TEXAS A&M UNIVERSITY
Engineering

Learning from Demonstrations: Applications to Minecraft

Student: Prabhasa Kalkur
Advisor: Dr. Dileep Kalathil

Oct 05, 2020

What is imitation learning?

Learning to imitate from expert behavior

Sample-efficient learning: learn behavior from as little expert data as possible

What is the presentation about?

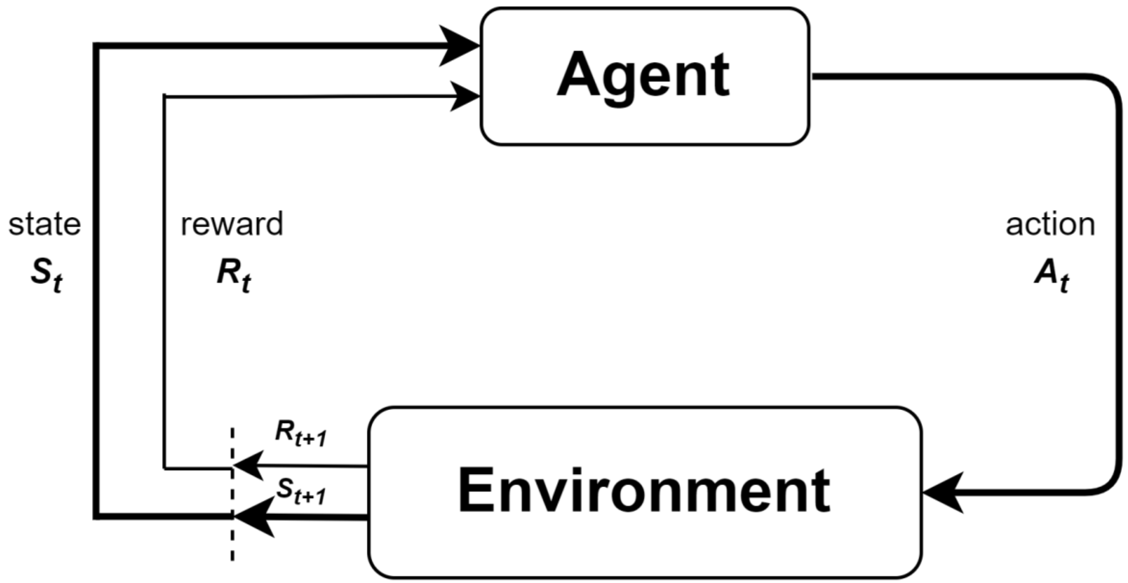
TEXAS A&M UNIVERSITY Engineering

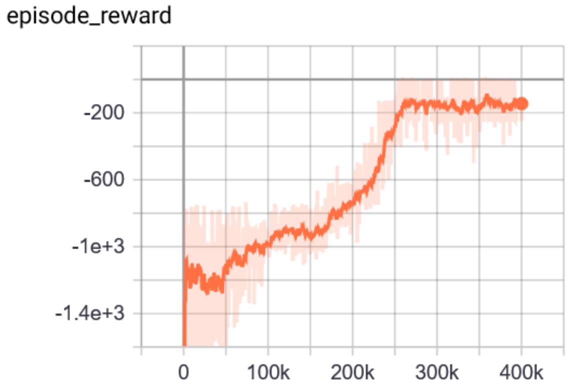
- Motivate the need for **sample-efficient** methods for learning behavior
- Pair vanilla RL algorithms with demonstration data to learn desired behavior
- Discuss potential of sample-efficient learning to solve complex tasks in Minecraft

Reinforcement Learning

TEXAS A&M UNIVERSITY
Engineering

- $s, s' \in S, a \in A$. Consider tuple $[\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}), \boldsymbol{R}(\boldsymbol{s}, \boldsymbol{a}), \gamma, H]$, define a policy (model) $\pi : S \rightarrow A$
  - **Reinforcement Learning (RL):** find an optimal π^* that maximizes $\sum_{t=0}^{\infty} \gamma^t R_t$





100 episodes of policy:
95/100 successful
Reward (mean, std): (-**175**, 50)

true reward/cost/score

↑

Performance metrics

↓

Rollouts: (S, A) **from** π^*

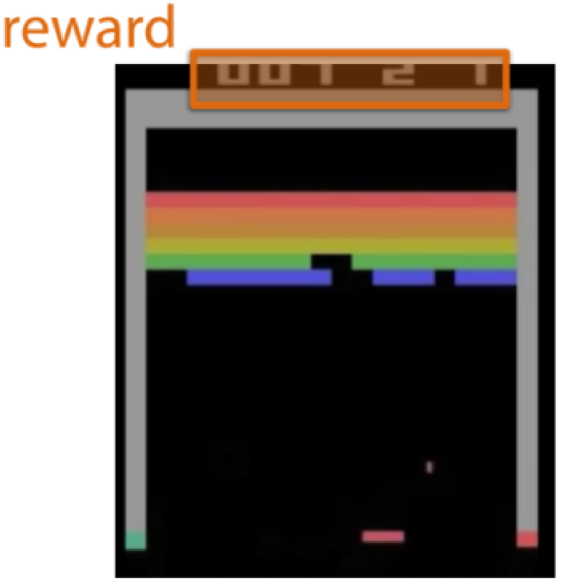
Organization of the talk

Sections

1. Need for sample-efficiency
2. Introduction to Imitation Learning
3. Application: Minecraft
4. Conclusions and Future Work

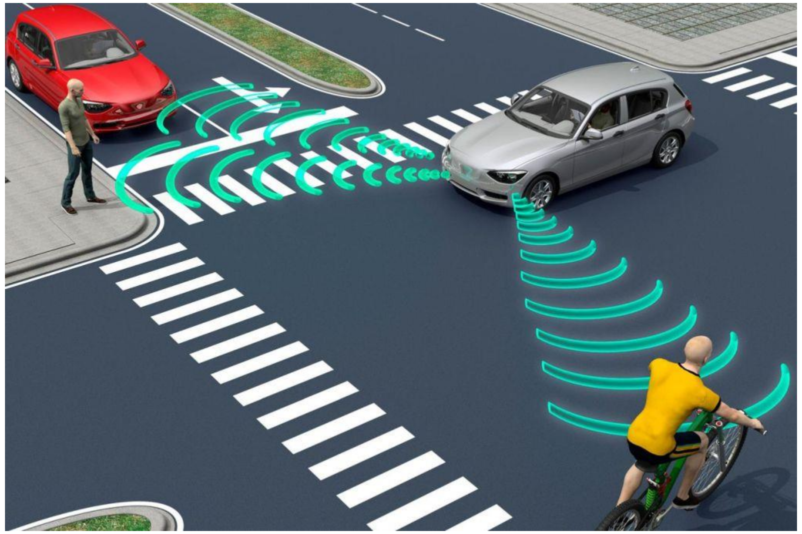
Why study imitation learning?

