnostic Reinforcement Learning

Tasks: $\mathcal{T} = \{\mathcal{T}_1, \dots, \mathcal{T}_i, \dots, \mathcal{T}_I\}$

Meta-Policy: π_θ

Fine-tuned Policy: $\{\pi_{\theta_1}, \dots, \pi_{\theta_i}, \dots, \pi_{\theta_I}\}$

With Trajectory $D_i^{train} = \{(\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{r}_1, \dots, \boldsymbol{s}_{T+1})_i\}$:

$$\theta_i = \theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta, D_i^{\text{train}})$$
$$= \theta + \alpha \sum_{(s_t, a_t, r_t)\in D_i^{\text{train}}} A^\pi(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

→ fine-tuning policy

With Trajectory $D_i^{test}, \boldsymbol{\theta}_i$:

$$\theta \leftarrow \theta - \beta \sum_{\mathcal{T}_i\sim\mathcal{T}} \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta_i, D_i^{\text{test}})$$

$$s.t.\ \mathcal{L}_{\mathcal{T}_i}(\theta_i, D_i^{\text{test}}) = \mathcal{L}_{\mathcal{T}_i}(\theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta, D_i^{\text{train}}), D_i^{\text{test}})$$

→ Update meta-policy

Model-Agnostic Reinforcement Learning

1) Tasks: $T = \{T_1, \dots, T_i, \dots, T_I\}$

2) Meta-Policy: π_θ

3) Fine-tuned Policy: $\{\pi_{\theta_1}, \dots, \pi_{\theta_i}, \dots, \pi_{\theta_I}\}$

With Trajectory $D_i^{train} = \{(\boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{r}_1, \dots, \boldsymbol{s}_{T+1})_i\}$:

$$\theta_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta, D_i^{\text{train}})$$

$$= \theta + \alpha \sum_{(s_t, a_t, r_t) \in D_i^{\text{train}}} A^\pi(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

$$A^\pi(s_t, a_t) = \sum_{t'=t}^{H} \gamma^{t'-t} r_{t'} - V^\pi(s_t)$$

→ fine-tuning policy

With Trajectory $D_i^{test}, \boldsymbol{\theta}_i$:

$$\theta \leftarrow \theta - \beta \sum_{\mathcal{T}_i \sim \mathcal{T}} \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta_i, D_i^{\text{test}})$$

$$\text{s.t. } \mathcal{L}_{\mathcal{T}_i}(\theta_i, D_i^{\text{test}}) = \mathcal{L}_{\mathcal{T}_i}(\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta, D_i^{\text{train}}), D_i^{\text{test}})$$

→ Update meta-policy

No Reward Meta Learning

Goal: develop model-free meta-RL algorithm that can learn to quickly adapt a policy to dynamics changes and sensor drifts *w/o external reward*.

Learned advantage function

$$A_\psi(s_t, a_t, s_{t+1})$$

Offset learning for better exploration

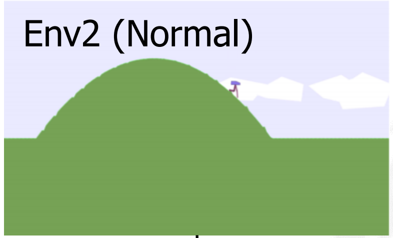
[5] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Jie Tan, Chelsea Finn,"NoRML: No-Reward Meta Learning," *AAMAS*, 2019
[6] Code Available: https://github.com/google-research/google-research/tree/master/norml

In Previous RL (Task Specific RL Method)

RL- Learner[1]

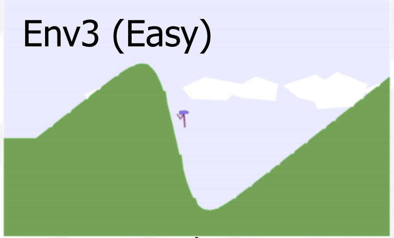
Not Trained

(sparse reward)



RL- Learner[2]

