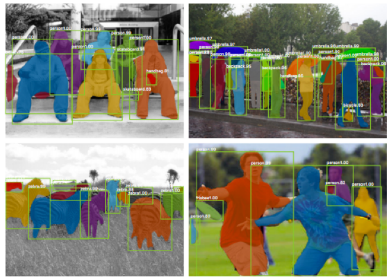
Building Scalable Framework and Environment of Reinforcement Learning

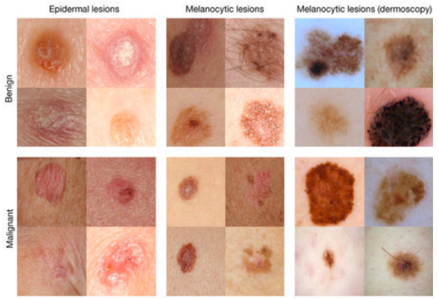
Yuandong Tian

Facebook AI Research

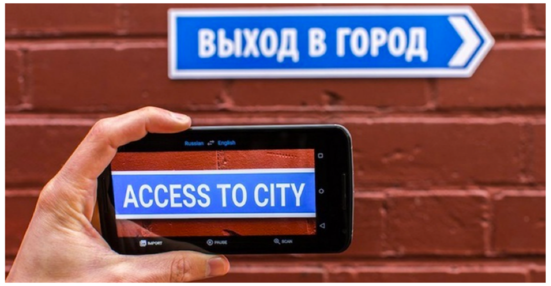
AI works in a lot of situations



Object Recognition

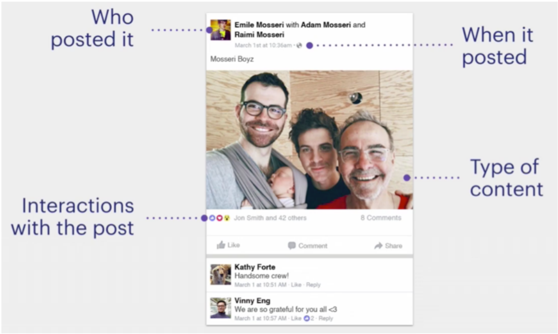


Medical



Translation

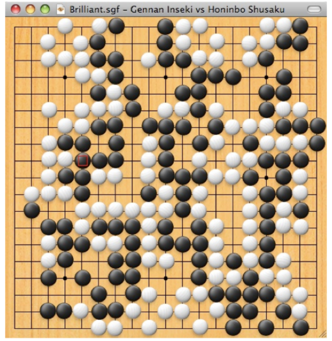
Speech Recognition



Personalization

Surveillance

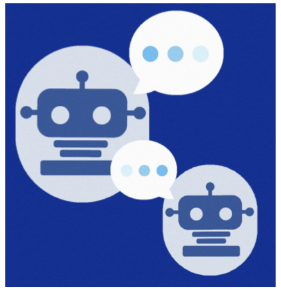
Smart Design



Board game

What AI still needs to improve

Question Answering



ChatBot

Common Sense

StarCraft

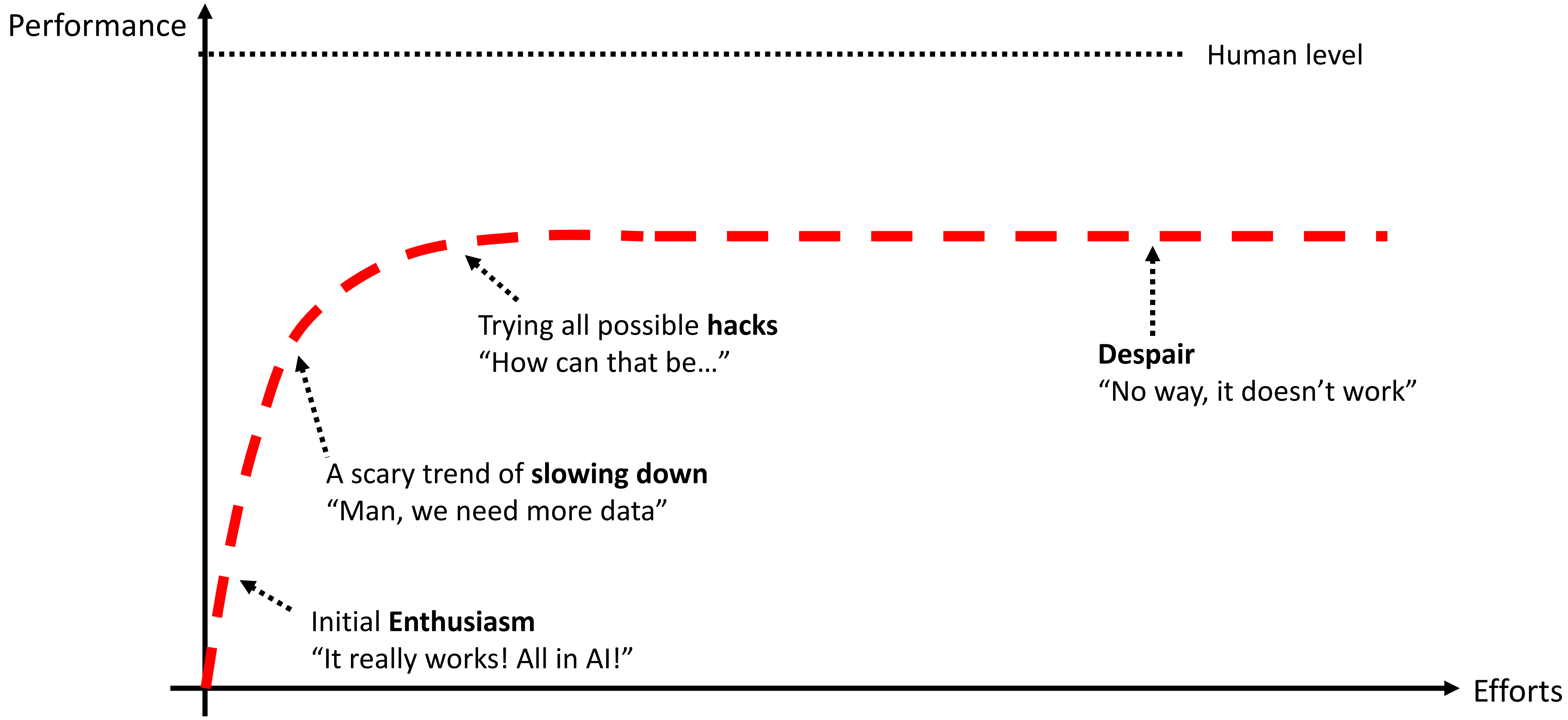
Exponential space to explore

Autonomous Driving

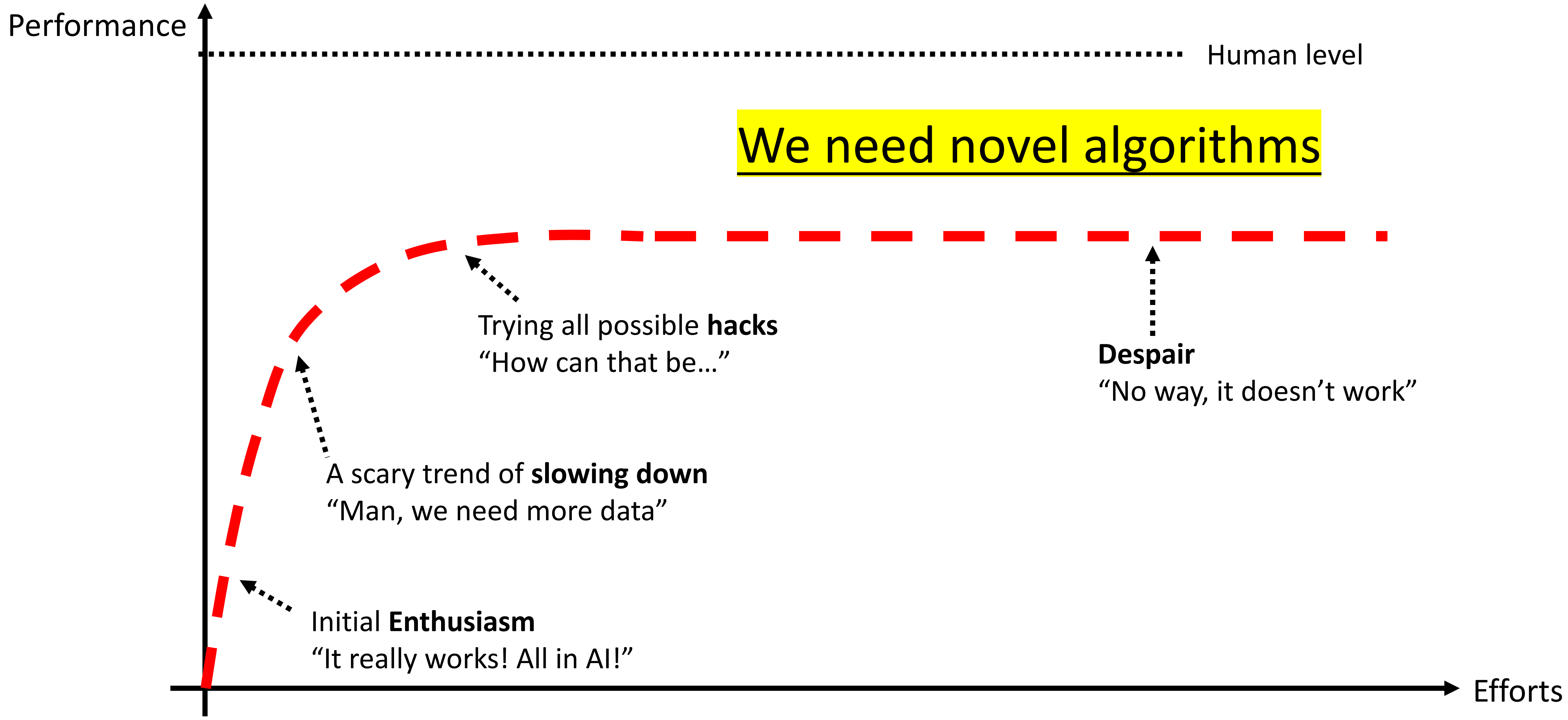
Home Robotics

Very few supervised data
Complicated environments
Lots of Corner cases.