1. Rewards obvious in computer games: maximize score
   – Not so obvious in real-word scenarios: use a proxy instead



Mnih et al.'15

VS



Why study imitation learning?

2. Can be easier to **demonstrate** desired behavior

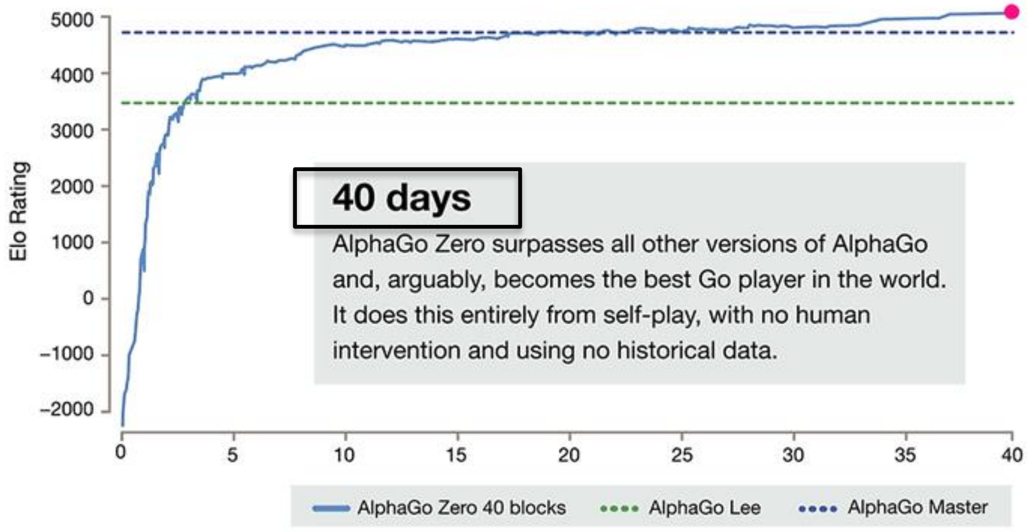
Levine et al. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection."

Why study imitation learning?

TEXAS A&M UNIVERSITY Engineering

3. Modern Deep-RL requires exponentially increasing number of samples: **sample-inefficient**

   – Challenging for the AI community to reproduce SOTA results



Go: AlphaGo Zero

	OPENAI 1V1 BOT	OPENAI FIVE
CPUs	60,000 CPU cores on Azure	128,000 preemptible CPU cores on GCP
GPUs	256 K80 GPUs on Azure	256 P100 GPUs on GCP
Experience collected	~300 years per day	~180 years per day (~900 years per day counting each hero separately)
Size of observation	~3.3 kB	~36.8 kB
Observations per second of gameplay	10	7.5
Batch size	8,388,608 observations	1,048,576 observations
Batches per minute	~20	~60

Dota 2: OpenAI Five

Why study imitation learning?

TEXAS A&M UNIVERSITY Engineering

3. Modern Deep-RL requires exponentially increasing number of samples
   - **Not practical, especially when env samples are expensive, and compute is limited**
   - One approach: use sample-efficient methods like Imitation Learning

Many competitions trying to promote compute and sample-efficient learning:

- **NeurIPS 2019:** Game of Drones
- **NeurIPS 2019 & 2020:** MineRL Challenge

Why study imitation learning?

4. How humans and animals fundamentally learn behavior

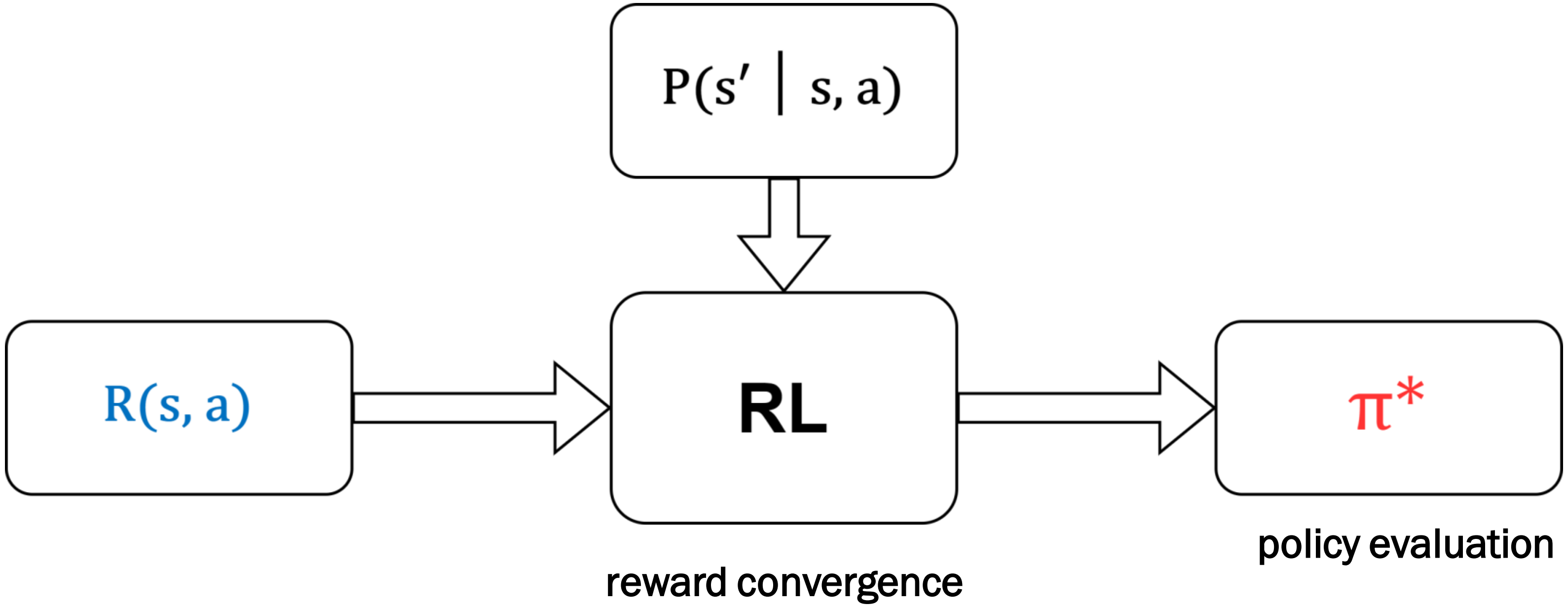
Picture credits: Sapana

Sections

1. Need for sample-efficiency
2. Introduction to Imitation Learning
3. Application: Minecraft
4. Conclusions and Future Work