Well Trained

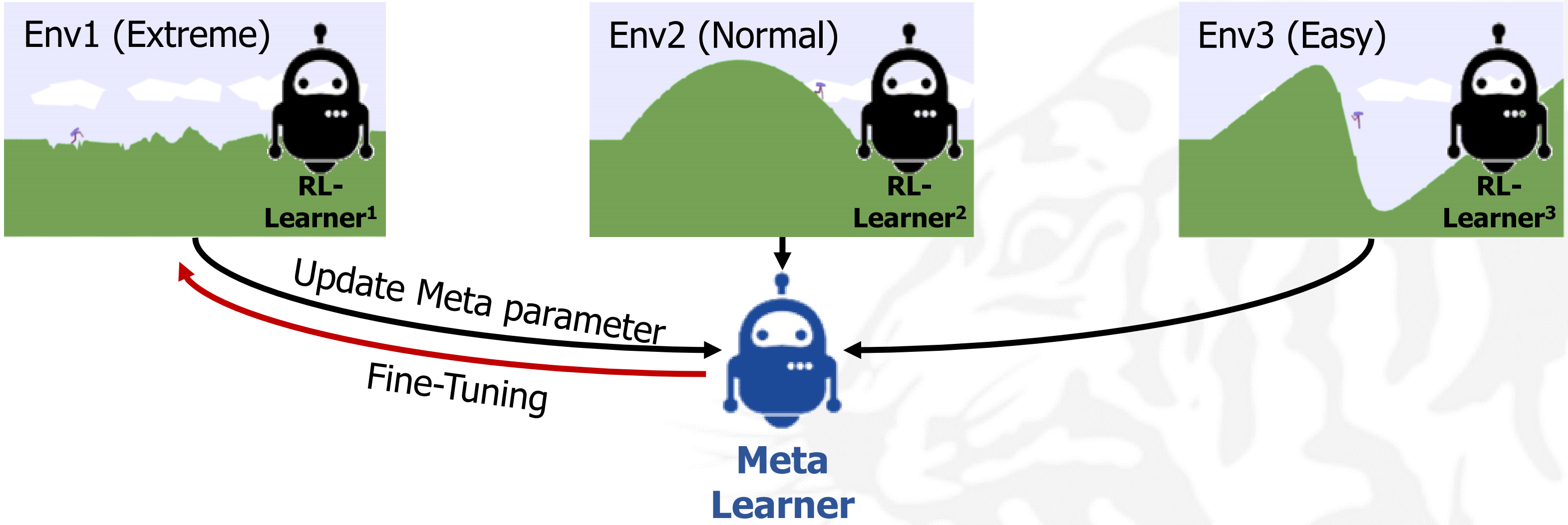


RL- Learner[3]