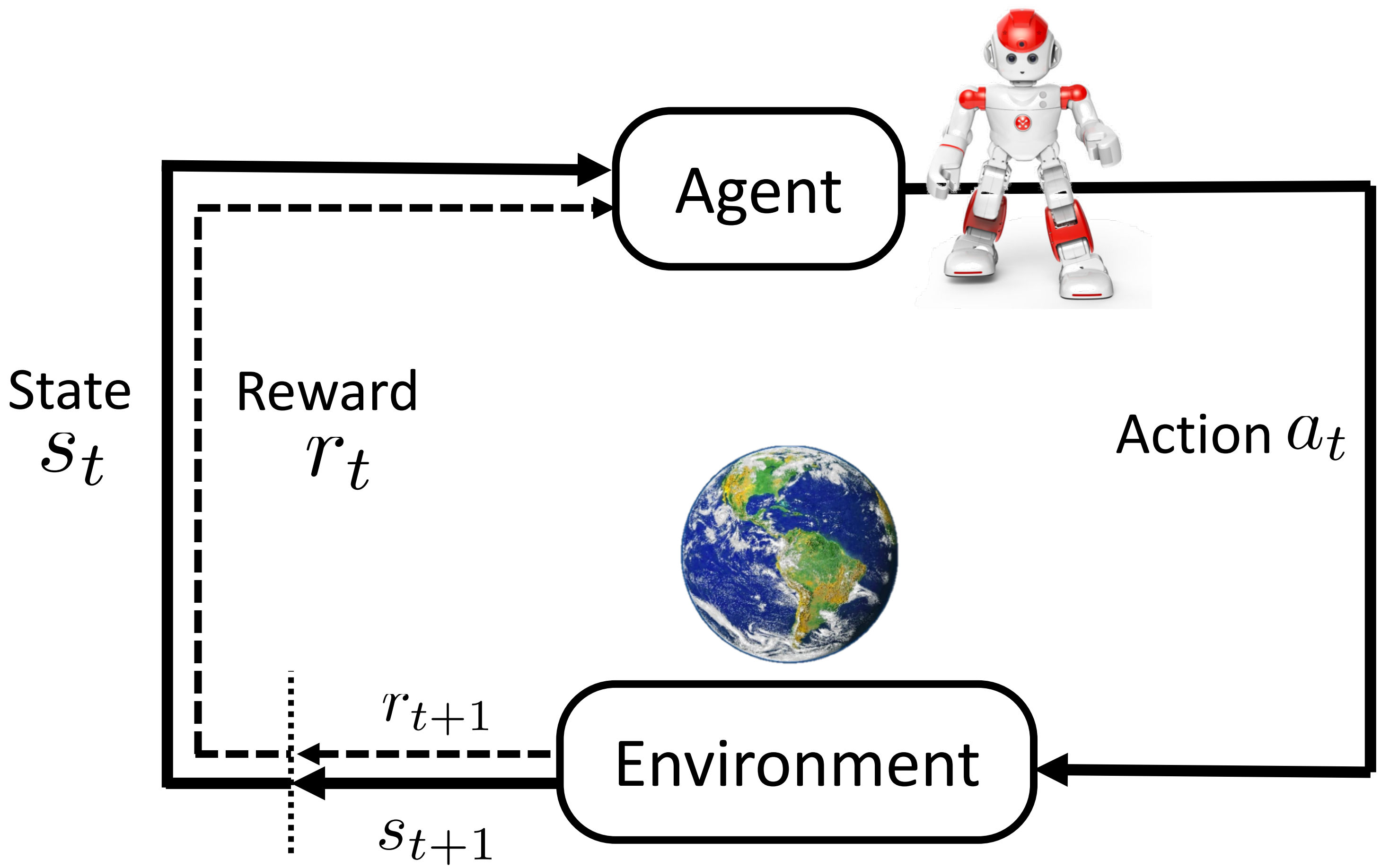
What AI still needs to improve

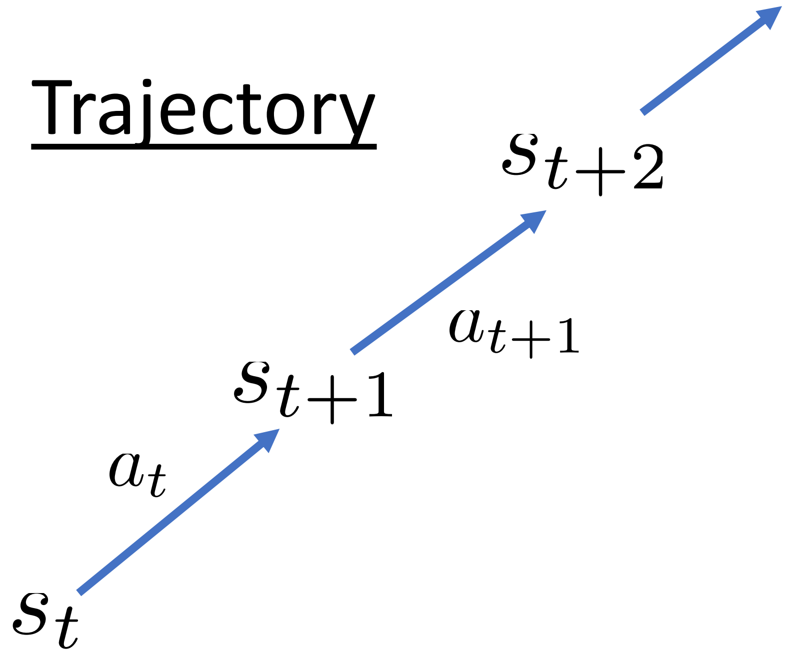


What AI still needs to improve



Reinforcement Learning

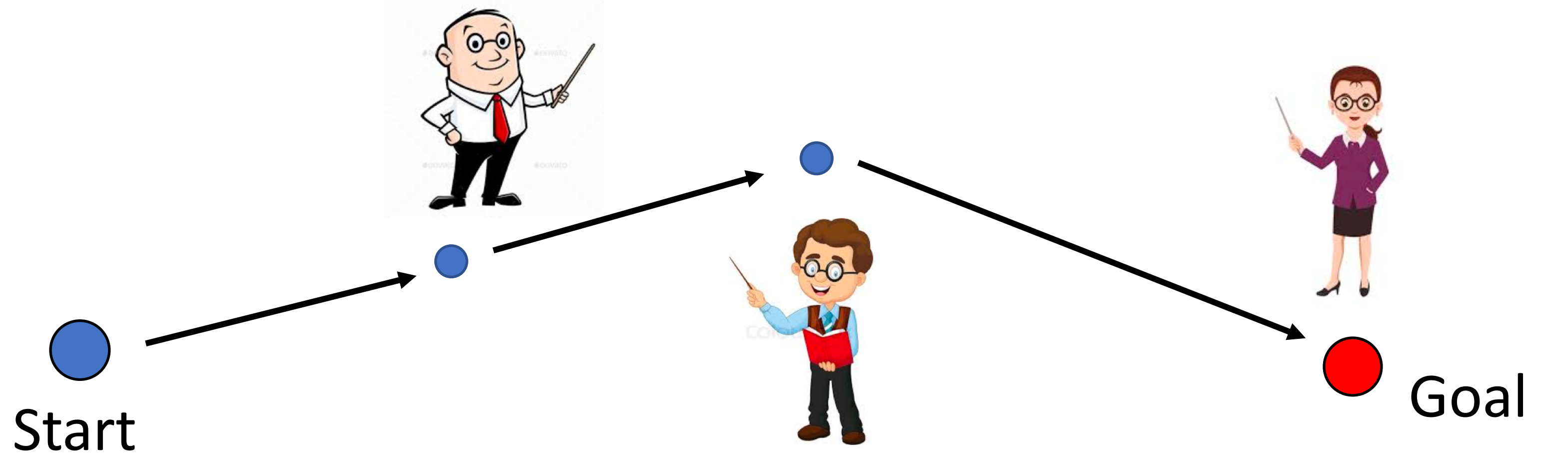




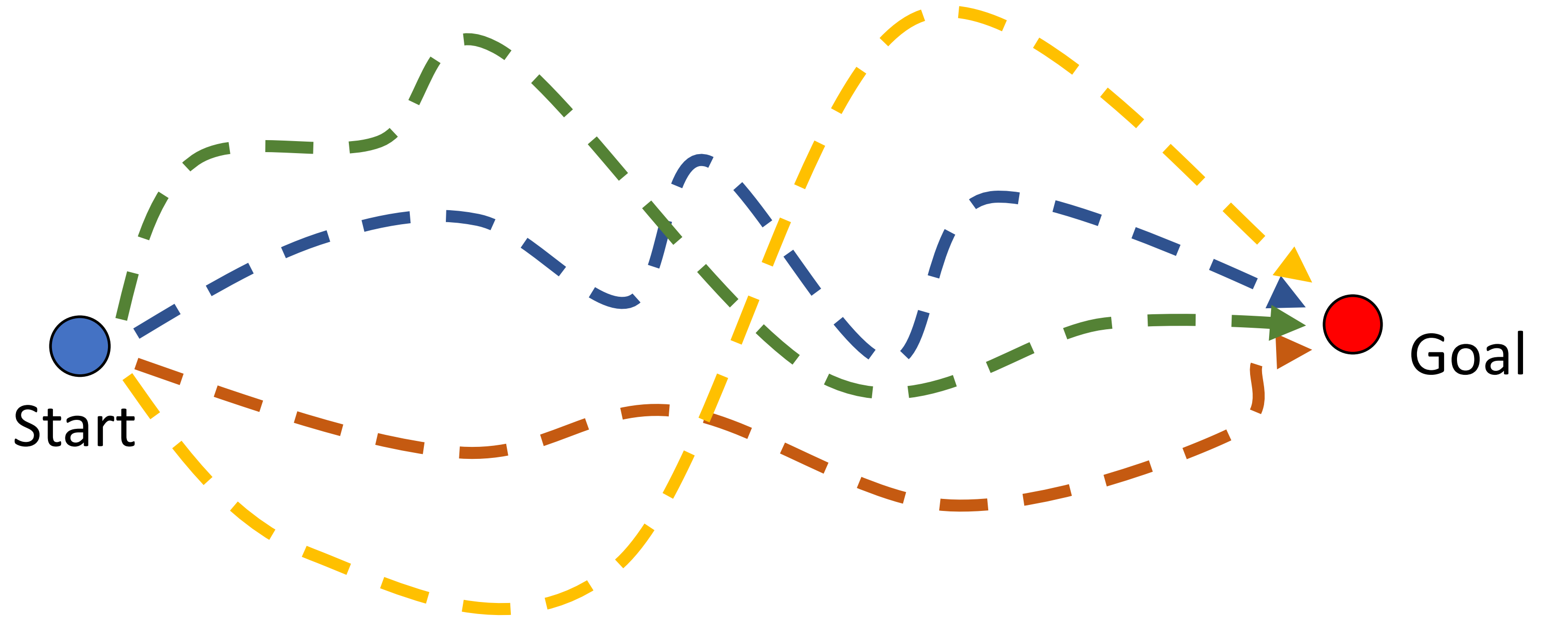
[R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction]

Supervised Learning v.s Reinforcement Learning

Supervised learning



Reinforcement learning



Supervised Learning v.s Reinforcement Learning

Supervised learning

The boss decides what you will learn

You work hard to get them right

Reinforcement learning

Explore the space to find a good solution

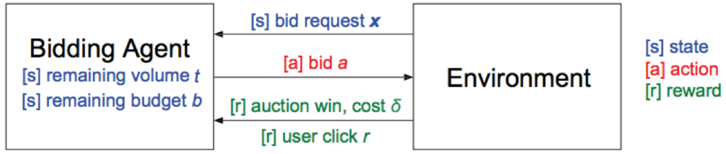
You decide what data you want to learn

More data hungry

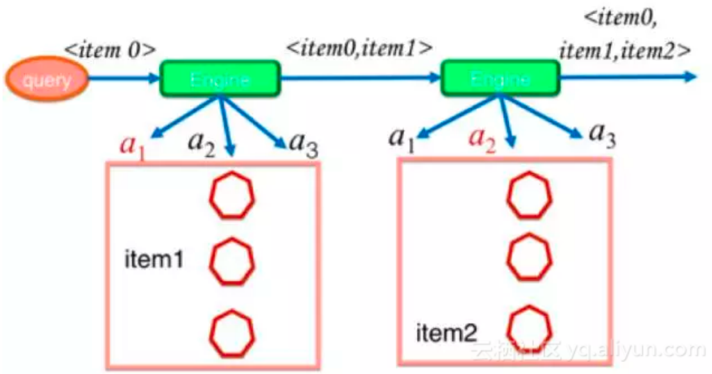
More computational resources

Applications of Reinforcement Learning

Sequential Decision Making



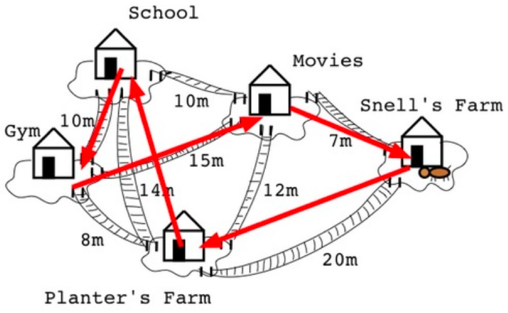
Real-time ads bidding



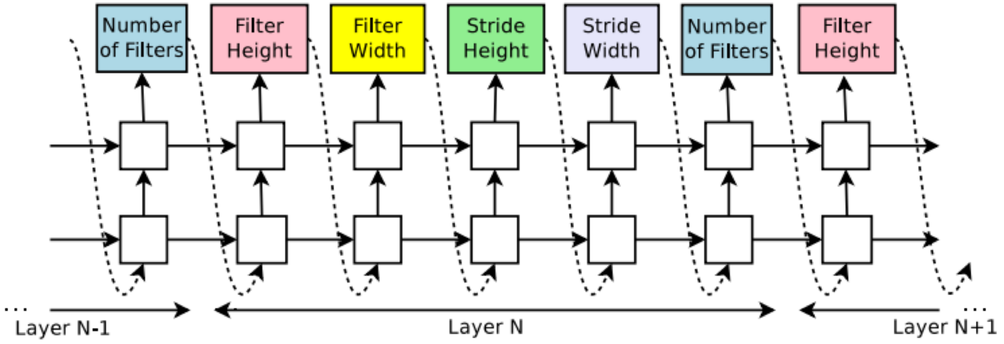
Recommendation

RL for Optimization

3D Bin-packing



Travel Salesman problem



Architecture Search

Game as a testbed of Reinforcement Learning

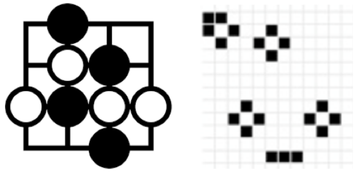
Infinite supply of *fully* labeled data

Controllable and replicable

Low cost per sample

Faster than real-time

Less safety and ethical concerns

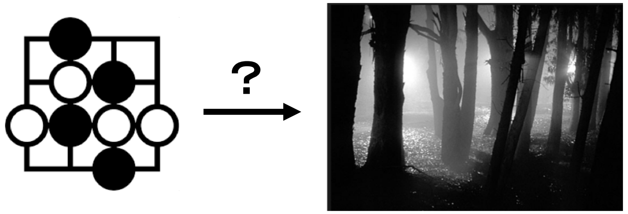


Complicated dynamics with simple rules.

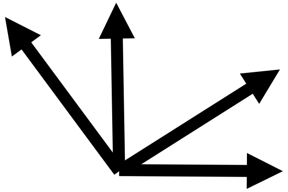
Game as a testbed of Reinforcement Learning

Algorithm is slow and data-inefficient

Require a lot of resources.



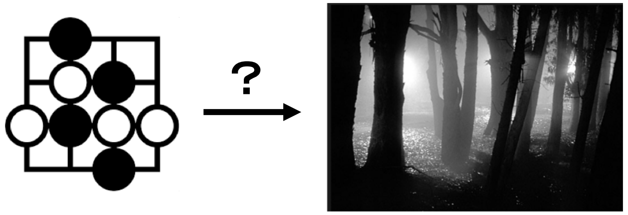
Abstract game to real-world



Hard to benchmark the progress

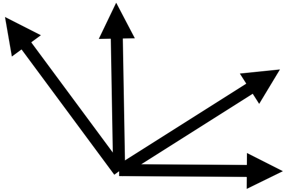
Game as a testbed of Reinforcement Learning

Algorithm is slow and data-inefficient



Abstract game to real-world

Require a lot of resources.



Hard to benchmark the progress

Better Algorithm/System

Better Environment

Our work

Better Algorithm/System

Go Engine
(*DarkForest*, Y. Tian, Y. Zhu, ICLR16)
(*ELF OpenGo*, Y. Tian et al.)

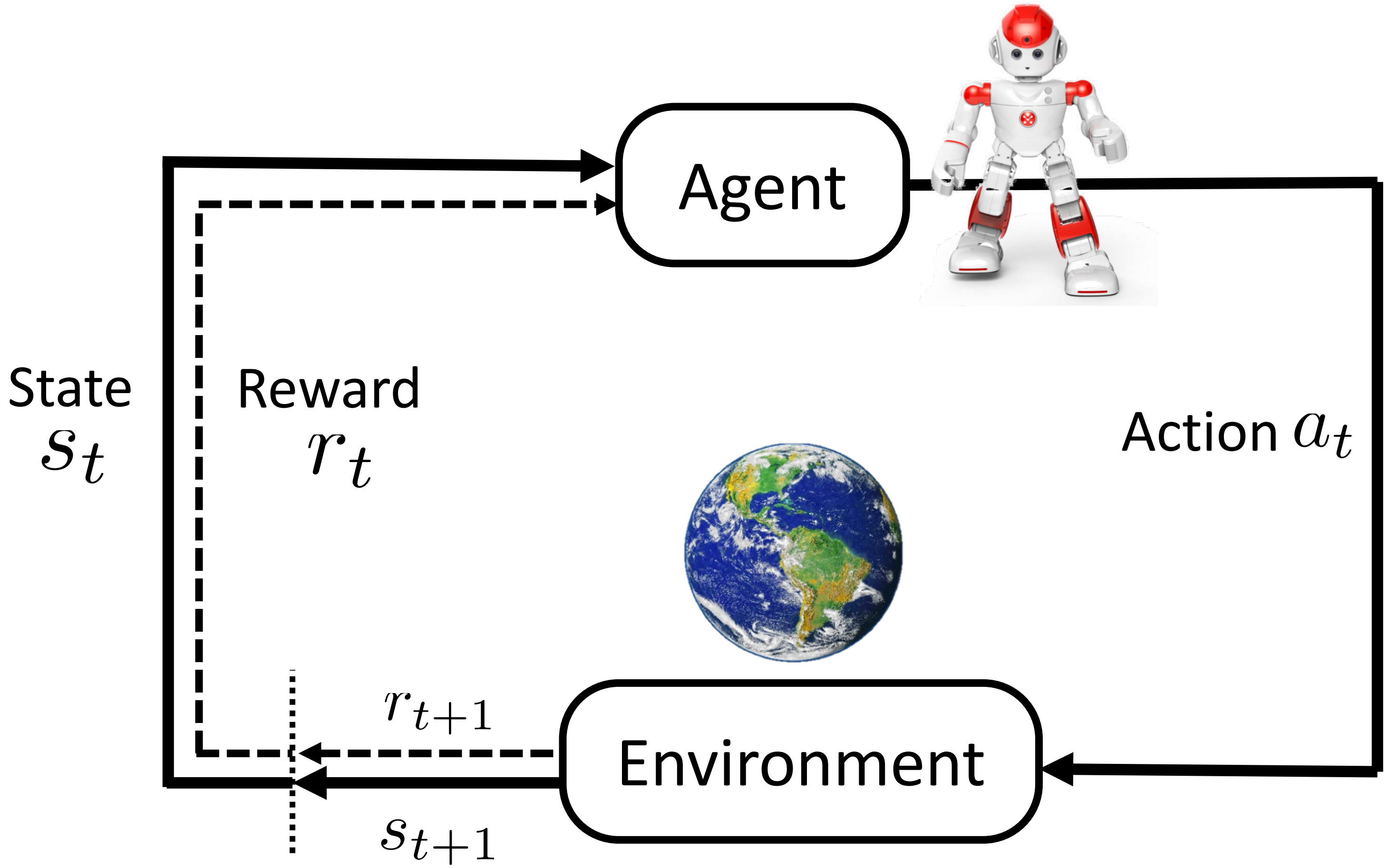
Doom AI
(Yuxin Wu, Yuandong Tian, ICLR17)

Better Environment

ELF: Extensive Lightweight and Flexible Framework
(Yuandong Tian et al, NIPS17)

House3D: An interactive 3D environment for navigation
(Yi Wu, Georgia Gkioxari, Yuxin Wu, Yuandong Tian)

A Framework for Deep Reinforcement Learning

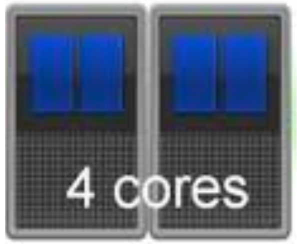


Design Choices:

CPU, GPU?
Simulation, Replays
Concurrency

[R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction]

A Framework for Deep Reinforcement Learning



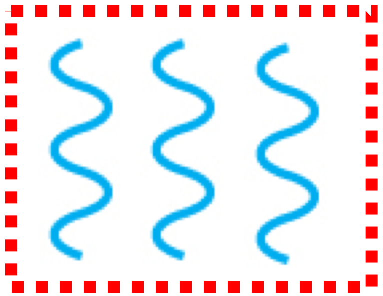
CPU

Game simulation

GPU

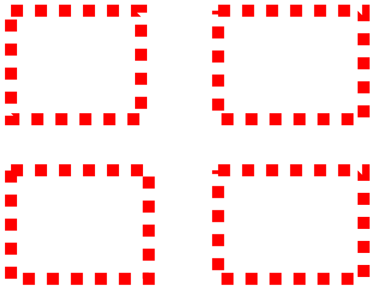
Neural network

Single thread

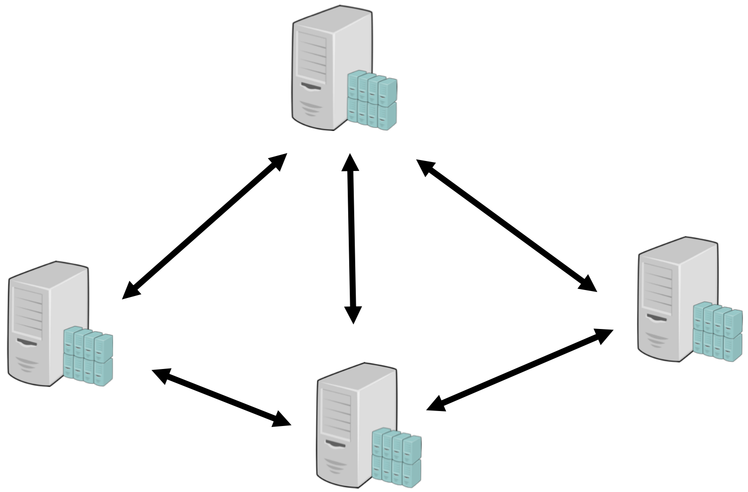


Single process,
multiple threads

Shared memory



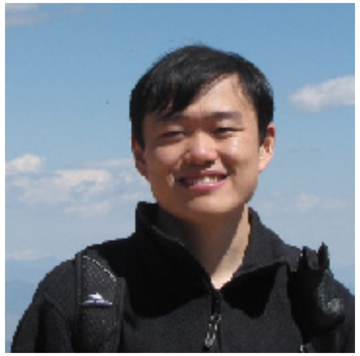
Multi-process



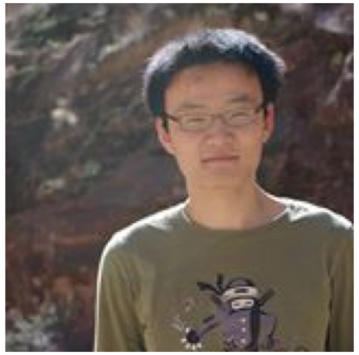
Distributed System

Scalable, high-latency, less robust

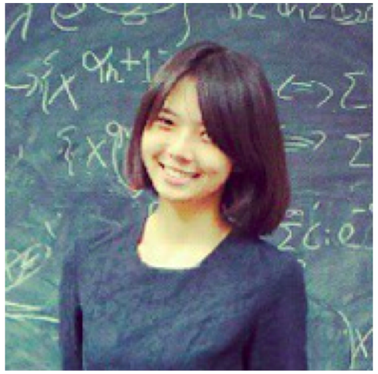
ELF: Extensive, Lightweight and Flexible Framework for Game Research