RL algorithms

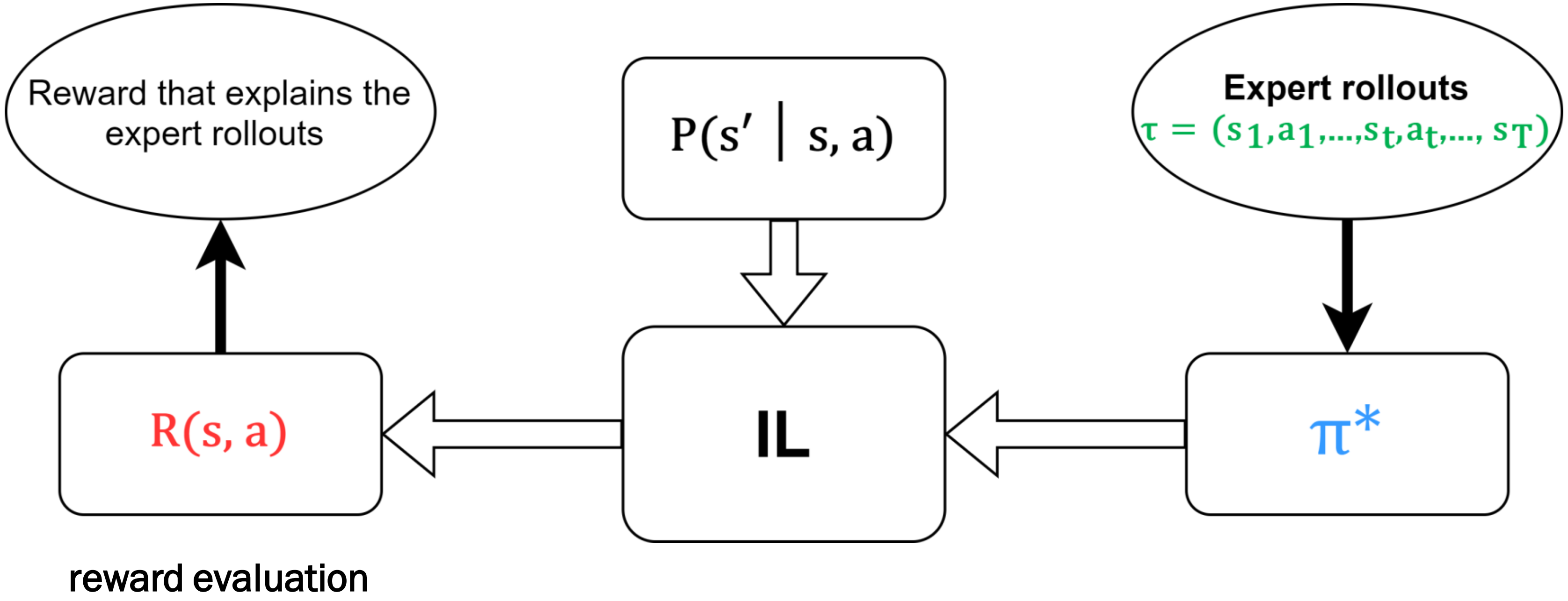
- $s, s' \in S, a \in A$. For MDP $[\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a}), \boldsymbol{R}(\boldsymbol{s}, \boldsymbol{a}), \gamma]$, define a policy $\boldsymbol{\pi} : S \rightarrow A$
  - **Goal:** find an optimal $\boldsymbol{\pi}^*$ that maximizes $\sum_{t=0}^{\infty} \boldsymbol{\gamma}^t \boldsymbol{R}_t$
  - **Metric:** (i) Reward convergence, (ii) Policy evaluation (testing)



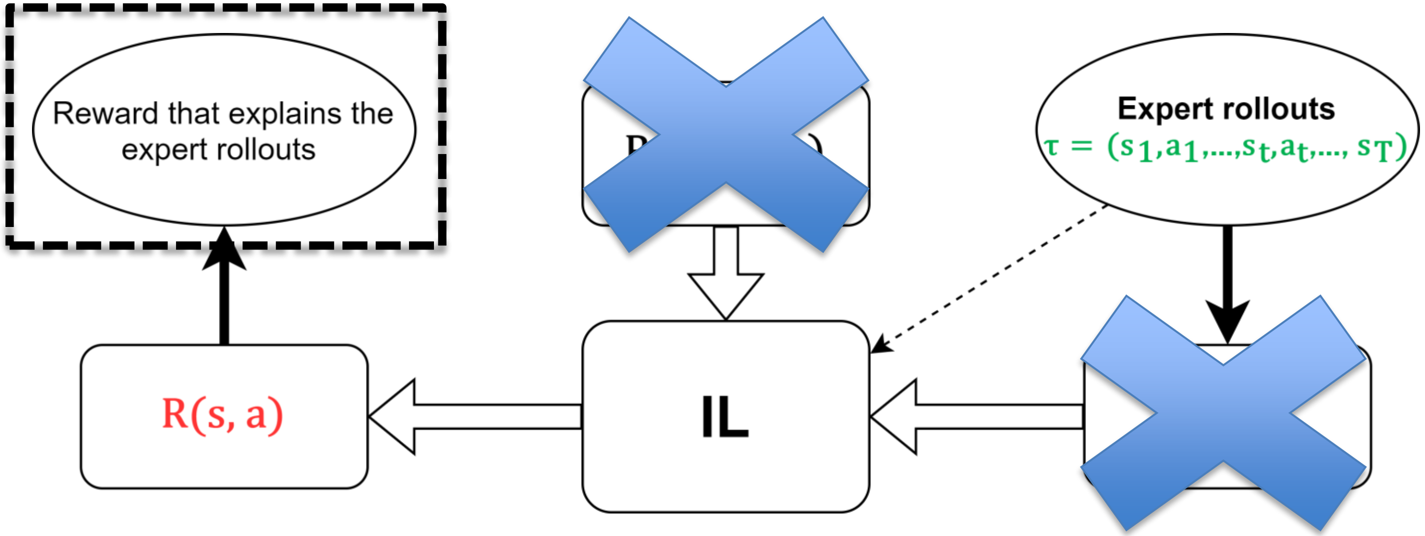
IL algorithms

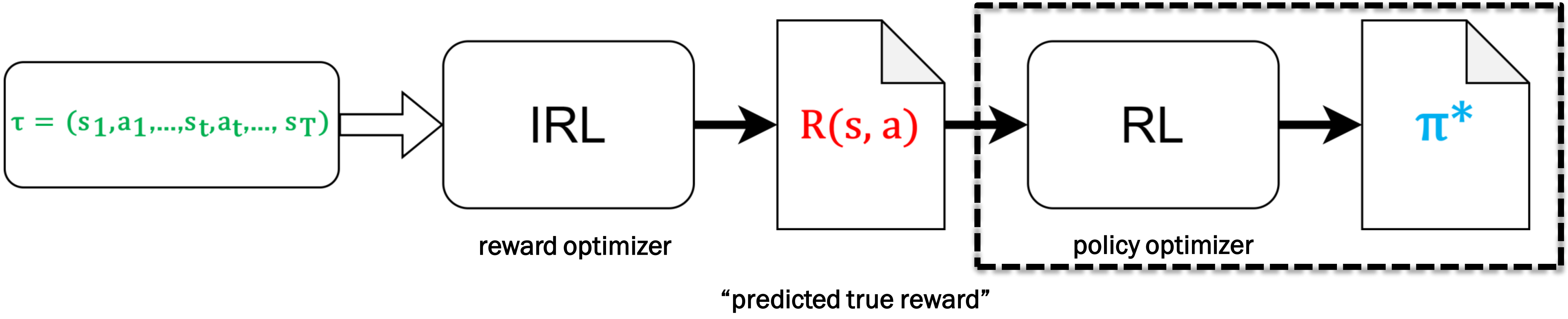
TEXAS A&M UNIVERSITY Engineering

- $s, s' \in S, a \in A$. For MDP $[\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}), \boldsymbol{R}(\boldsymbol{s}, \boldsymbol{a}), \gamma]$, define a policy $\boldsymbol{\pi} : S \rightarrow A$
  - **Goal:** given $\boldsymbol{\tau} = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_t, \boldsymbol{a}_t, \ldots, \boldsymbol{s}_T)$ generated from a $\boldsymbol{\pi}^*$, extract its $\boldsymbol{R}(\boldsymbol{s}, \boldsymbol{a})$
  - **Metric:** Reward evaluation (?)