Well Trained

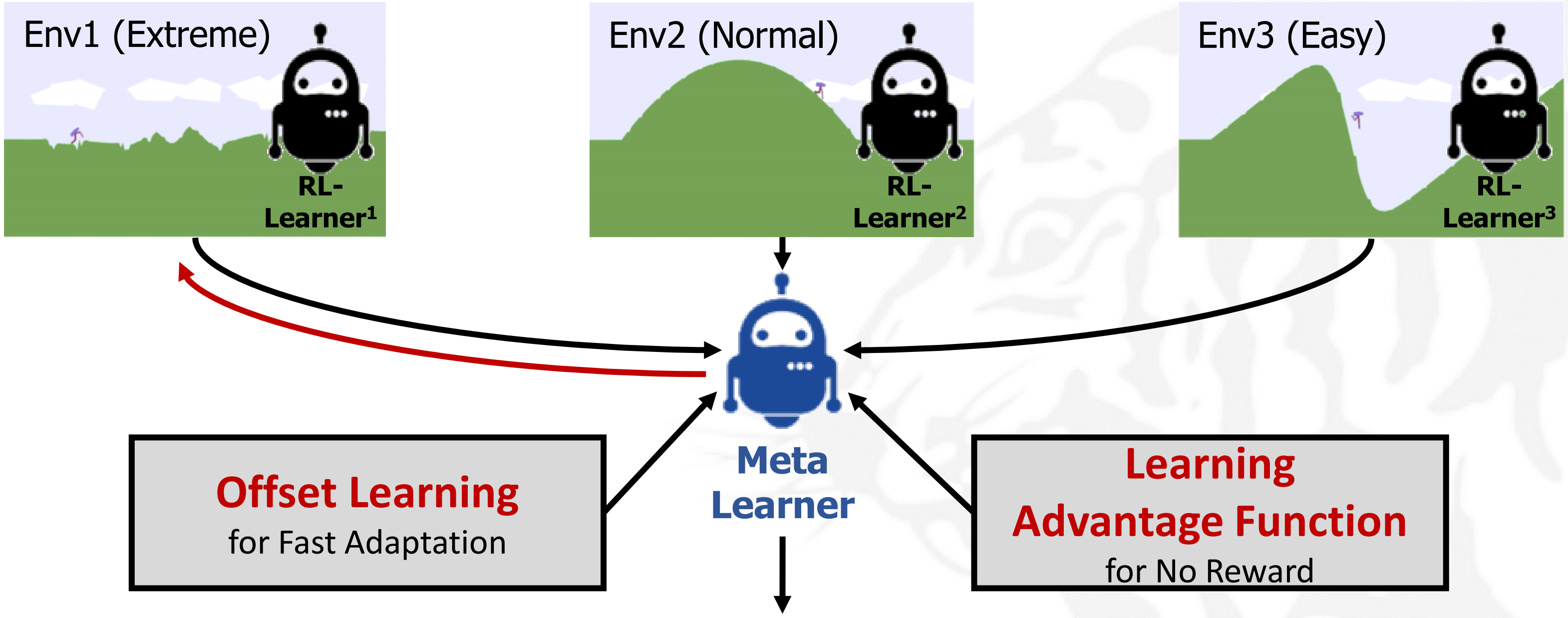
Should remodel reward by reward shaping

In MAML-RL (Task General RL Method)



Fast Adaptation & Well Trained
but, still not guarantee sparse reward environment

In NoRML-RL (Task General RL Method)



Fast Adaptation & Well Trained

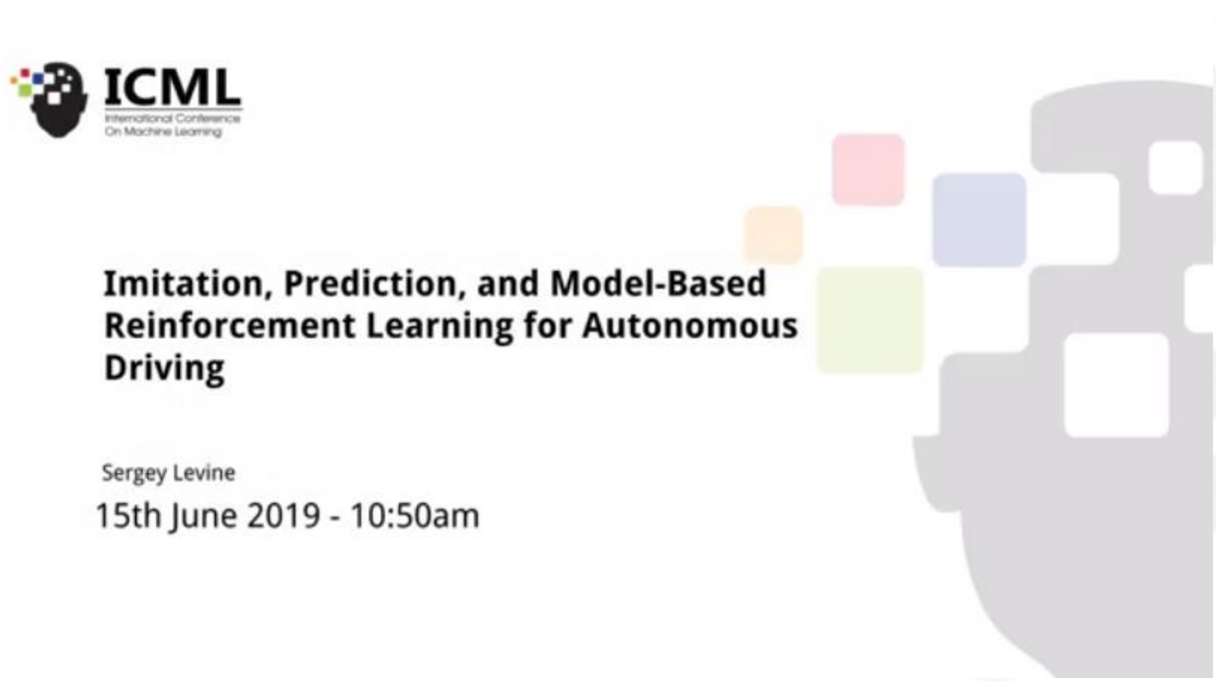
Good performance on sparse reward environment