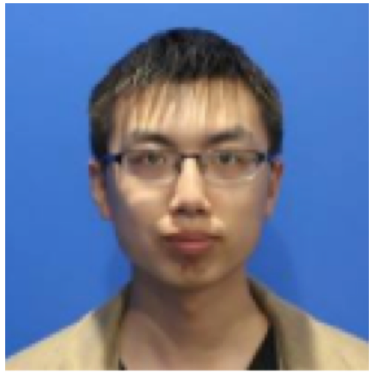
Yuandong Tian



Qucheng Gong



Wenling Shang



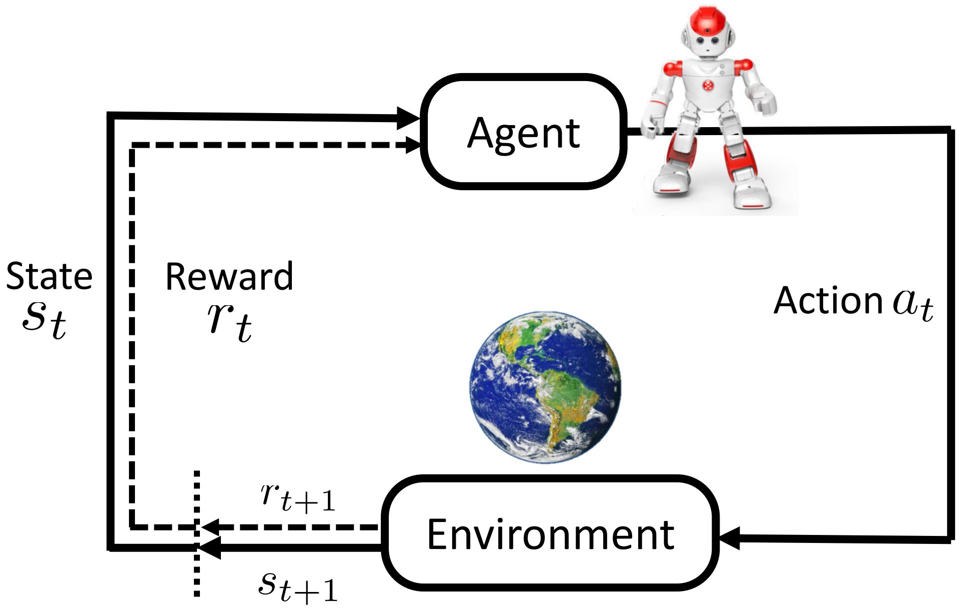
Yuxin Wu

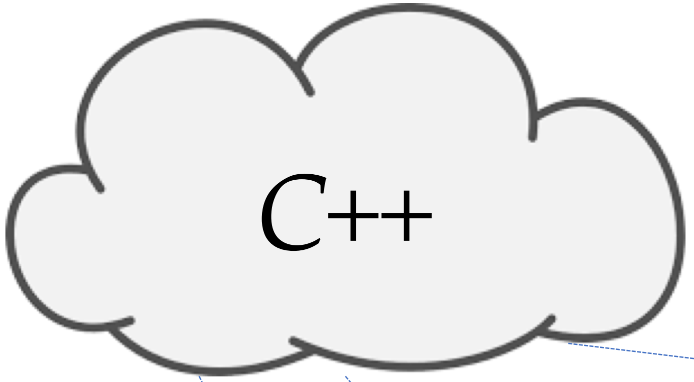
Larry Zitnick

https://github.com/facebookresearch/ELF

[Y. Tian et al, ELF: An Extensive, Lightweight and Flexible Research Platform for Real-time Strategy Games, NIPS 2017]

ELF: A simple for-loop

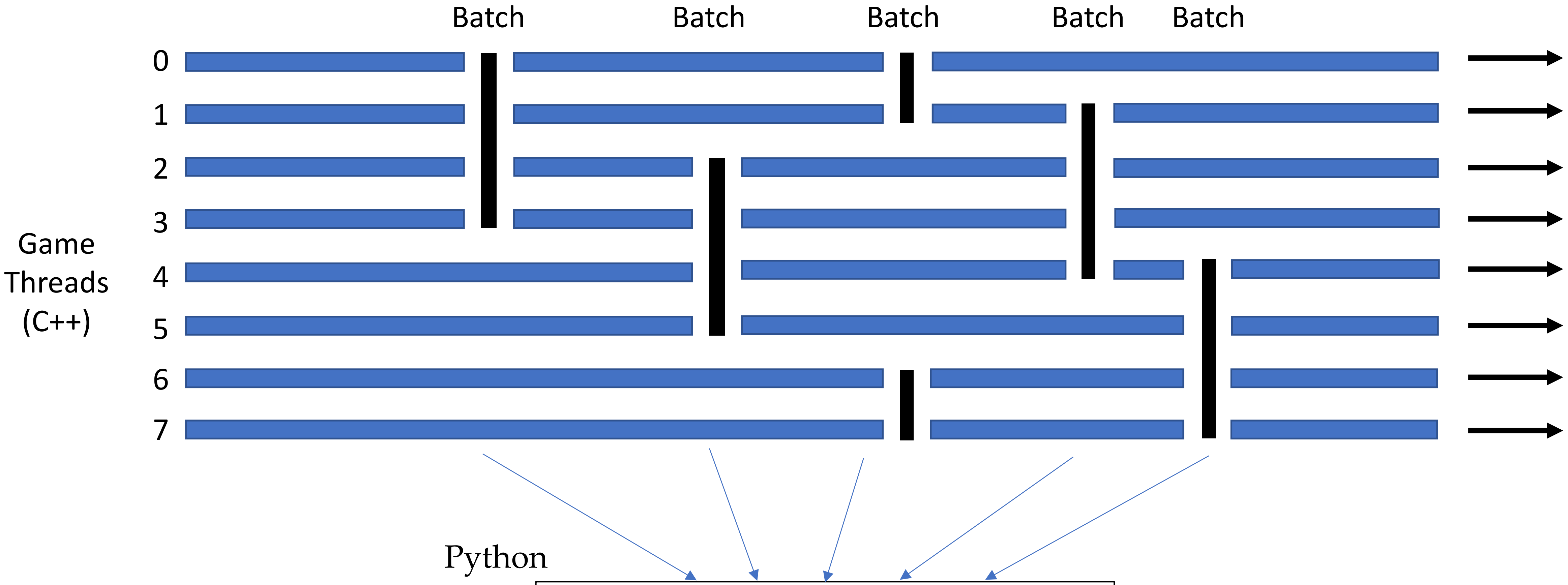




Python

```
while True:
    batched_states = GameContext.Wait()
    replies = model(batched_states)
    GameContext.Steps(replies)
```

How ELF works




```python
while True:
    batched_states = GameContext.Wait()
    replies = model(batched_states)
    GameContext.Steps(replies)
```

ELF Characteristics

Extensive

Any games with
C++ interfaces
can be incorporated.

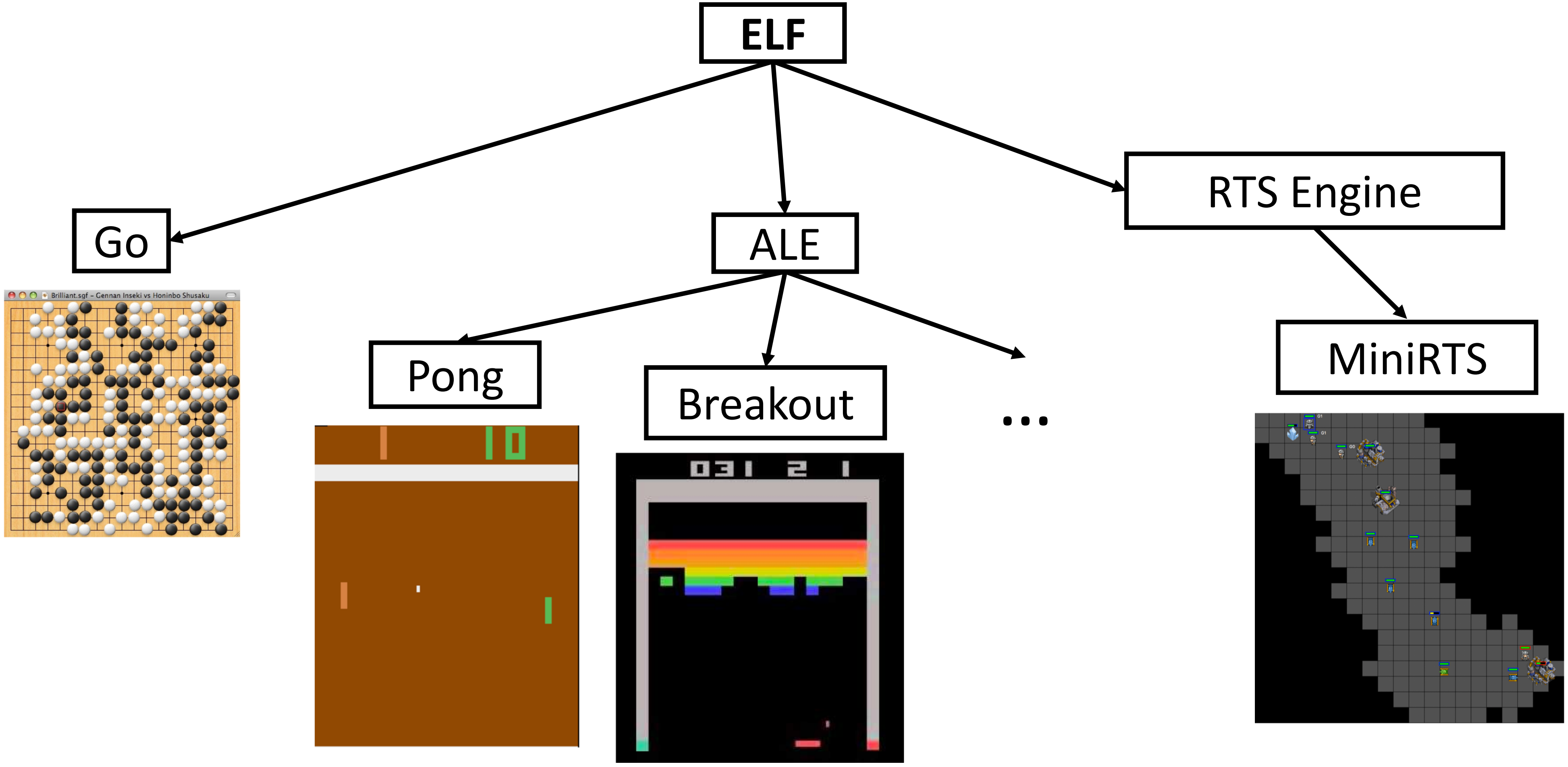
Flexible

Environment-Actor topology
Parametrized game environments.
Choice of different RL methods.

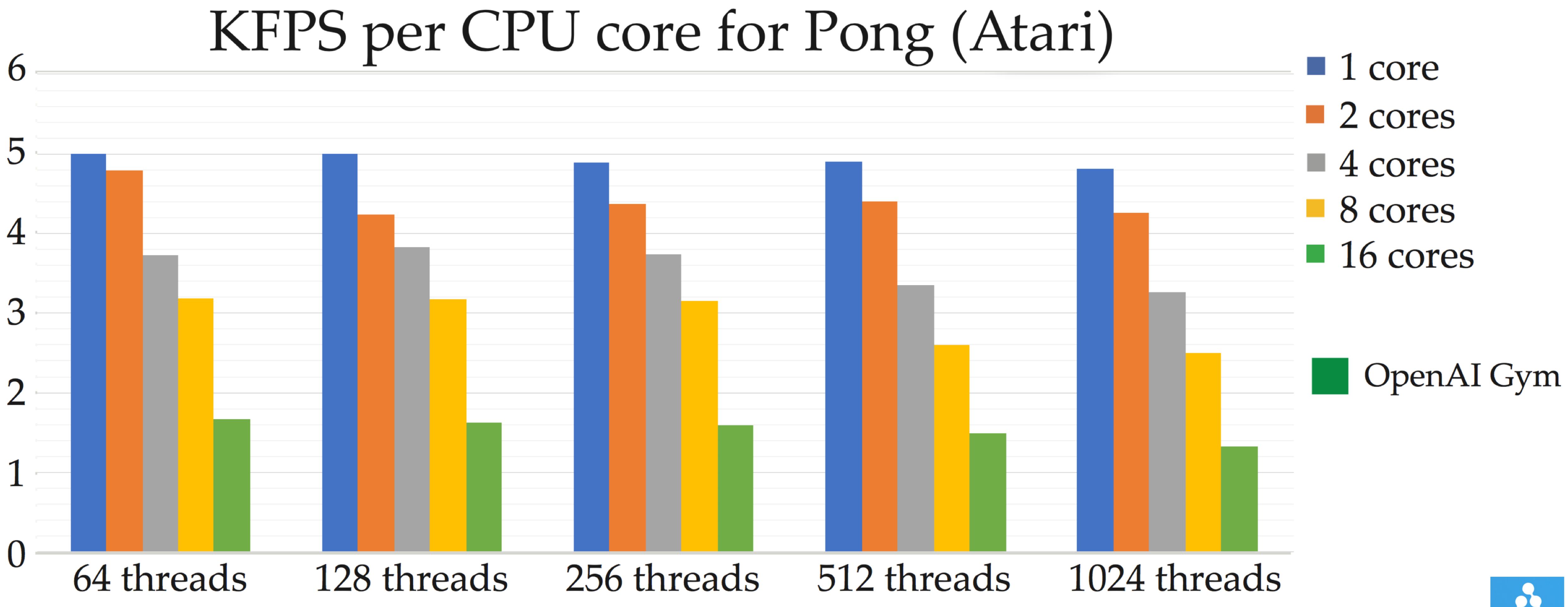
Lightweight

Fast. Mini-RTS (40K FPS per core)
Minimal resource usage (1GPU+several CPUs)
Fast training (half a day for a RTS game)