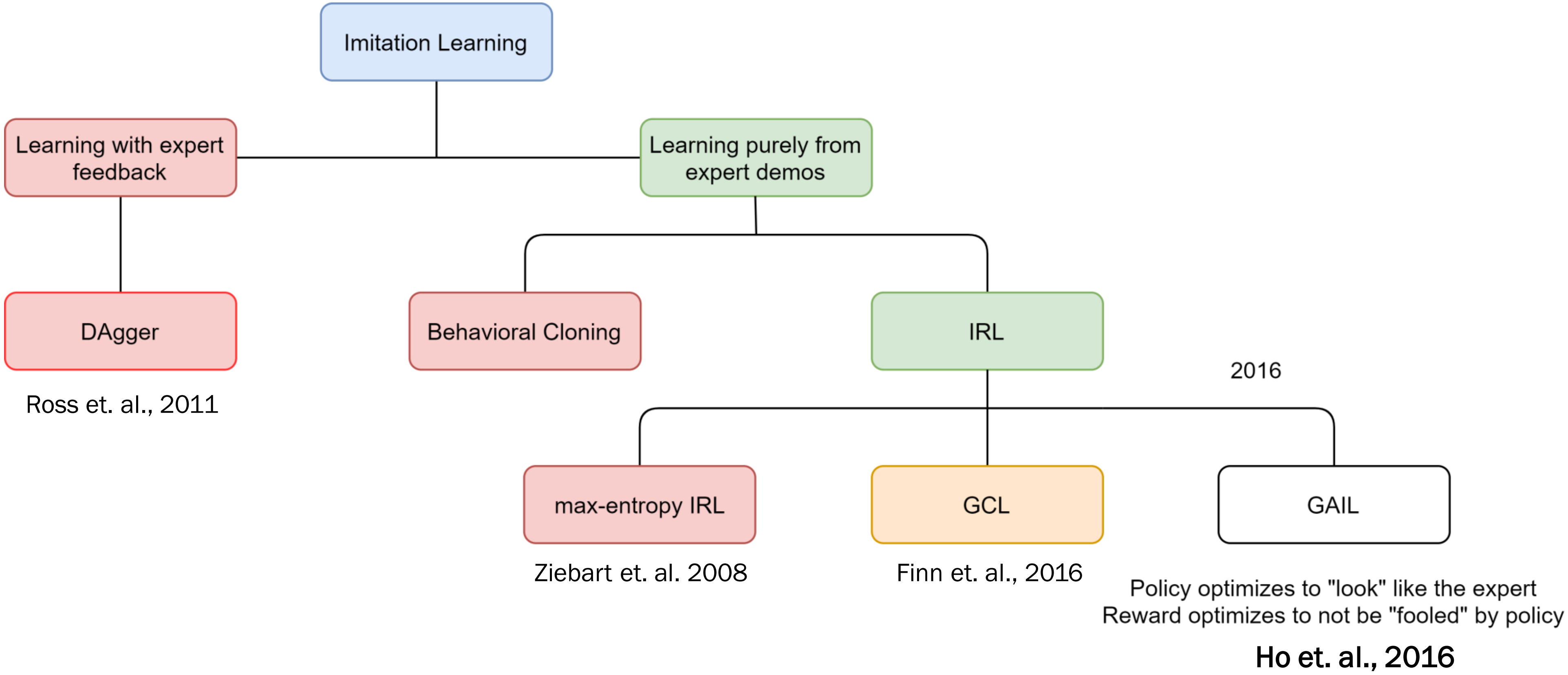


Flowchart credits: Sapana


Reward that explains the expert rollouts
Expert rollouts
$\tau = (s_1, a_1, \ldots, s_t, a_t, \ldots, s_T)$
$R(s, a)$
IL