Imitation Learning

- ICML 2018 Tutorial
  - https://sites.google.com/view/icml2018-imitation-learning/

Imitation Learning Tutorial ICML 2018

- ICML 2019 Tutorial
    - https://slideslive.com/38917941/imitation-prediction-and-modelbased-reinforcement-learning-for-autonomous-driving



- Gameplay

Pro-Gamer

Trained Agent

The goal of Imitation Learning is to train a policy to mimic **the expert's demonstrations**

- Problems of RL

1. Reward Shaping

2. Safe Learning

3. Exploration process

Imitation Learning **handles with** these problems through the demonstration of the experts.

Inverse Reinforcement Learning (IRL)

RL

Environment Model(MDP)

Reward Function R → Reinforcement Learning $\arg\max_\pi \mathrm{E}[\sum_t \gamma^t R(s_t)|\pi]$ → Optimal Policy π

IRL

Environment Model(MDP)

Reward Function R ← Inverse Reinforcement Learning ← Optimal Policy π

Expert Policy

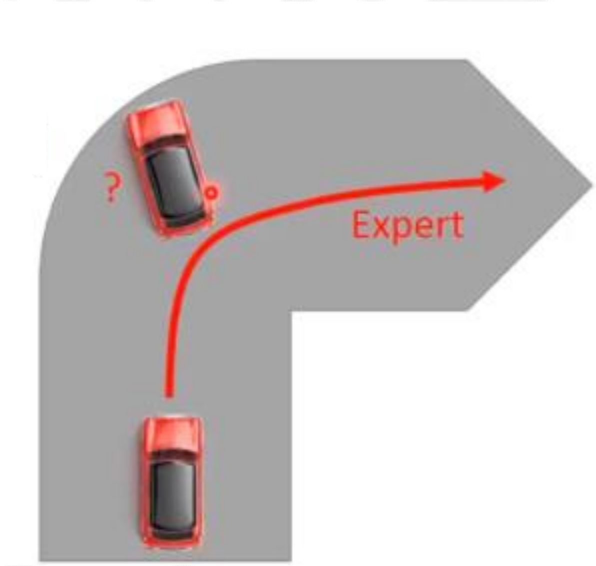
R that explains Expert Trajectories

Expert Trajectories $s_0, a_0, s_1, a_1, s_2, a_2, \cdots$

- **Behavior Cloning**
  - Define $P^* = P(s|\pi^*)$ (distribution of states visited by **expert**)
  - **Learning objective**

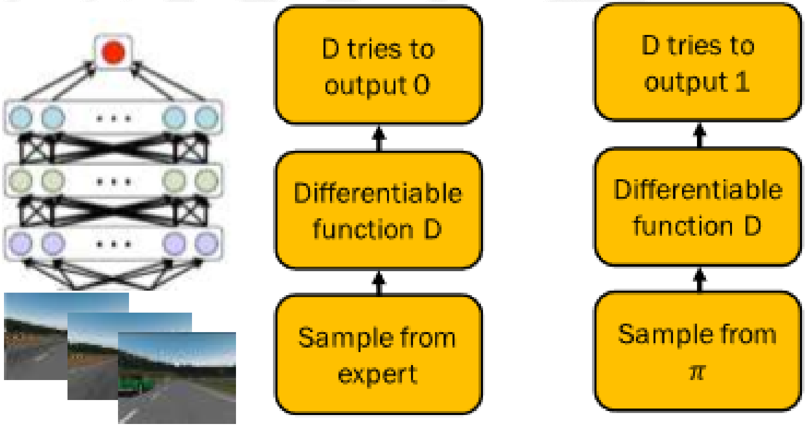
$$argmin_{\theta} \, E_{(s,a_E)\sim P^*} L(a_E, \pi_\theta(s))$$
$$L\left(a_E, \pi_\theta(s)\right) = \left(a_E - \pi_\theta(s)\right)^2$$