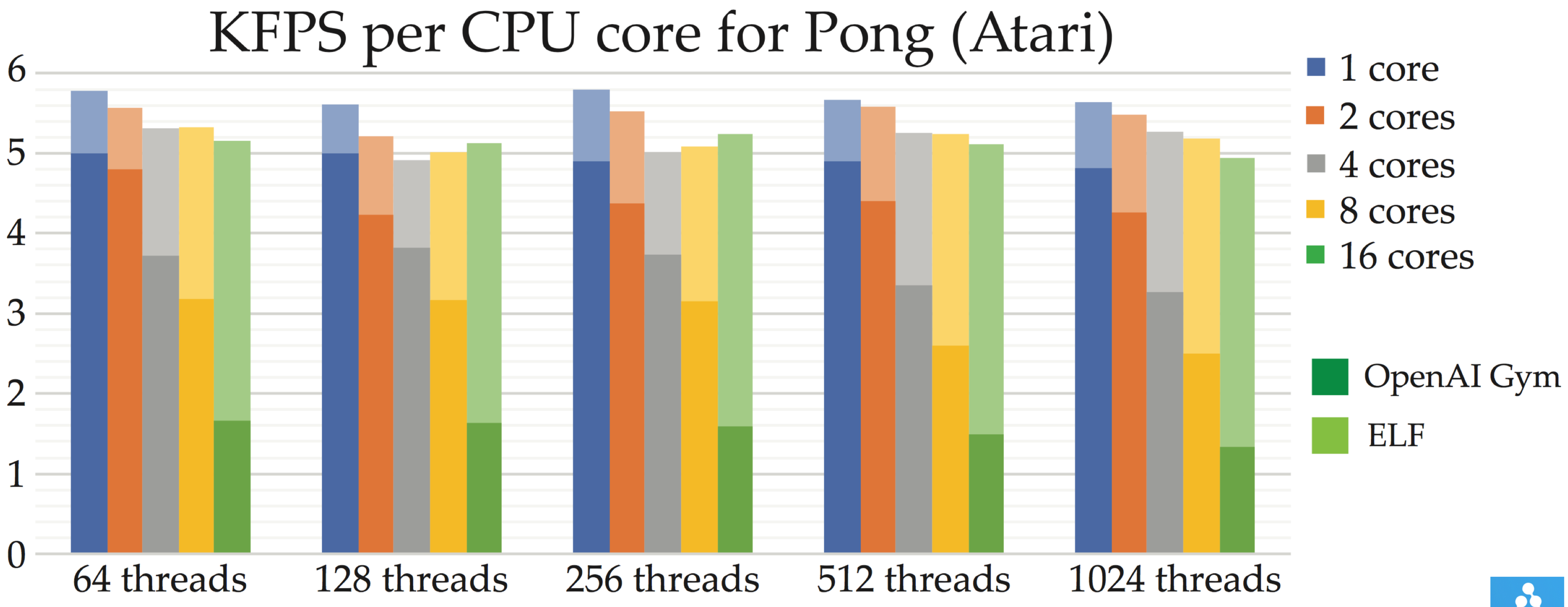
Extensibility



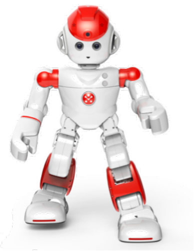
Lightweight

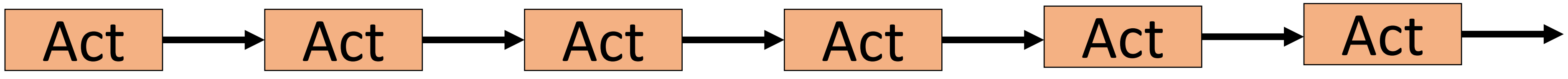


Lightweight



Flexibility

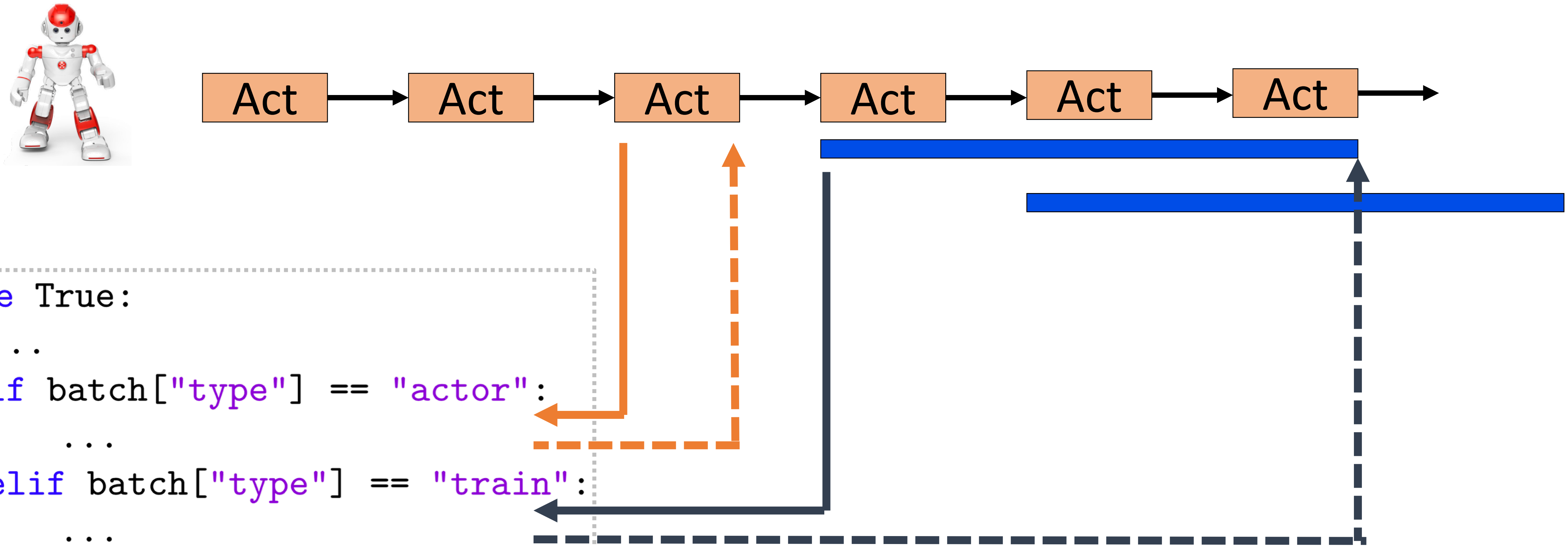





```
while True:
    batched = GameContext.Wait()
    replies = model(batched)
    GameContext.Steps(replies)
```

Evaluation

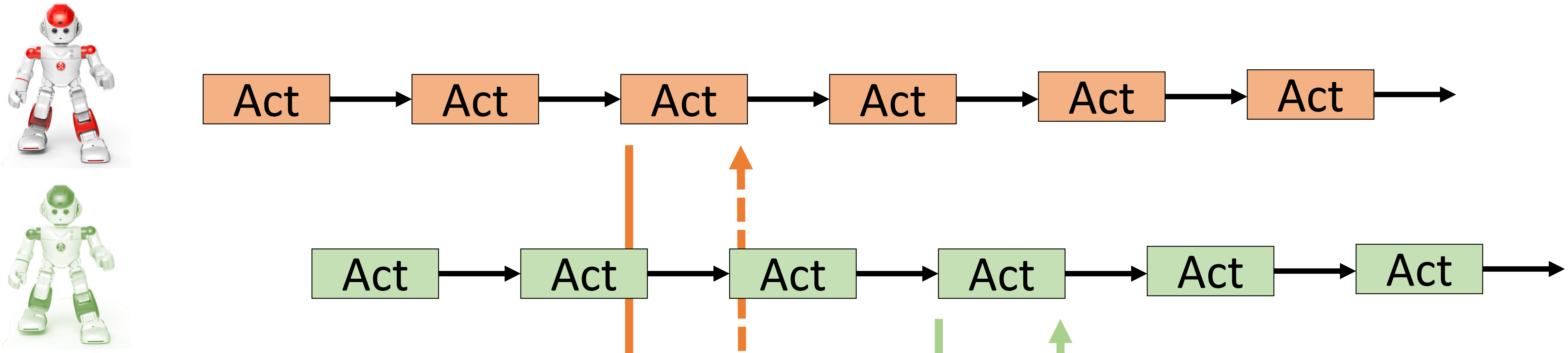
Flexibility




```python
while True:
    ...
    if batch["type"] == "actor":
        ...
    elif batch["type"] == "train":
        ...
```

Training

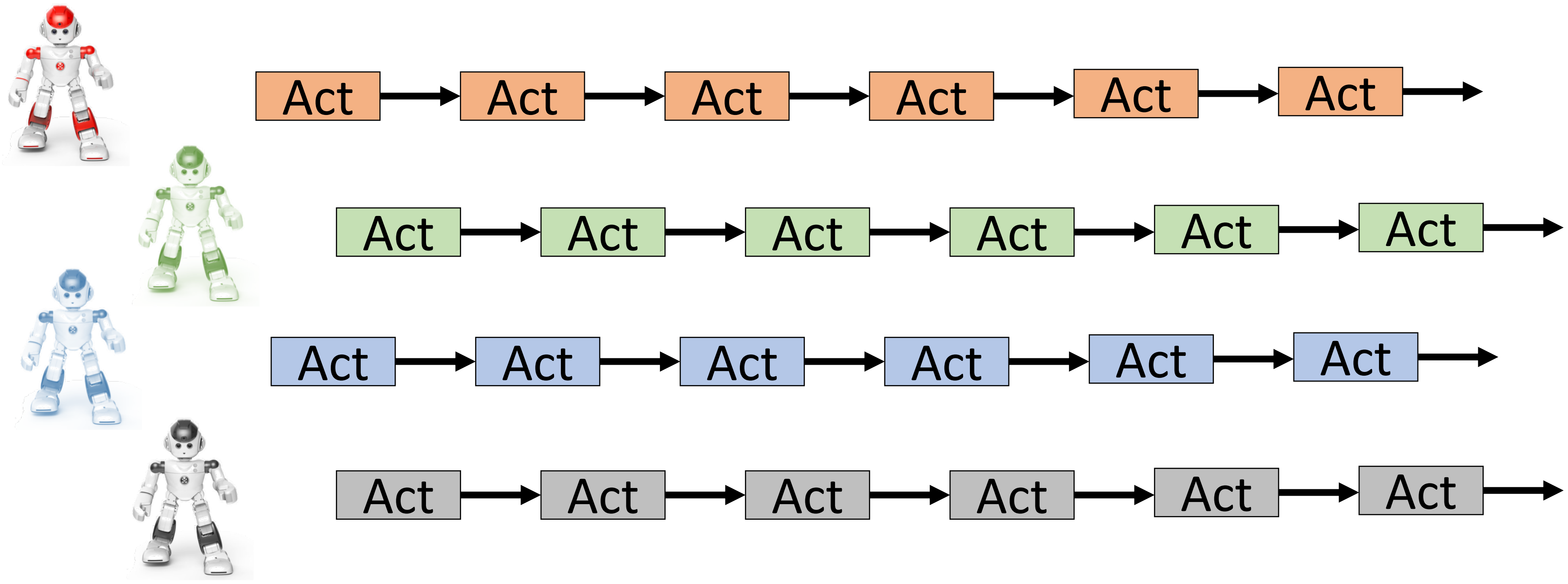
Flexibility




```python
while True:
    ...
    if batch["type"] == "actor0":
        ...
    elif batch["type"] == "actor1":
        ...
```

Self-play

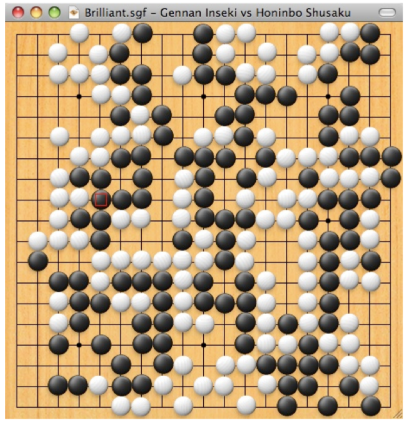
Flexibility



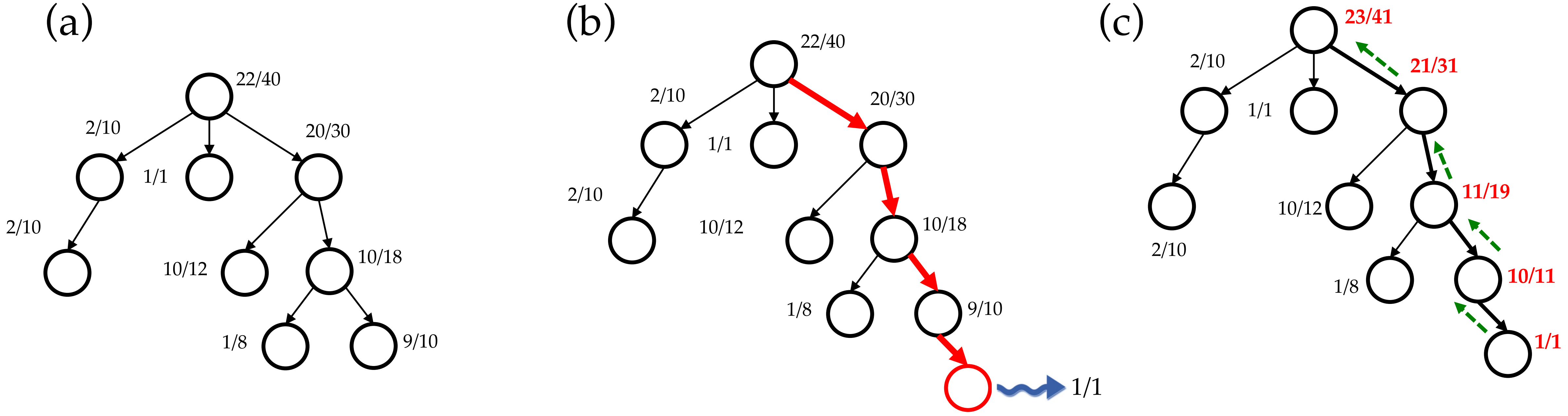

```
while True:
    ...
    for i in range(n):
        if batch["type"] == "actor%d" % i:
            ...
```

Multi-agent

Monte Carlo Tree Search



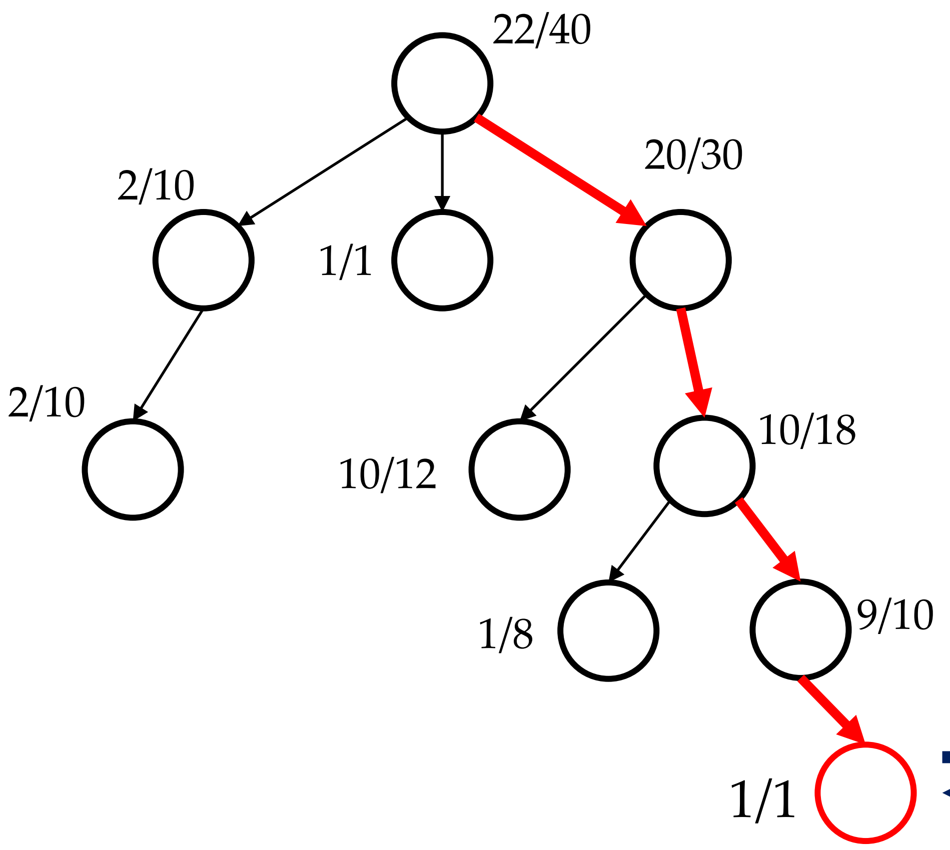
Aggregate win rates, and search towards the good nodes.



Tree policy
Default policy

$$a_t = \operatorname*{argmax}_a (Q(s_t, a) + u(s_t, a)) \qquad u(s,a) \propto \frac{P(s,a)}{1 + N(s,a)}$$ PUCT

Flexibility




```python
while True:
    batched = GameContext.Wait()
    replies = model(batched)
    GameContext.Steps(replies)
```

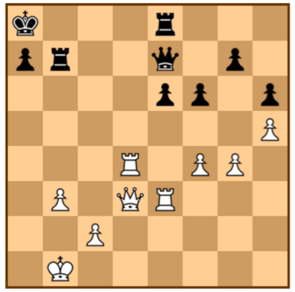
Monte-Carlo Tree Search

How Game AI works

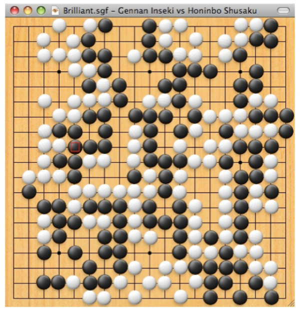
Even with a super-super computer,
it is not possible to search the entire space.

How Game AI works

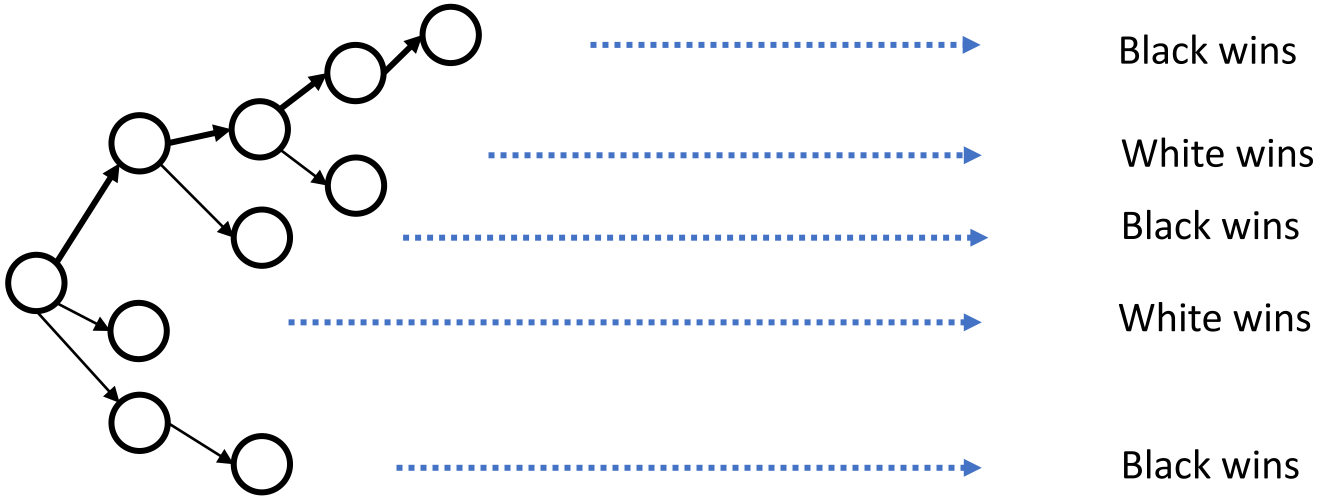
Even with a super-super computer,
it is not possible to search the entire space.