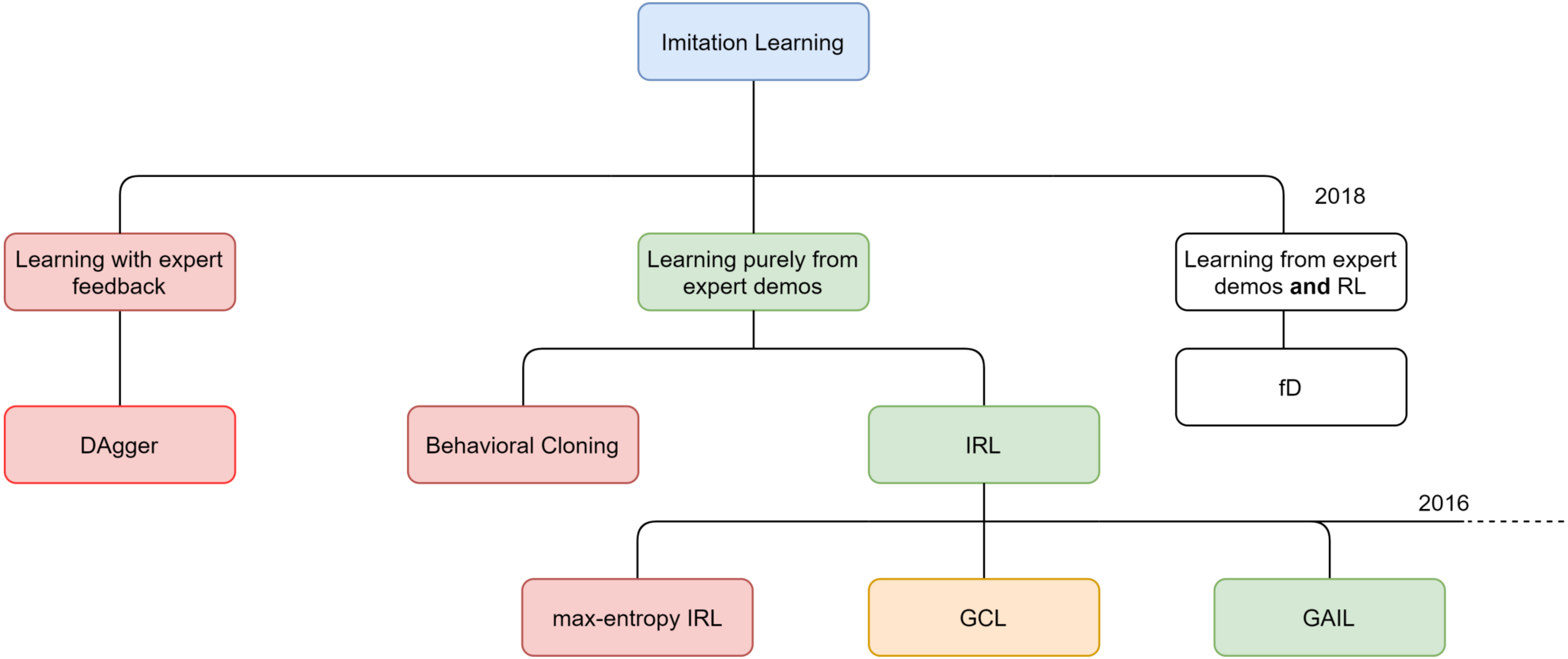

$\tau = (s_1, a_1, \ldots, s_t, a_t, \ldots, s_T)$
IRL
reward optimizer
$R(s, a)$
"predicted true reward"
RL
policy optimizer
π^*

Imitation Learning approaches



Generative Adversarial Imitation Learning (GAIL) is the **SOTA IL algorithm**

RL with human priors (RL + IL!)



Some questions…

1. How does imitation accuracy scale with problem dimensionality and demo data?
2. How ‘smooth’ are the learned policies compared to the expert policy?
3. Can behaviors with sparse rewards be learned? At what cost?
4. Can RL+IL imitate suboptimal experts? At what cost?

Let us learn how to imitate a simple control task: balance an inverted pendulum!

Problem setup

Train RL -> rollout **expert** -> Train GAIL -> **policy** evaluation (test)

Goal: GAIL should be able to ‘imitate’ expert (optimal/suboptimal?)

Discuss: imitation accuracy, sample efficiency, effect of reward quality on learning

- **Expert trajectories / rollout / demonstrations:** sample demos [5, 10, 20]
- **Policy evaluation / rollout / testing:** Check policy performance for 100 episodes
- **Task solved each episode:** True reward for 100 consecutive episodes during training

Tools

- **RL library:** Stable Baselines 2.10
- **Framework:** TensorFlow 1.14
- **Hyperparameters (HPs):** RL Baselines Zoo, etc.
- **Performance metrics (learned reward vs episodes, test scores):** Tensorboard 1.14, W&B 0.10

RL/IL Algorithms

- **SAC** – Soft Actor-Critic (optimal experts)
- **TRPO** – Trust Region Policy Optimization (policy optimizer for GAIL)
- **BC** - Behavioral Cloning* (comparison with GAIL)

*with policy: “MlpPolicy” [100, 100], optimizer: Adam, batch size: 256, train-val: 70-30

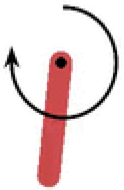
OpenAI Gym and MuJoCo

TEXAS A&M UNIVERSITY
Engineering

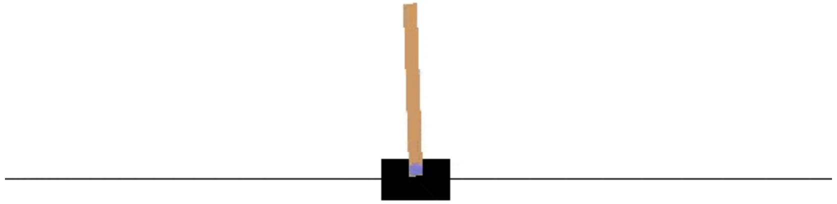
- **Gym:** "Toolkit for developing and comparing reinforcement learning algorithms"
- Platform for teaching agents to perform simulated tasks under a true reward
- E.g. Atari games, Robotic manipulation, control tasks

- **MuJoCo:** "A physics engine that does very detailed, efficient simulations with contacts"
- E.g. Continuous control tasks like hopping, walking, or running

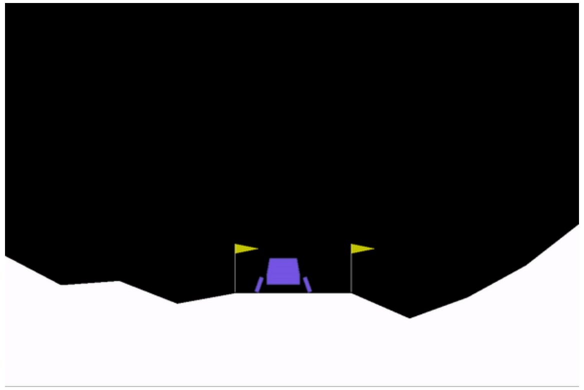
- **Why is this important?** Standard benchmark tasks for testing RL, IL algorithms