  - **Discussion**
    - Works well when P^* close to the distribution of states visited by π_θ
    - **Minimize 1-step deviation error** along the expert trajectories



- **Generative Adversarial Imitation Learning (GAIL)**, **NIPS 2016**
  - Generative adversarial imitation learning (GAIL) learns a policy that can imitate expert demonstration using **the adversarial network** from generative adversarial network (GAN).
  - **Learning Objective**

$$\boxed{argmin_{\theta}\ argmax_{\emptyset}\ E[\log(D_{\emptyset}(s,a)] + E[\log(1 - D_{\emptyset}(s,a))]}$$



Artificial Intelligence and Mobility Lab

- Starcraft2

States: s = **minimap**, **screen**
Action: a = **select**, **drag**
Training set: $D = \{\tau \coloneqq (s, a)\}$ from expert
Goal: learn $\pi_\theta(s) \rightarrow$ a

States: s
Action: a
Policy: π_θ

- Policy maps states to actions : $\pi_\theta(s) \rightarrow \mathrm{a}$
- Distributions over actions : $\pi_\theta(s) \rightarrow P(\mathrm{a})$

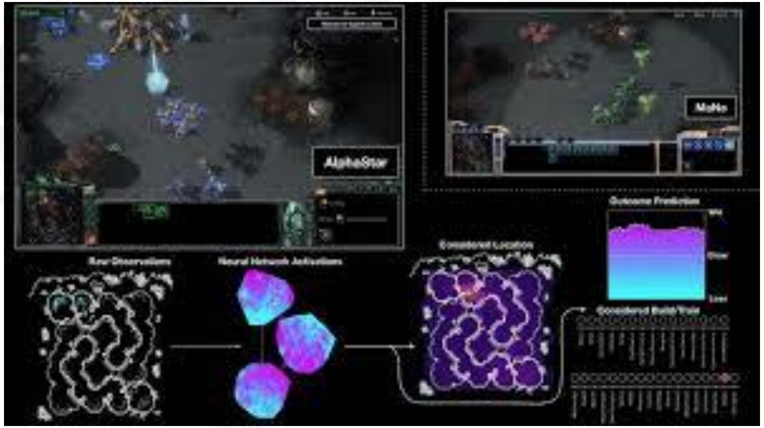
State Dynamics: $P(\mathrm{s}'|\mathrm{s,a})$

- Typically not known to policy
- Essentially the simulator/environment

Rollout: sequentially execute $\pi_\theta(s_0)$ on initial state

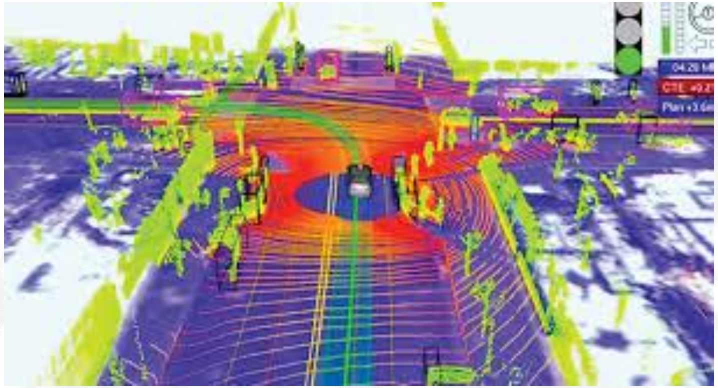
- Produce trajectories $\boldsymbol{\tau}$

$\mathbf{P}(\boldsymbol{\tau}|\boldsymbol{\pi})$**:** distribution of trajectories induced by a policy
$\mathbf{P}(\mathbf{s}|\boldsymbol{\pi})$**:** distribution of states induced by a policy