Lufei Ruan vs. Yifan Hou (2010)



Current game situation

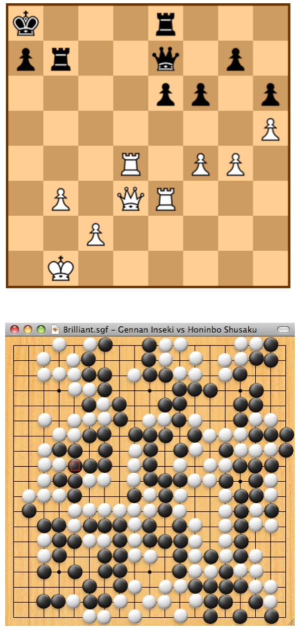


Extensive search ***Evaluate*** ***Consequence***

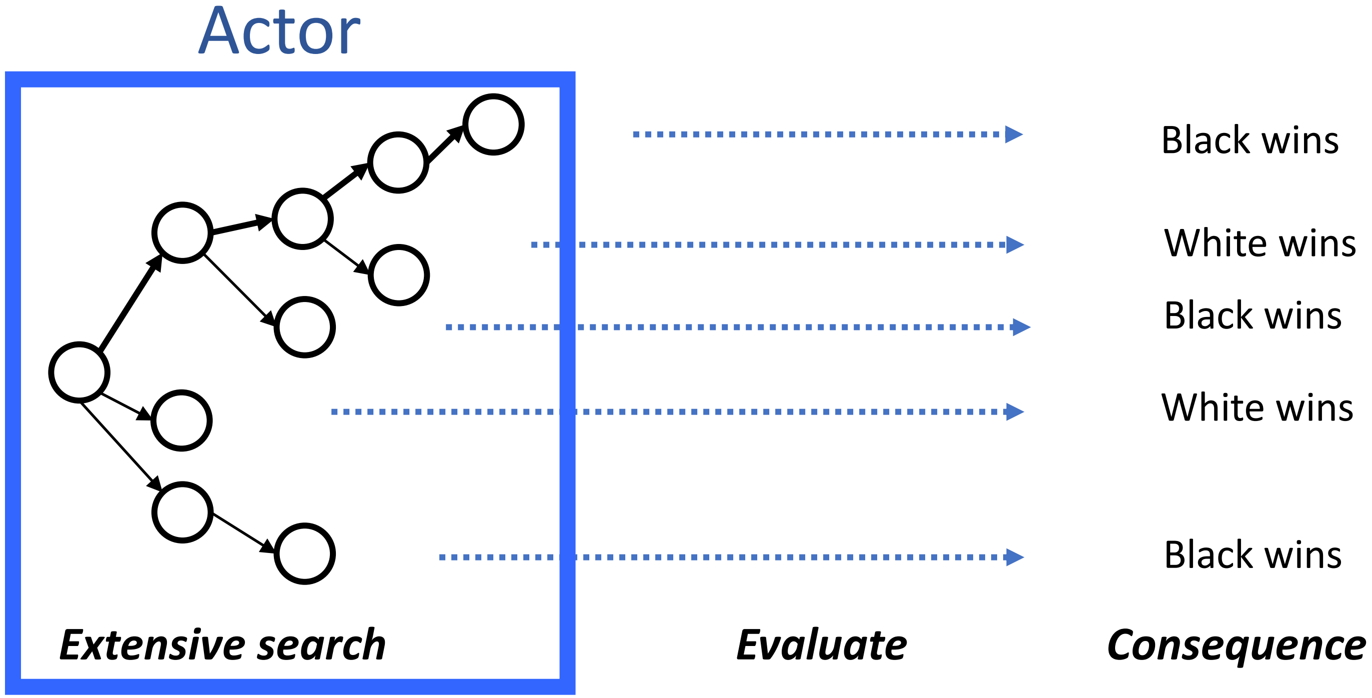
How Game AI works

How many action do you have per step?

Checker: a few possible moves → Alpha-beta pruning + Iterative deepening [Major Chess engine]
Poker: a few possible moves → Counterfactual Regret Minimization [Libratus, DeepStack]
Chess: 30-40 possible moves → Alpha-beta pruning + Iterative deepening [Major Chess engine]
Go: 100-200 possible moves → Monte-Carlo Tree Search + UCB exploration [Major Go engine]
StarCraft: 50^100 possible moves → ???



Current game situation

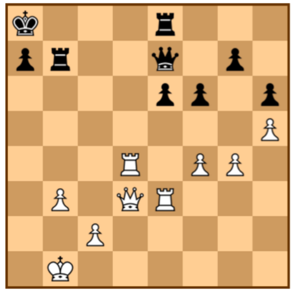


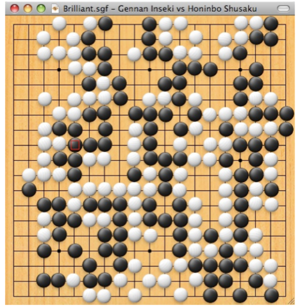
How Game AI works

How complicated is the game situation? How deep is the game?

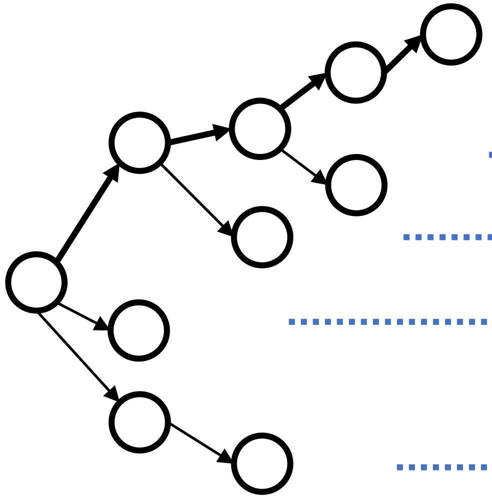
Chess
Go
Poker
StarCraft

- Rule-based
- Linear function for situation evaluation [Stockfish]
- End game database
- Random game play with simple rules [Zen, CrazyStone, DarkForest]
- Deep Value network [AlphaGo, DeepStack]





Current game situation



Extensive search

Critic

Black wins
White wins
Black wins
White wins
Black wins

Evaluate ***Consequence***

How to model Policy/Value function?

Non-smooth + high-dimensional
Sensitive to situations. One stone changes in Go leads to different game.

Traditional approach

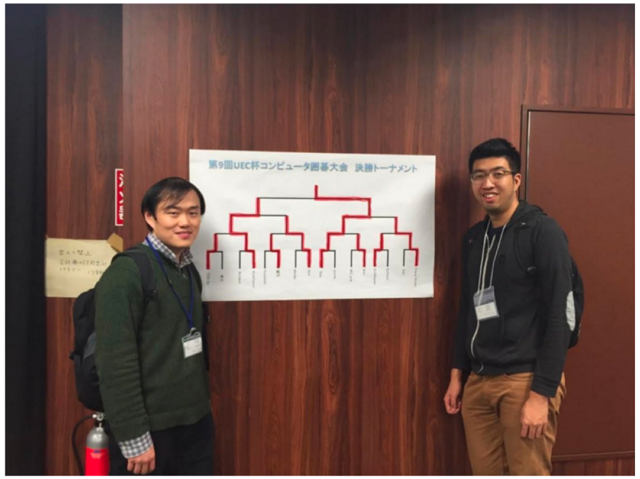
- Many manual steps
- Conflicting parameters, not scalable.
- Need strong domain knowledge.

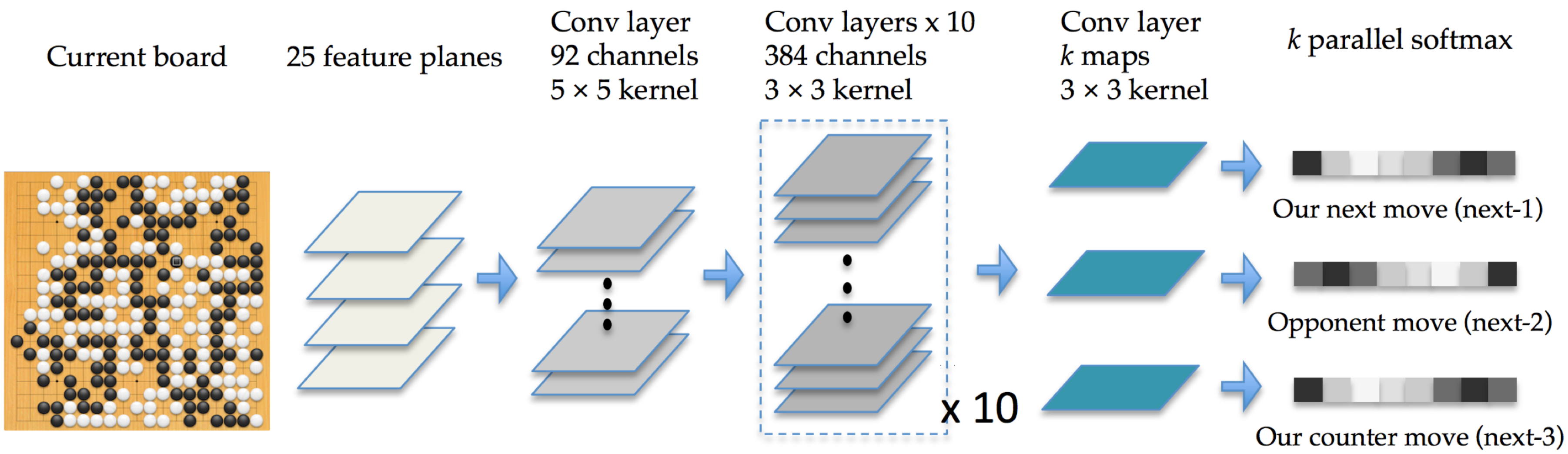
Deep Learning

- End-to-End training
    - Lots of data, less tuning.
- Minimal domain knowledge.
- Amazing performance

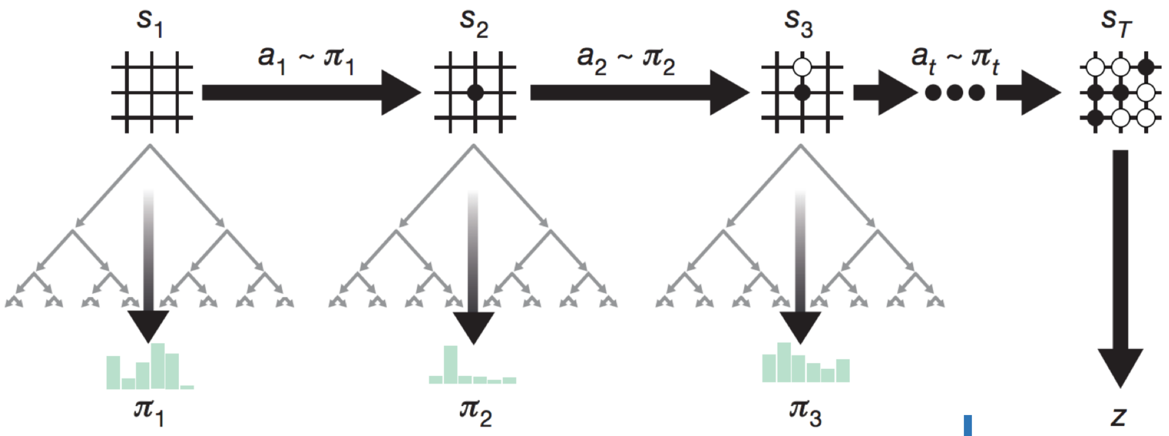
Our computer Go player: DarkForest

- Top ameuter level
- Release 3 month before AlphaGo, < 1% GPUs





Reimplementation of AlphaGo Zero



Generate Training data

Zero-human knowledge

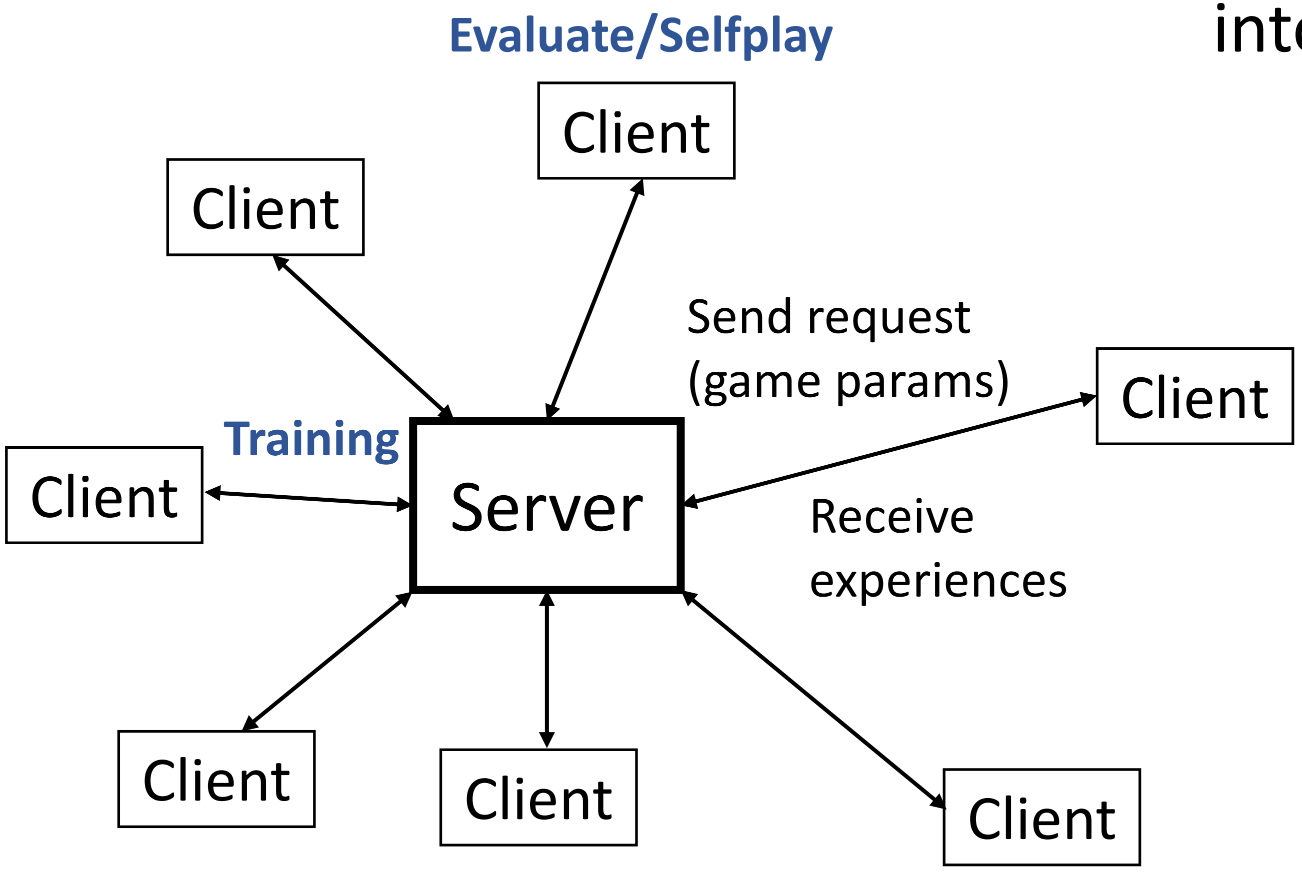
θ_i ← Update Models ← Self-Replays

[Silver et al, Mastering the game of Go without human knowledge, Nature 2017]

Demystifying AlphaGoZero/AlphaZero

- Amazing performance but no code available.
  - Huge computational cost (15.5 years to generate 4.9M selfplays with 1 GPU)
  - Sophisticated (distributed) systems.
- Lack of ablation analysis
  - What factor is critical for the performance?
  - Is the algorithm robust to random initialization and changes of hyper parameters?
  - How the ladder issue is solved?
- Lots of mysteries
  - Is the proposed algorithm really universal?
  - Is the bot almighty? Is there any weakness in the trained bot?