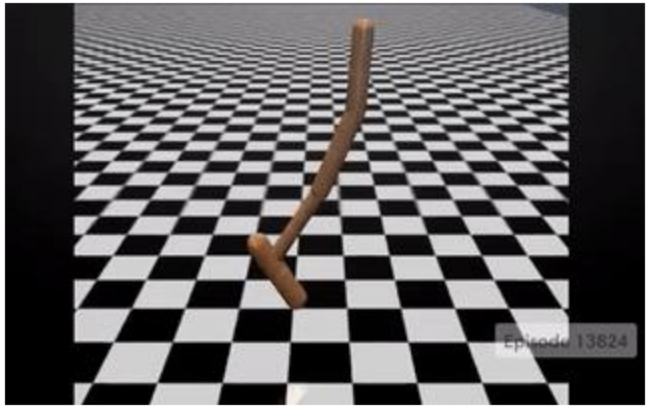
Pendulum-v0



CartPole-v1



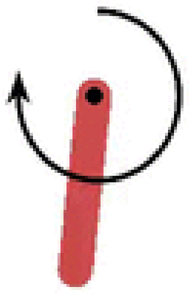
LunarLanderCts-v2



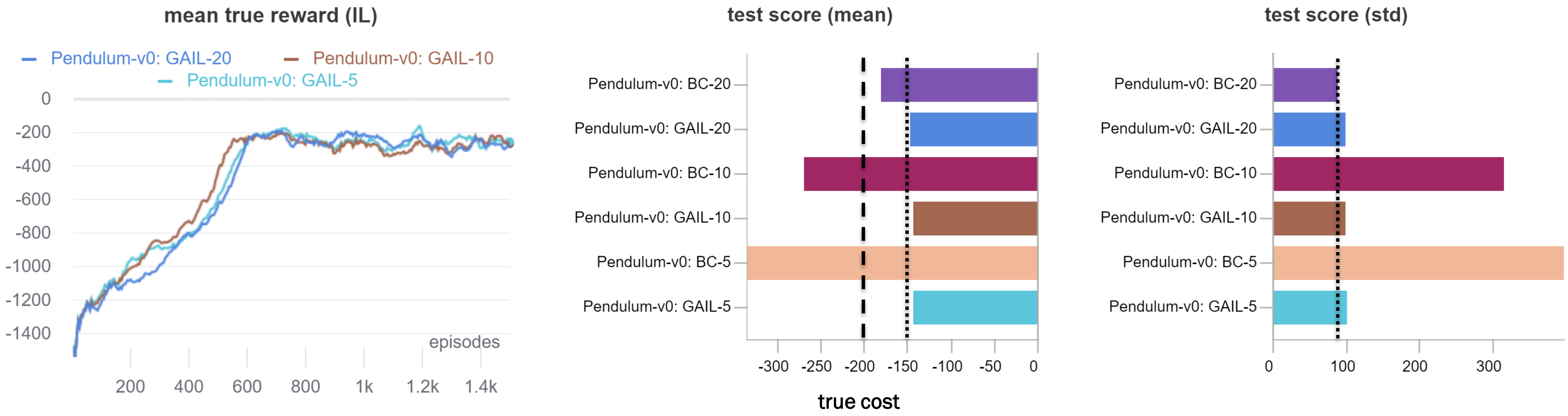
Hopper-v2

The Pendulum-v0 environment

Properties	Description
State space (cts, dim = 3)	Cosine, sine of angle θ [-1, 1], θ_0 [-8, 8]
Action space (cts, dim = 1)	Joint effort [-2, 2]
Reward	$-(\theta^2 + 0.1*\theta_0^2 + 0.001*action^2)$, **dense**
Termination / Horizon	200 steps, **finite**
Solved / learned task	defined as **-200** mean reward over 100 consecutive episodes of training
Expert Trajectories for IL	**[5, 10, 20]** with reward (mean, var): **(-147, 84)**



Pendulum-v0: GAIL and BC



reward convergence

policy evaluation

GAIL learns to achieve true cost **AND** imitate expert
GAIL score (mean, var) consistent over # demos – **sample-efficient**
BC improves over # demos, but only for optimal experts

Some questions...

TEXAS A&M UNIVERSITY Engineering

1. How does imitation accuracy scale with dimensionality, demo data? GAIL sample-efficient (low-dim)
2. How 'smooth' are the learned policies compared to the expert policy? **Demo-dependent**
3. Can behaviors with sparse rewards be learned? At what cost?
4. Can GAIL imitate suboptimal experts? At what cost? **BC cannot.** GAIL can, with the right HPs
5. Can GAIL generalize?

Answered for a **low-dimensional, densely-rewarded, finite-horizon** control task. Let's try harder!

Sections

1. Need for sample-efficiency
2. Introduction to Imitation Learning
3. Application: Minecraft
4. Conclusions and Future Work

MineRL: Chopping trees and mining a Diamond in Minecraft

APPLICATION 2

MineRL Competition: NeurIPS 2020

- Lack of large-scale imitation learning datasets
- **MineRL:** a large-scale dataset of seven different tasks on Minecraft (60 mil pairs)

Why Minecraft:

- Open-world env, sparse rewards, many innate task hierarchies and sub-goals
- 90 million monthly active users, easy to collect a large-scale dataset
- Env simulator available: Microsoft Malmo

MineRL Competition: Description

TEXAS A&M UNIVERSITY Engineering

- Competition on sample-efficient reinforcement learning using human priors

- Address two crucial challenges in RL. Solving hierarchical environments with
  - Sparse rewards
  - Long time horizon

- Develop algorithms to mine a Diamond object in Minecraft using limited
  - Train time (4 days)
  - Compute (single GPU)
  - Samples from the environment simulator (8 million)

MineRL Competition: Solution approaches

TEXAS A&M UNIVERSITY Engineering

- "…highlight a variety of research challenges, including open-world multi-agent interactions, long-term planning, vision, control, navigation, and explicit and implicit subtask hierarchies"

- Want to avoid massive datasets and hand-engineered features

- Complex, hierarchical, sparsely-rewarded task that demands use of:
  - Efficient exploration techniques
  - Training with human priors (e.g. fD algorithms) ☑
  - Reward shaping using IL techniques

MineRL Competition: Details

TEXAS A&M UNIVERSITY Engineering

- Two competition tracks:
  - **Demonstrations and Environment:** MineRL dataset + 8M env interactions ☑
  - **Demonstrations Only:** MineRL dataset only

- What's new from 2019: Vectorized state, action space that **obfuscates** the agent's actions
  - Prevent participants from using domain knowledge
  - **State:** images + 1-D vector containing comprehensive set of features from the game
  - **Actions:** 1-D vector containing keyboard presses, mouse movements (pitch, yaw), player GUI interactions, and agglomerative actions such as item crafting

Visualizing the MineRL envs & dataset

MineRLTreeChopVectorObf-v0: https://youtu.be/q9DtmFJMc5I

MineRLObtainDiamondVectorObf-v0: https://youtu.be/mexGyw1PoT0

Competition Environments

MineRLTreechopVectorObf-v0

In treechop, the agent must collect 64 *minercaft:log*. This replicates a common scenario in Minecraft, as logs are necessary to craft a large amount of items in the game, and are a key resource in Minecraft.

The agent begins in a forest biome (near many trees) with an iron axe for cutting trees. The agent is given +1 reward for obtaining each unit of wood, and the episode terminates once the agent obtains 64 units.

Observation Space

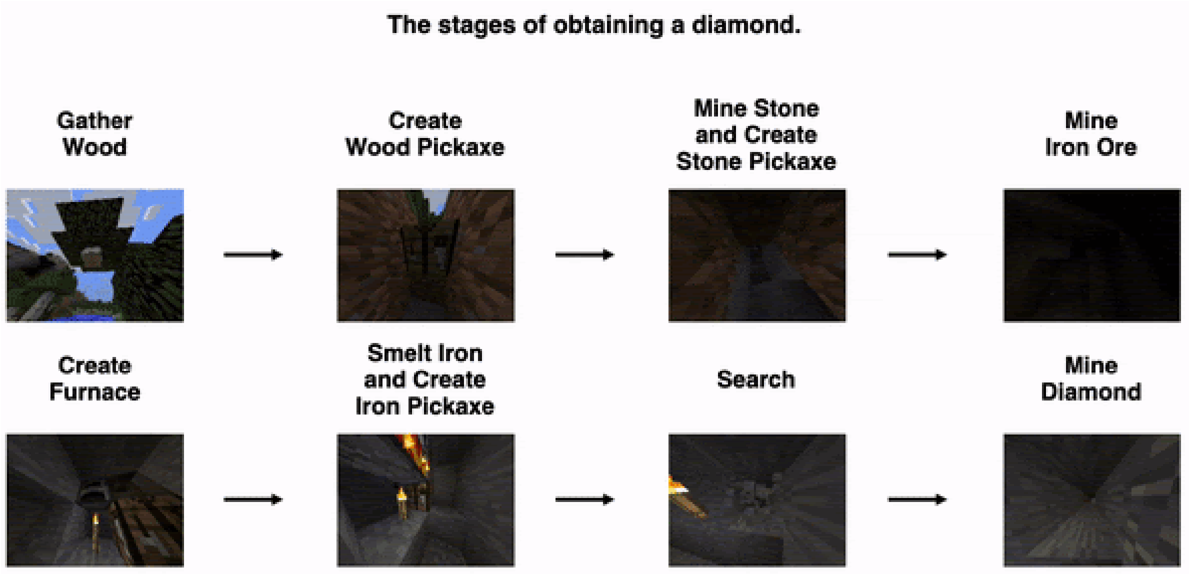
```
Dict({
    "pov": "Box(low=0, high=255, shape=(64, 64, 3))",
    "vector": "Box(low=-1.2000000476837158, high=1.2000000476837158, shape=(64,))"
})
```