- Autonomous Driving Control

States: s = **sensors**
Action: a = **steering wheel**, **brake**, …
Training set: $D = \{\tau \coloneqq (s, a)\}$ from expert
Goal: learn $\pi_\theta(s) \rightarrow$ a

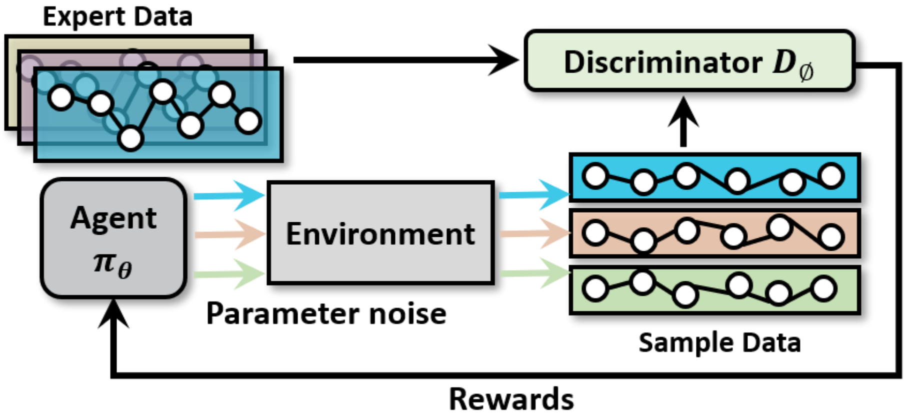


- Smartphone Security

States: s = **apps**, …
Action: a = **use patterns**, …
Training set: $D = \{\tau \coloneqq (s, a)\}$ from expert
Goal: learn $\pi_\theta(s)$ → a

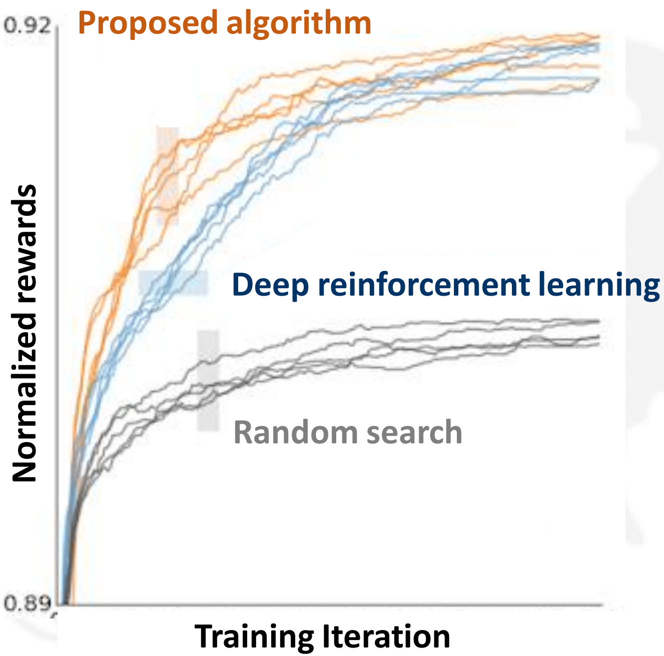
- PPF/RFTN Injection Control in Medicine

States: s = **BIS**, **BP**, …
Action: a = **PPF**, **RFTN**, …
Training set: $D = \{\tau \coloneqq (s, a)\}$ from expert
Goal: learn $\pi_\theta(s) \rightarrow$ a



M. Shin and J. Kim, "**Adversarial Imitation Learning via Random Search in Lane Change Decision-Making**," *ICML 2019 Workshop on AI for Autonomous Driving*, 2019.

M. Shin and J. Kim, "**Randomized Adversarial Imitation Learning for Autonomous Driving**," *IJCAI*, 2019., (Acceptance Rate: 850/4752=17.89%)



Generative Adversarial Network (GAN) + **Random Search**
for Autonomous Driving

Thank you for your attention!

- More questions?
  - joongheon@korea.ac.kr