Distributed ELF



Putting AlphaGoZero and AlphaZero into the same framework

AlphaGoZero (more synchronization)
AlphaZero (less synchronization)

ELF OpenGo

- System can be trained with 2000 GPUs in 2 weeks.
- Decent performance against professional players and strong bots.
- Abundant ablation analysis
- Decoupled design, code highly reusable for other games.

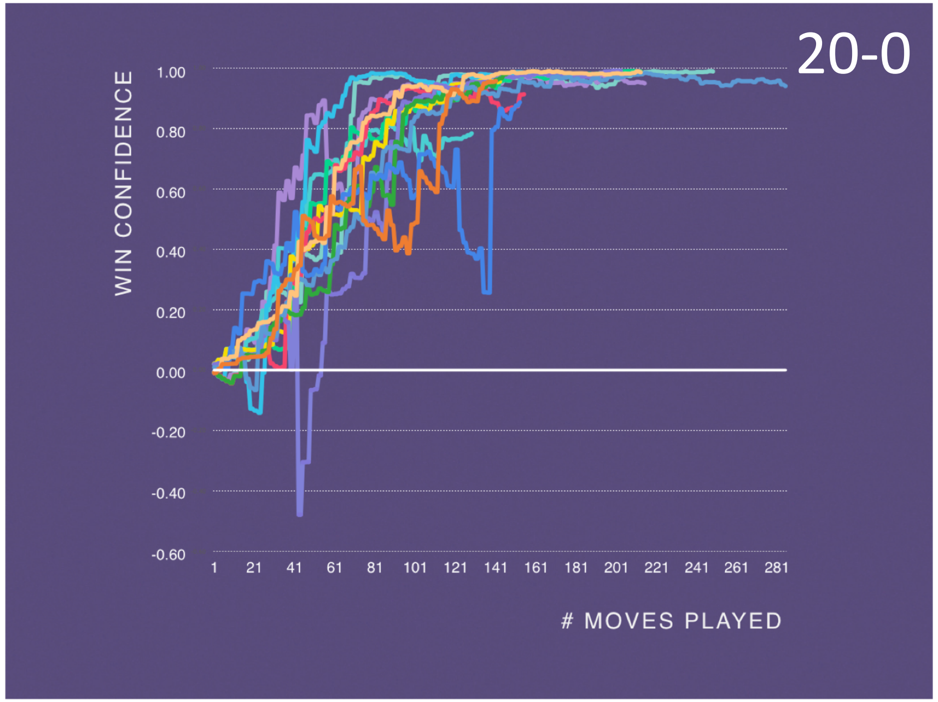
We open source the code and the pre-trained model for the Go and ML community

http://github.com/pytorch/elf

Performance

Vs top professional players

Name (rank)	ELO (world rank)	Result
Kim Ji-seok	3590 (#3)	5-0
Shin Jin-seo	3570 (#5)	5-0
Park Yeonghun	3481 (#23)	5-0
Choi Cheolhan	3466 (#30)	5-0

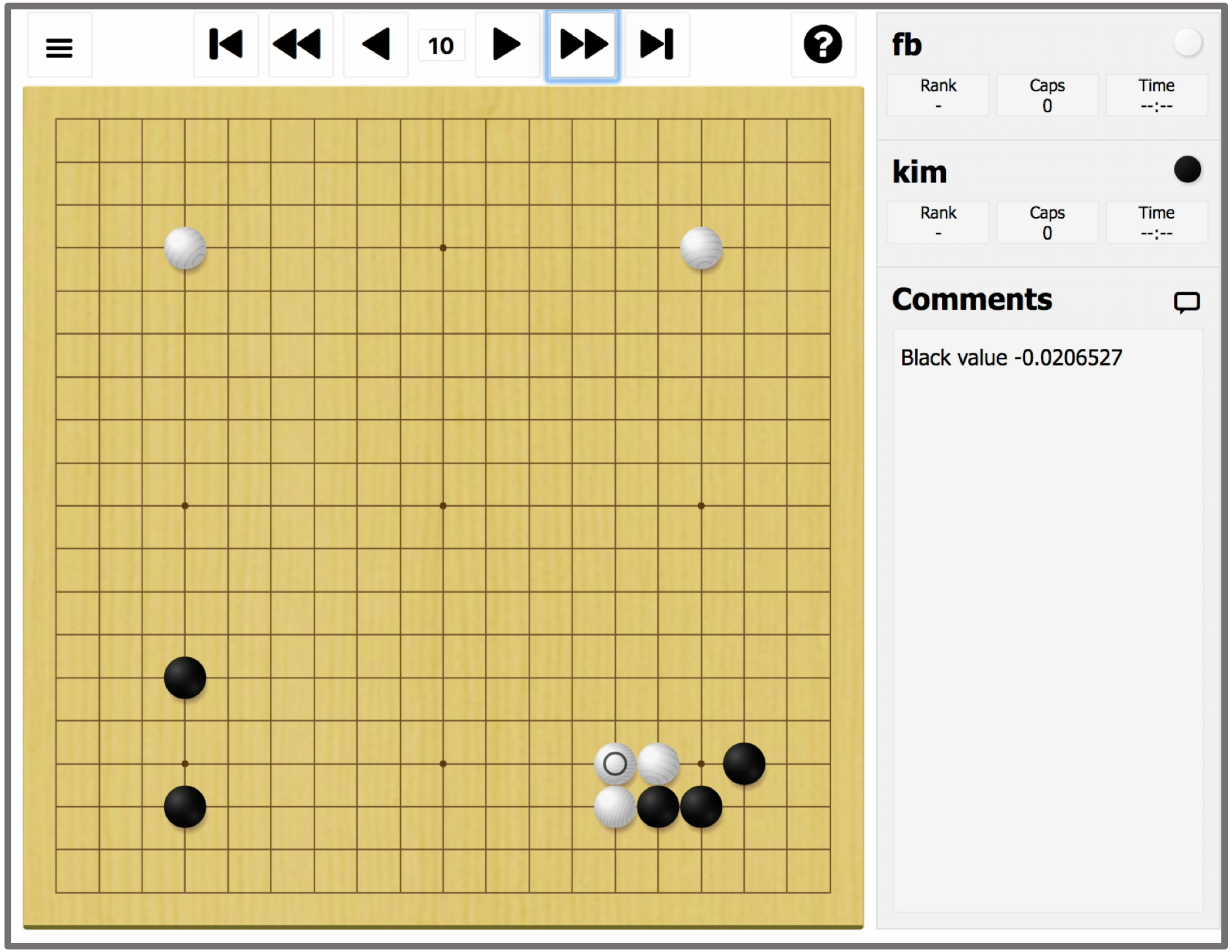


Single GPU, 80k rollouts, 50 seconds
Offer unlimited thinking time for the players

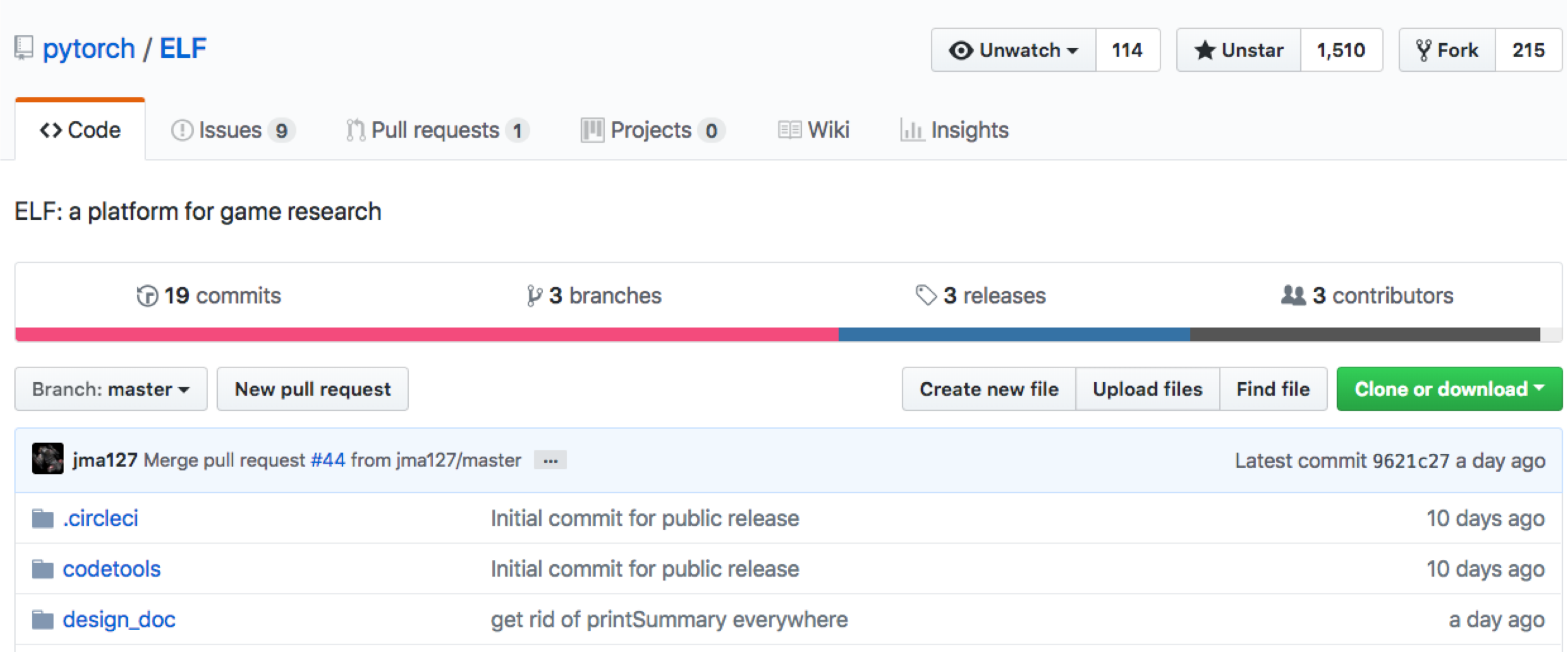
Vs strong bot (LeelaZero)

[158603eb, 192x15, Apr. 25, 2018]: 980 wins, 18 losses (98.2%)

Sample games versus Kim Jiseok (world #3)

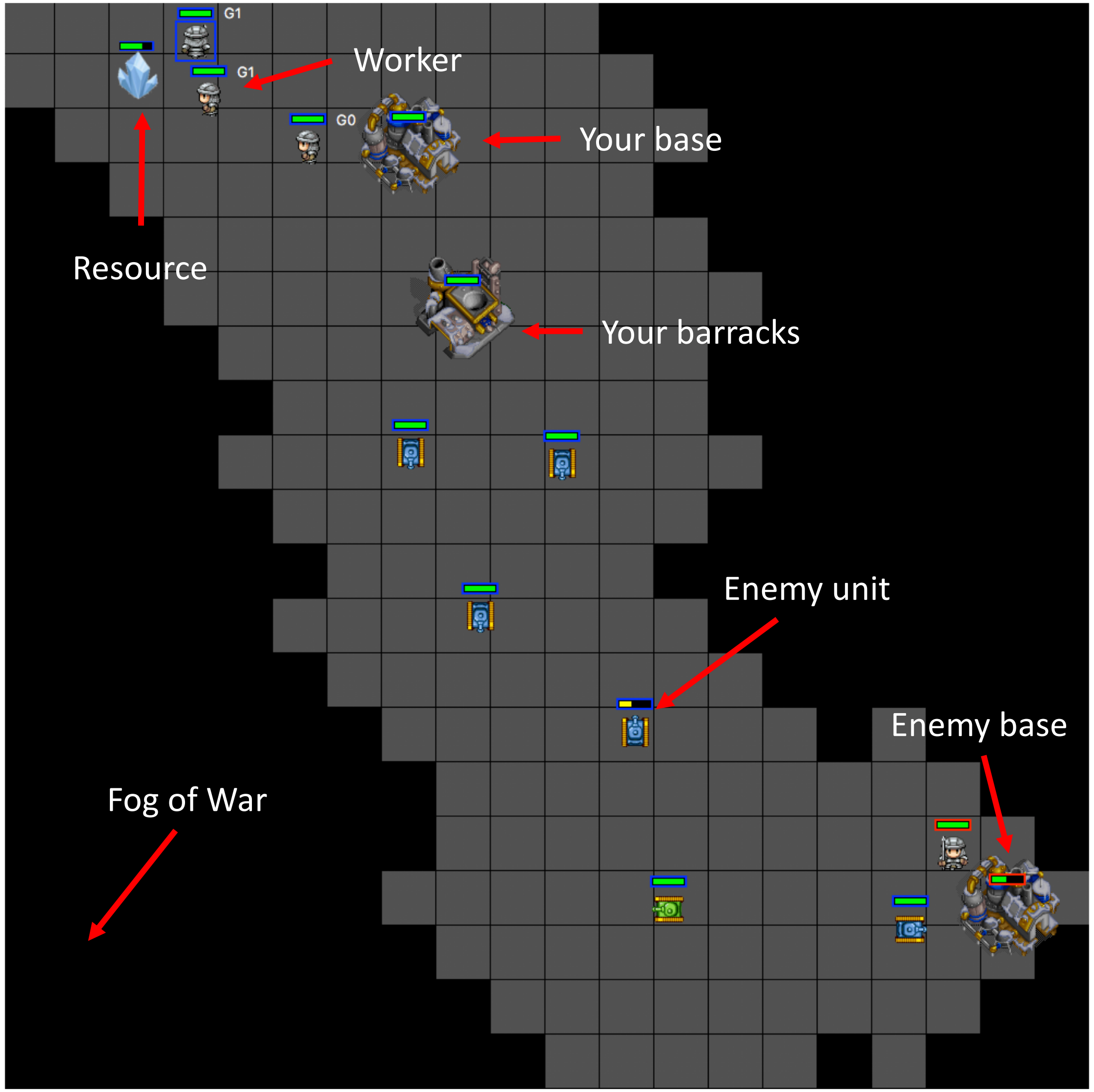


Open Source



https://github.com/pytorch/ELF

MiniRTS: A miniature RTS engine



Platform	**Frame per second**
ALE	6,000
Open AI Universe	60
Malmo	120
DeepMind Lab	287*/866**
VizDoom	7,000
TorchCraft	2,000
MiniRTS	40,000

* Using CPU only   ** Using CPUs and GPU

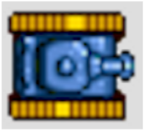
MiniRTS

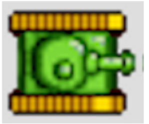
Base  Build workers and collect resources.

Resource  Contains 1000 minerals.

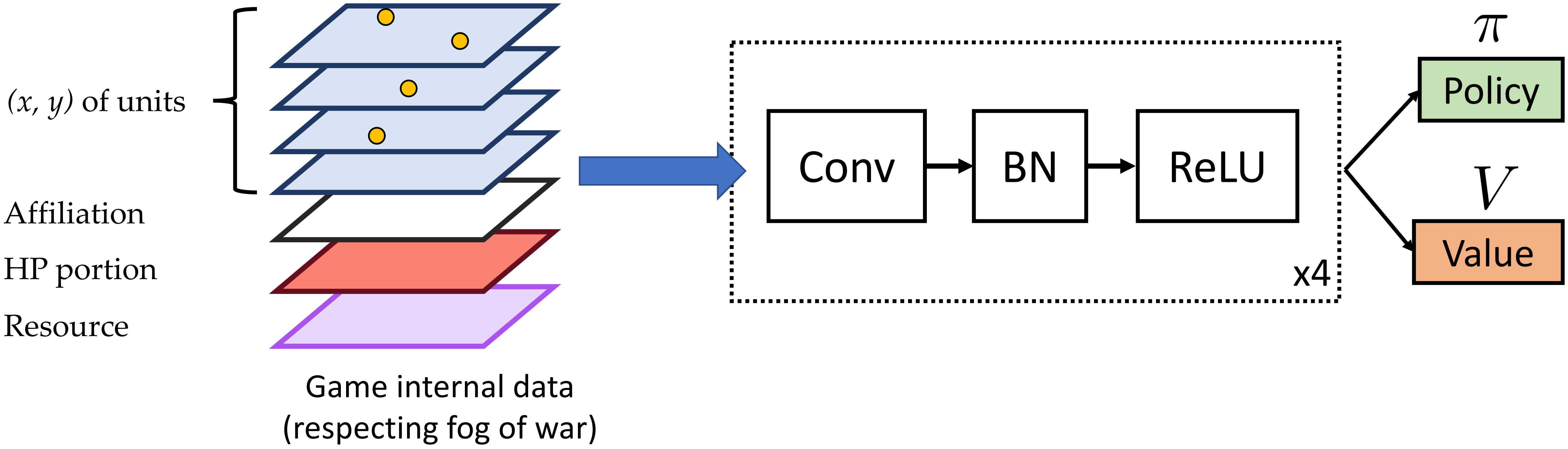
Barracks  Build melee attacker and range attacker.

Worker  Build barracks and gather resource.
Low speed in movement and low attack damage.

Melee Tank  High HP, medium movement speed, short attack range, high attack damage.