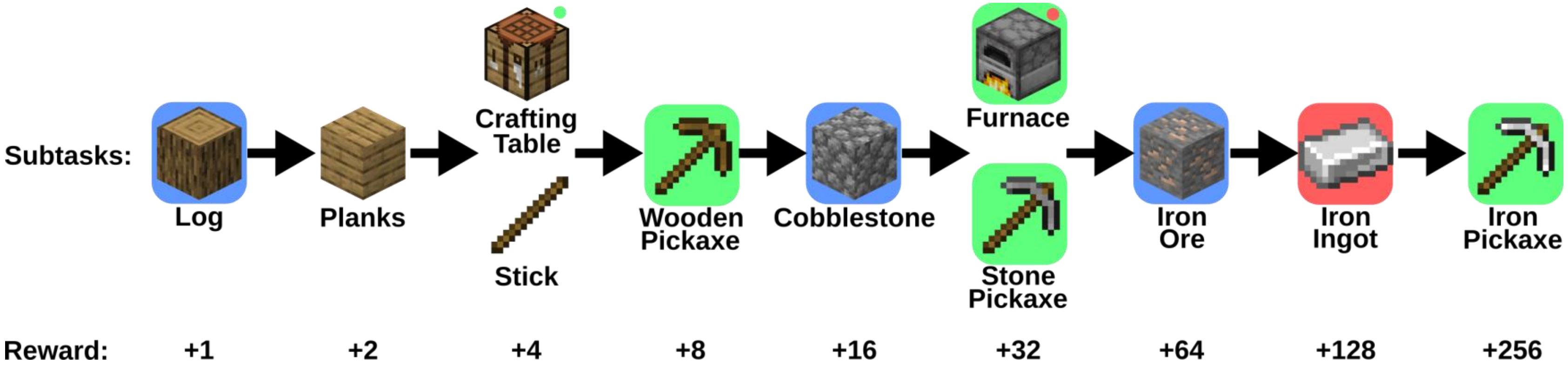
Action Space

```
Dict({
    "vector": "Box(low=-1.0499999523162842, high=1.0499999523162842, shape=(64,))"
})
```

```
"attack": "Discrete(2)",
"back": "Discrete(2)",
"camera": "Box(low=-180.0, high=180.0, shape=(2,))",
"forward": "Discrete(2)",
"jump": "Discrete(2)",
"left": "Discrete(2)",
"right": "Discrete(2)",
"sneak": "Discrete(2)",
"sprint": "Discrete(2)"
```

Obtain Diamond: Tasks and Rewards





Observation Space

```
Dict({
    "equipped_items.mainhand.damage": "Box(low=-1, high=1562, shape=())",
    "equipped_items.mainhand.maxDamage": "Box(low=-1, high=1562, shape=())",
    "equipped_items.mainhand.type": "Enum(air,iron_axe,iron_pickaxe,none,other,stone_axe,stone_pickaxe,w
    "inventory": {
            "coal": "Box(low=0, high=2304, shape=())",
            "cobblestone": "Box(low=0, high=2304, shape=())",
            "crafting_table": "Box(low=0, high=2304, shape=())",
            "dirt": "Box(low=0, high=2304, shape=())",
            "furnace": "Box(low=0, high=2304, shape=())",
            "iron_axe": "Box(low=0, high=2304, shape=())",
            "iron_ingot": "Box(low=0, high=2304, shape=())",
            "iron_ore": "Box(low=0, high=2304, shape=())",
            "iron_pickaxe": "Box(low=0, high=2304, shape=())",
            "log": "Box(low=0, high=2304, shape=())",
            "planks": "Box(low=0, high=2304, shape=())",
            "stick": "Box(low=0, high=2304, shape=())",
            "stone": "Box(low=0, high=2304, shape=())",
            "stone_axe": "Box(low=0, high=2304, shape=())",
            "stone_pickaxe": "Box(low=0, high=2304, shape=())",
            "torch": "Box(low=0, high=2304, shape=())",
            "wooden_axe": "Box(low=0, high=2304, shape=())",
            "wooden_pickaxe": "Box(low=0, high=2304, shape=())"
    },
    "pov": "Box(low=0, high=255, shape=(64, 64, 3))"
})
```

Action Space

```
t({
 "attack": "Discrete(2)",
 "back": "Discrete(2)",
 "camera": "Box(low=-180.0, high=180.0, shape=(2,))",
 "craft": "Enum(crafting_table,none,planks,stick,torch)",
 "equip": "Enum(air,iron_axe,iron_pickaxe,none,stone_axe,stone_pickaxe,wooden_axe,wooden_pickaxe)",
 "forward": "Discrete(2)",
 "jump": "Discrete(2)",
 "left": "Discrete(2)",
 "nearbyCraft": "Enum(furnace,iron_axe,iron_pickaxe,none,stone_axe,stone_pickaxe,wooden_axe,wooden_pickaxe
 "nearbySmelt": "Enum(coal,iron_ingot,none)",
 "place": "Enum(cobblestone,crafting_table,dirt,furnace,none,stone,torch)",
 "right": "Discrete(2)",
 "sneak": "Discrete(2)",
 "sprint": "Discrete(2)"
```

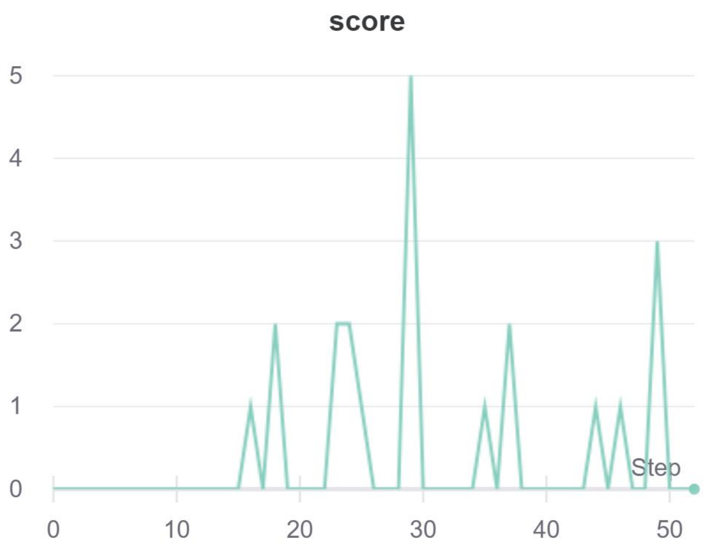
Tools

- **RL library:** Medipixel 0.10
- **Framework:** Pytorch 1.3.1
- **Hyperparameters (HPs):** Medipixel 0.10
- **Results (train score vs episodes, test score):** W&B 0.10