Range Tank  Low HP, high movement speed, long attack range and medium attack damage.

Training AI

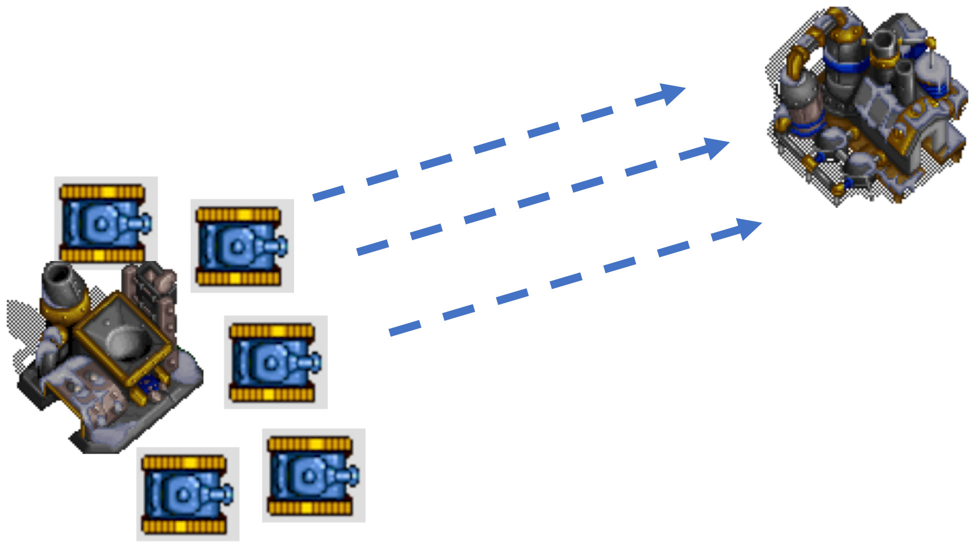


Using Internal Game data and Off-policy Actor-Critic Methods.
Reward is only available once the game is over.

9 Discrete Strategic Actions

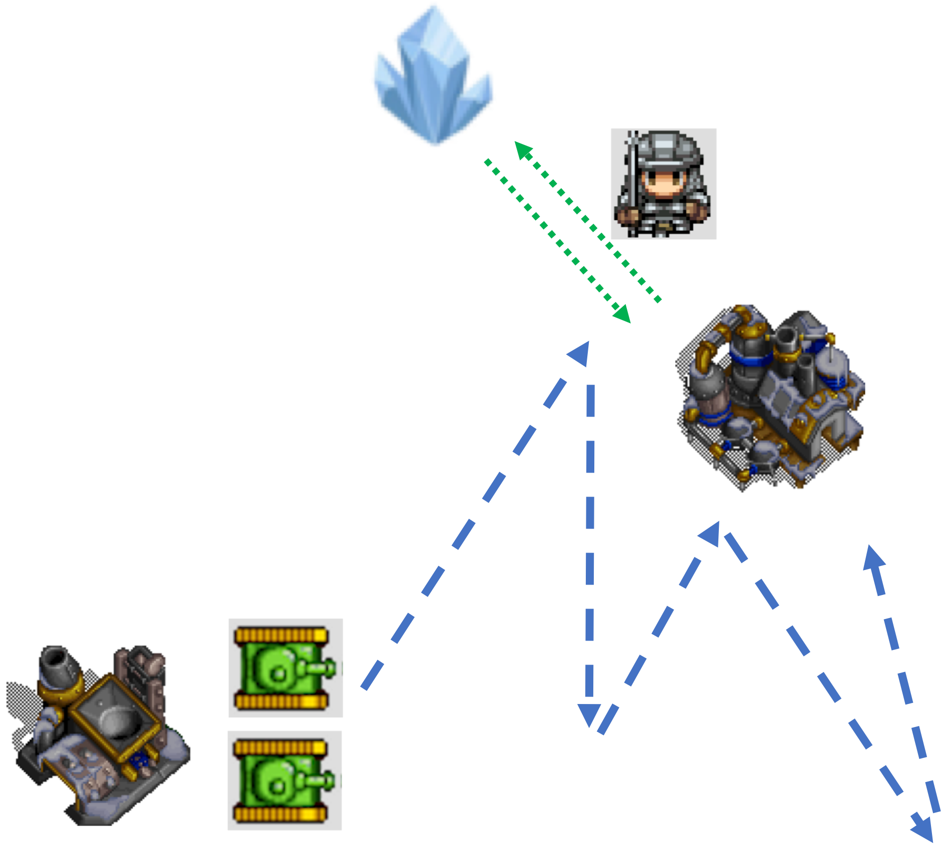
No.	Action name	Descriptions
1	IDLE	Do nothing
2	BUILD WORKER	If the base is idle, build a worker
3	BUILD BARRACK	Move a worker (gathering or idle) to an empty place and build a barrack.
4	BUILD MELEE ATTACKER	If we have an idle barrack, build an melee attacker.
5	BUILD RANGE ATTACKER	If we have an idle barrack, build a range attacker.
6	HIT AND RUN	If we have range attackers, move towards opponent base and attack. Take advantage of their long attack range and high movement speed to hit and run if enemy counter-attack.
7	ATTACK	All melee and range attackers attack the opponent's base.
8	ATTACK IN RANGE	All melee and range attackers attack enemies in sight.
9	ALL DEFEND	All troops attack enemy troops near the base and resource.

Rule-based AIs



AI_SIMPLE

Build 5 tanks and attack

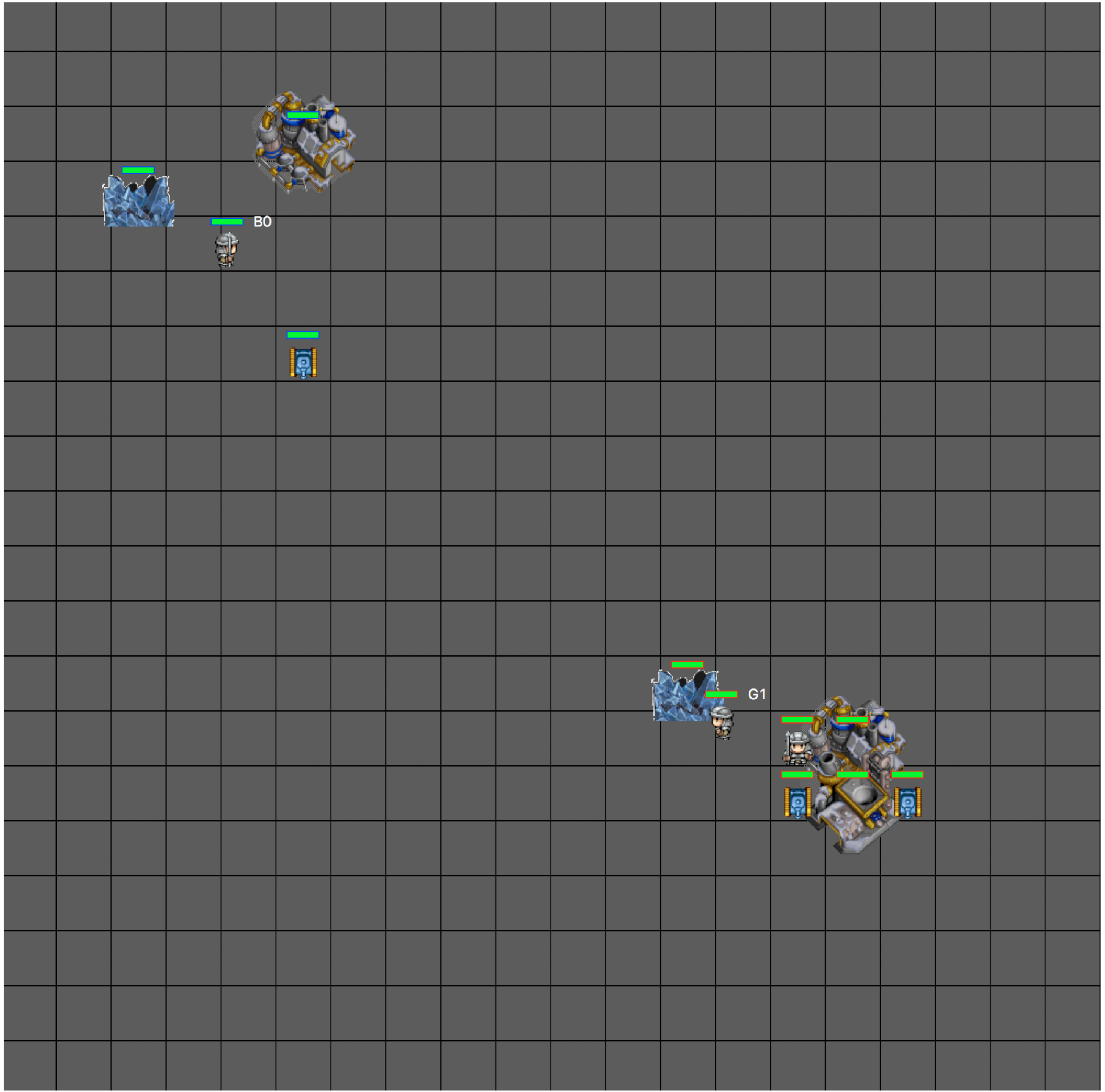


AI_HIT_AND_RUN

Build 2 tanks and harass

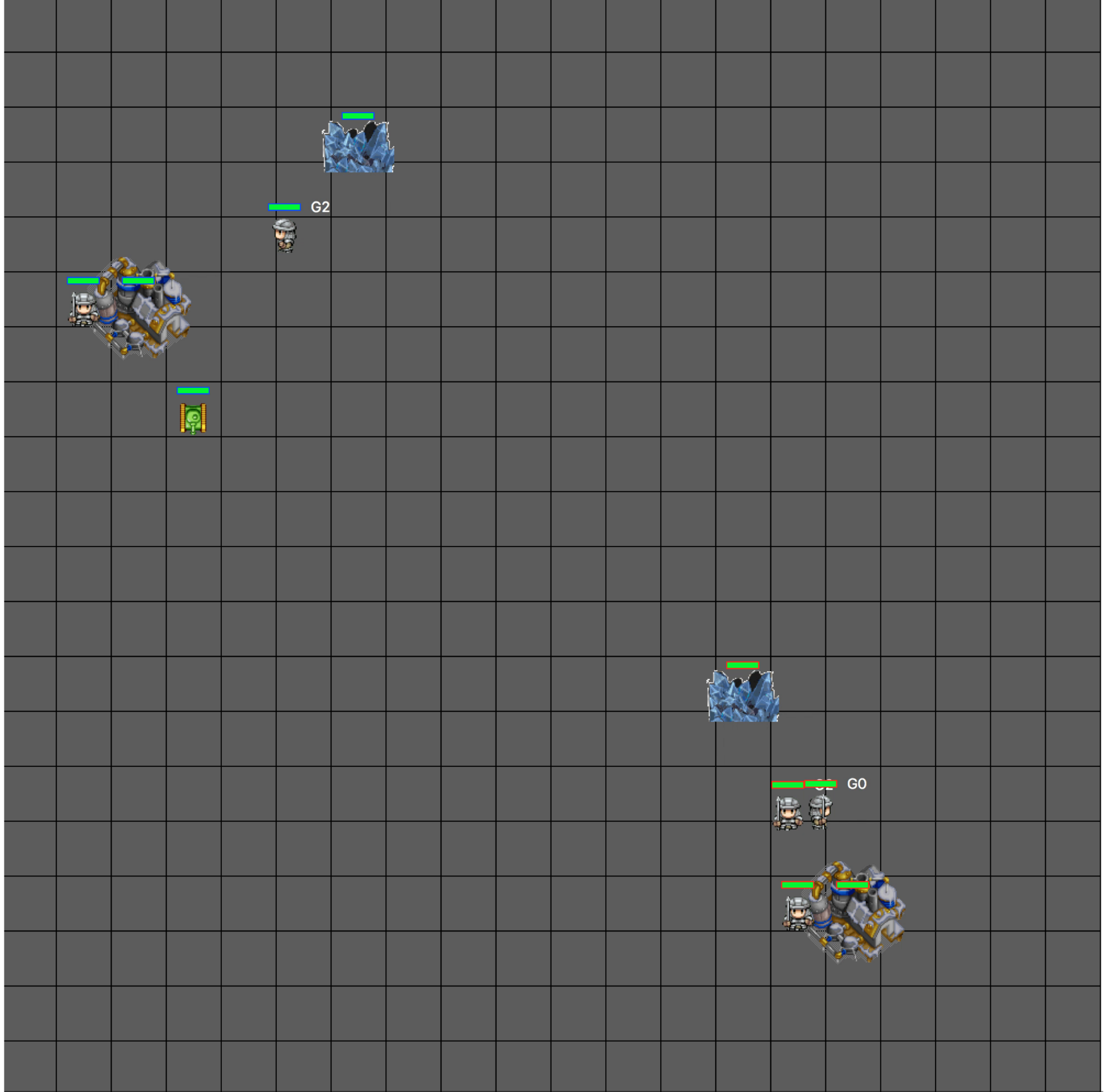
<u>MiniRTS trains with a single GPU and 6 CPUs in half a day.</u>