DQN (RL) vs DQfD (RL+IL)

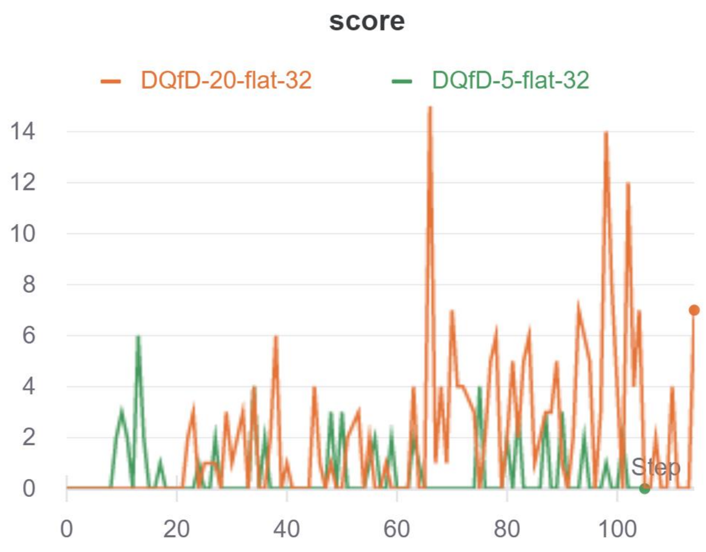
MineRLTreeChopVectorObf-v0: https://youtu.be/YDpVRyZndCg

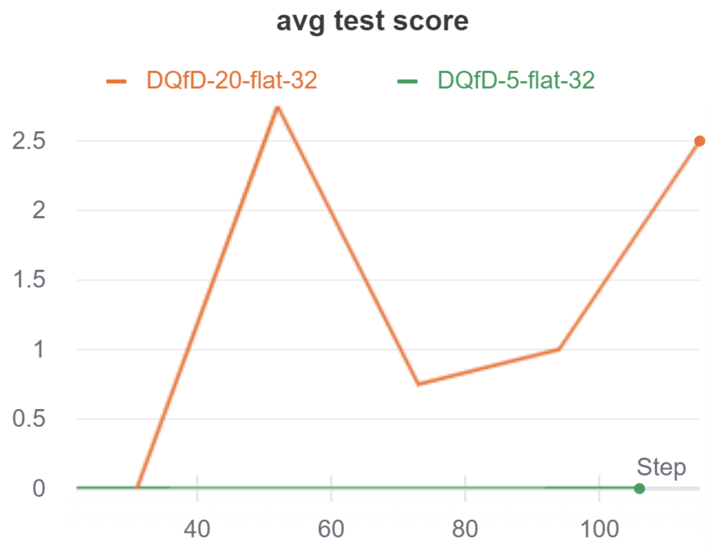
MineRLObtainDiamondVectorObf-v0: https://youtu.be/b-SGp7PKbxM


score
Step


avg test score
Step

DQN


score
DQfD-20-flat-32
DQfD-5-flat-32
Step


avg test score
DQfD-20-flat-32
DQfD-5-flat-32
Step

DQfD

Submitted to MineRL Competition: NeurIPS 2020

TEXAS A&M UNIVERSITY
Engineering

85764	prabhasak	submitted	0.000	0.000	RL+ IL	Thu, 1 Oct 2020	View	Code

Δ	#	Participants	Media	Reward	N/A	tags	Entries
	01	NoActionWasted	-	9.64	0.0	IL	15
	02	michal_opano...	-	9.29	0.0	IL	11
▲	03	CU-SF	-	6.47	0.0	RL+ IL	12
▲	04	HelloWorld	-	6.01	0.0	RL+ IL	7
	05	NuclearWeapon	-	4.34	0.0	RL+ IL	7

Sections

1. Need for sample-efficiency
2. Introduction to Imitation Learning
3. Application: Minecraft
4. Conclusions and Future Work

Tasks Studied

Task	*State dim	*Action dim	Reward quality	Termination, Horizon	Imitation successful?
Pendulum-v0	3C	1C	Dense	Fixed, small	**Yes**
LunarLanderCts-v2	4C + 2D	2D	Semi-sparse	Not fixed, large	**Yes**
Hopper-v2	11C	3C	Dense	Fixed, large	**Yes (better)**
AirSim-v0	6C	3C	Sparse	Fixed, small	****Yes**
MineRLTreechopVectorObf-v0	pov: 64x64x3 vector: 64C	64C (64D)	(extremely) sparse	Fixed, very large	**No**
MineRLObtainDiamondVectorObf	pov: 64x64x3 vector: 64C	64C (64D)	(extremely) sparse	Not fixed, very large	**No**

*C: continuous. D: discrete

**Suboptimal landings, >20 optimal demos

RL vs IL

Learning with a cost function vs imitating with demo data

Control Task	Reward	Task length (max)	Episodes / env interactions (RL)	Episodes / env interactions (IL)	Converged in episodes (RL)	Converged in episodes (IL)
Pendulum-v0	Dense	200	500 (1e5)	1500 (3e5)	200	800
CartPole-v1	Dense	500	500 (1e5)	2500 (1e6)	400	1400
LunarLanderCts-v2	Sparse	N/A	1300 (5e5)	1650 (1e6)	**800**	**1000**
Hopper-v2	Dense	1000	5300 (2e6)	3400 (2e6)	**3500**	**2500**
AirSim-v0	Sparse	400	10k (5e5)	16k (1e6)	**3000**	**7300**

CONCLUSIONS

TEXAS A&M UNIVERSITY Engineering

- Need sample-efficient learning for complex, long-horizon tasks
- IL (GAIL) is a sample-efficient approach to learn from demonstrations
- IL can be used to imitate (even suboptimal) experts from sparsely-rewarded environments
  - Requires smooth experts and careful HP tuning for perfect imitation
- Application: Discussed potential of IL + RL on a complex, sparse, long-horizon, hierarchical task

Future Extensions: MineRL

- Use CNNs to learn representations from the image
- Employ hierarchical learning, multi-agent RL to learn implicit/explicit hierarchies in tasks
- Train on datasets of individual tasks in hierarchy, to bring in diversity among demonstrations
- Algorithmic contributions for sparsely-rewarded, hierarchical tasks with long-horizons
- LeNS Lab should participate in MineRL NeurIPS 2021!

TEXAS A&M UNIVERSITY

Engineering

THANK YOU!

QUESTIONS?