Trained AI



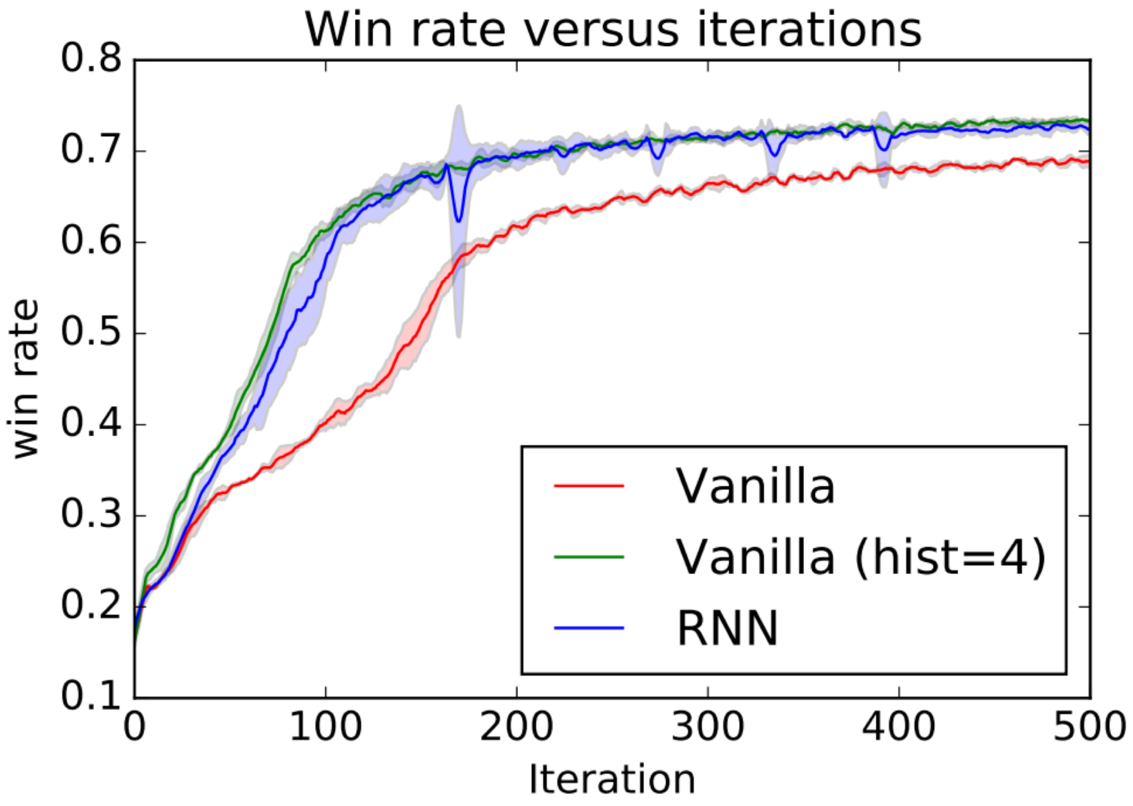
AI_SIMPLE

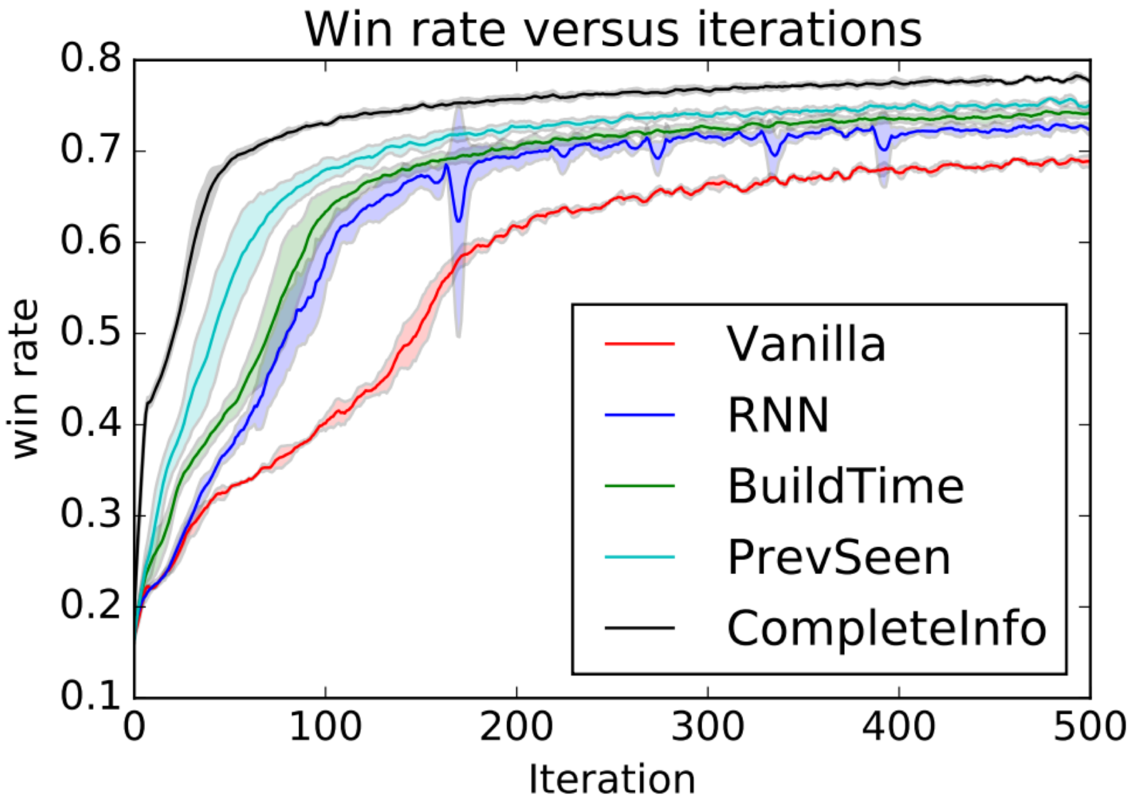
Trained AI



AI_SIMPLE

Comparison between different models



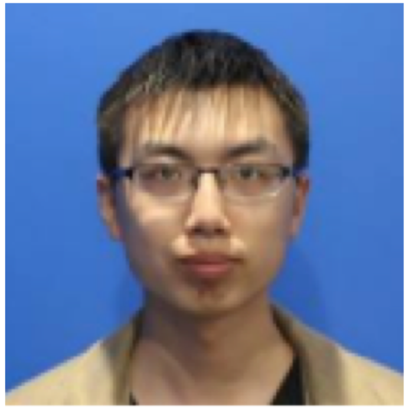


Method	*Vanilla*	*Vanilla*(hist=4)	RNN	*BuildHistory*	*PrevSeen*	Complete Info
Win rate	72.9±1.8	79.8±0.7	79.7±1.3	80.8±1.7	81.4±0.8	81.7±0.7

First Person Shooter (FPS) Game

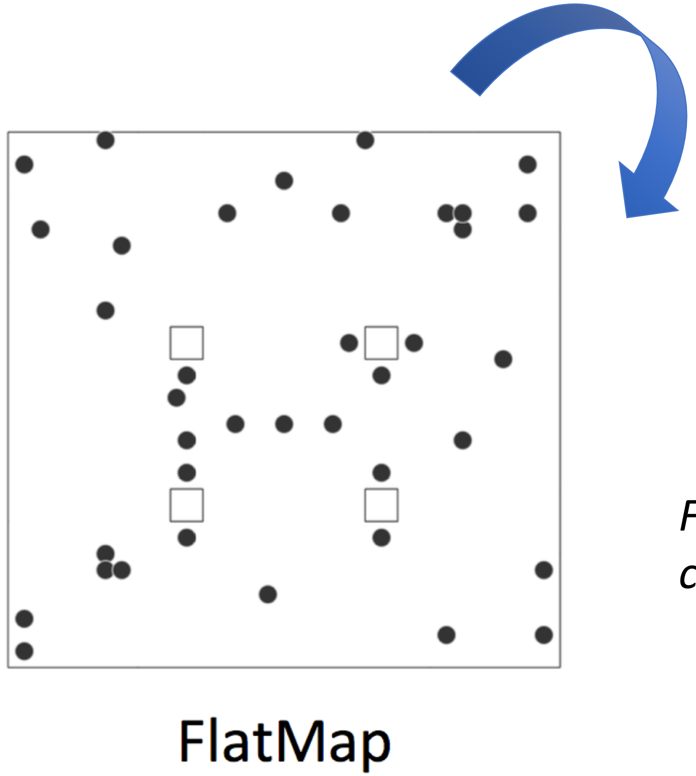
Yuxin Wu, Yuandong Tian, ICLR 2017

Play the game from the raw image!

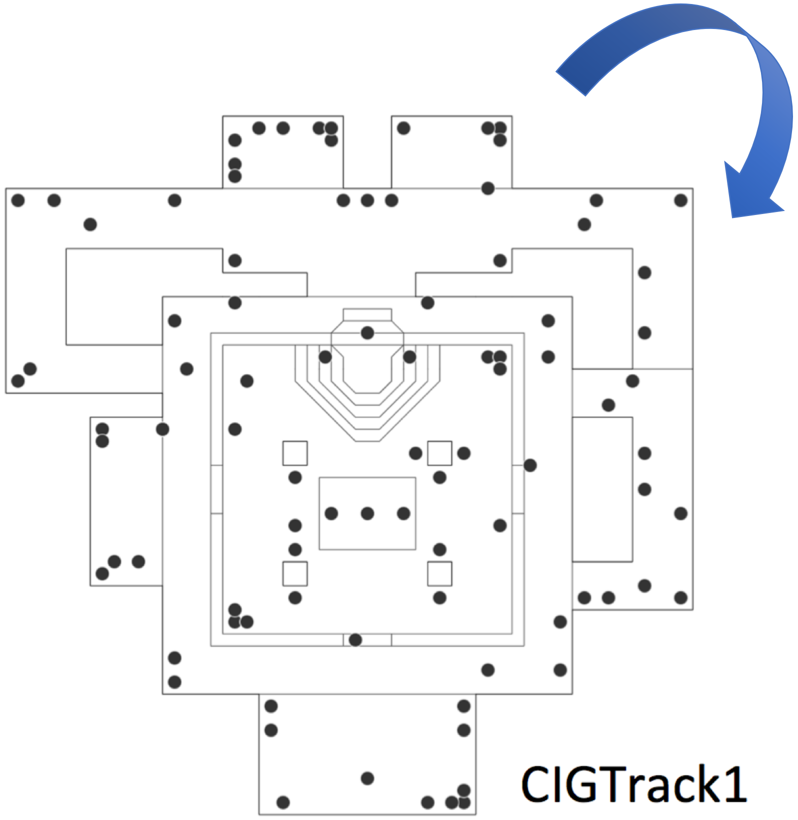


Yuxin Wu

Curriculum Training



From simple to complicated



VizDoom AI Competition 2016 (Track1)

We won the first place!

Rank	Bot	1	2	3	4	5	6	7	8	9	10	11	Total frags
1	F1	**56**	**62**	n/a	**54**	**47**	43	**47**	**55**	**50**	**48**	**50**	**559**
2	Arnold	36	34	**42**	36	36	**45**	36	39	n/a	33	36	413
3	CLYDE	37	n/a	38	32	37	30	46	42	33	24	44	393

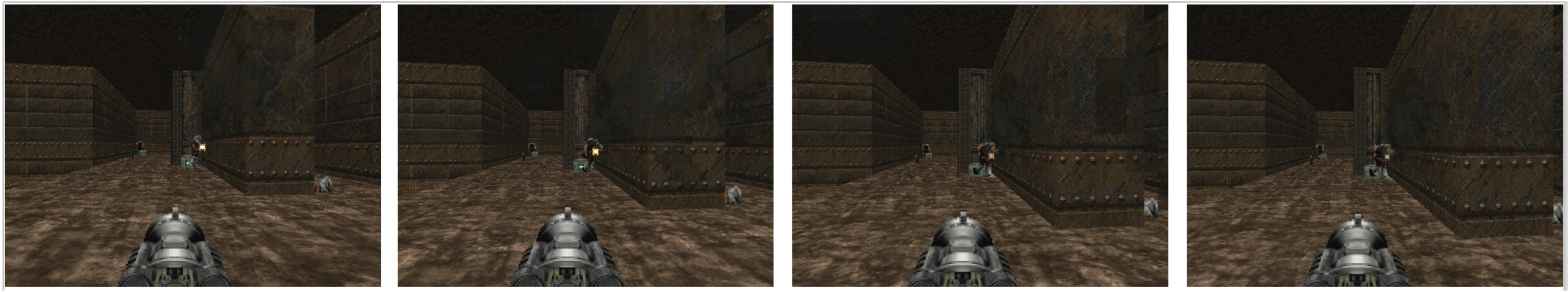
Videos:

https://www.youtube.com/watch?v=94EPSjQH38Y
https://www.youtube.com/watch?v=Qv4esGWOg7w&t=394s

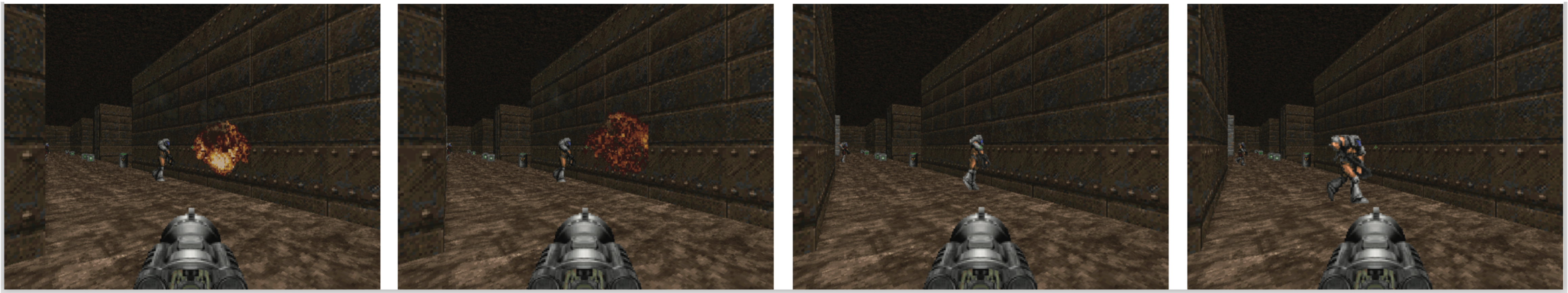

14
AMMO
93%
HEALTH
1
FRAGS
0%
ARMOR
0 / 200
0 / 50
14 / 50
0 / 300

Visualization of Value functions

Best 4 frames (agent is about to shoot the enemy)



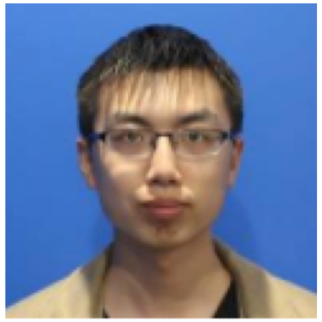
Worst 4 frames (agent missed the shoot and is out of ammo)



House3D: A rich and realistic 3D environment

Yi Wu

Georgia Gkioxari



Yuxin Wu

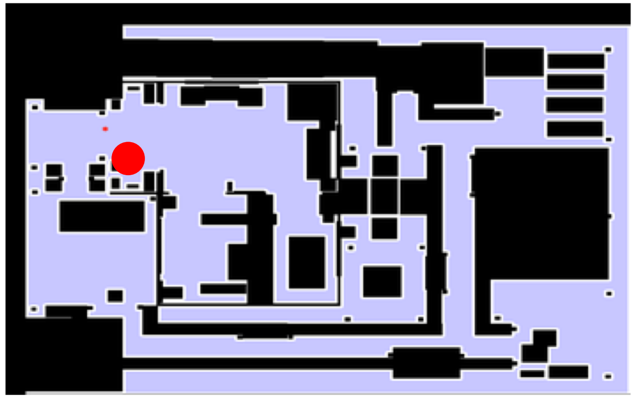
[Yi Wu et al, Building Generalizable Agents with a Realistic and Rich 3D Environment, ICLR 2018 workshop]

SUNCG Dataset

SUNCG dataset, 45K scenes, all objects are fully labeled.

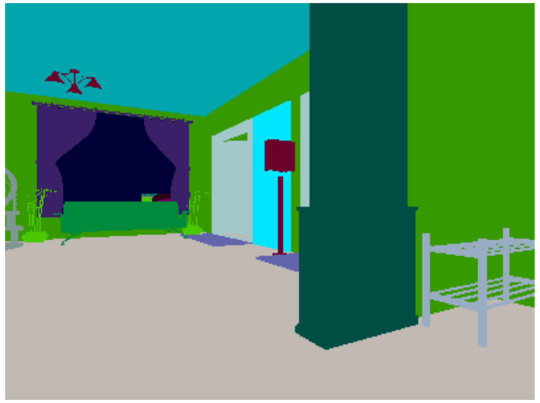
House3D

SUNCG dataset, 45K scenes, all objects are fully labeled.



Top-down map

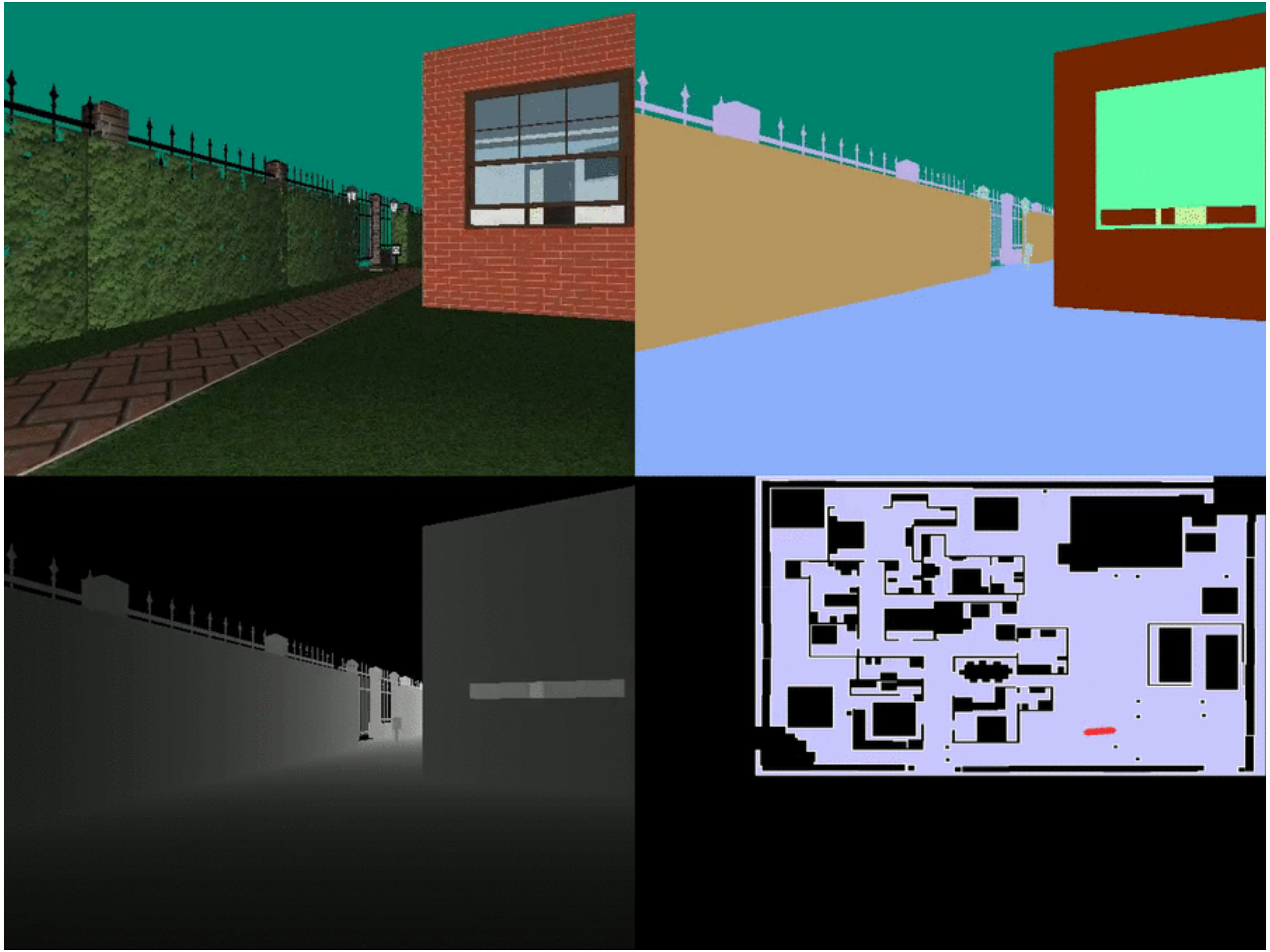
Depth



Segmentation mask

RGB image

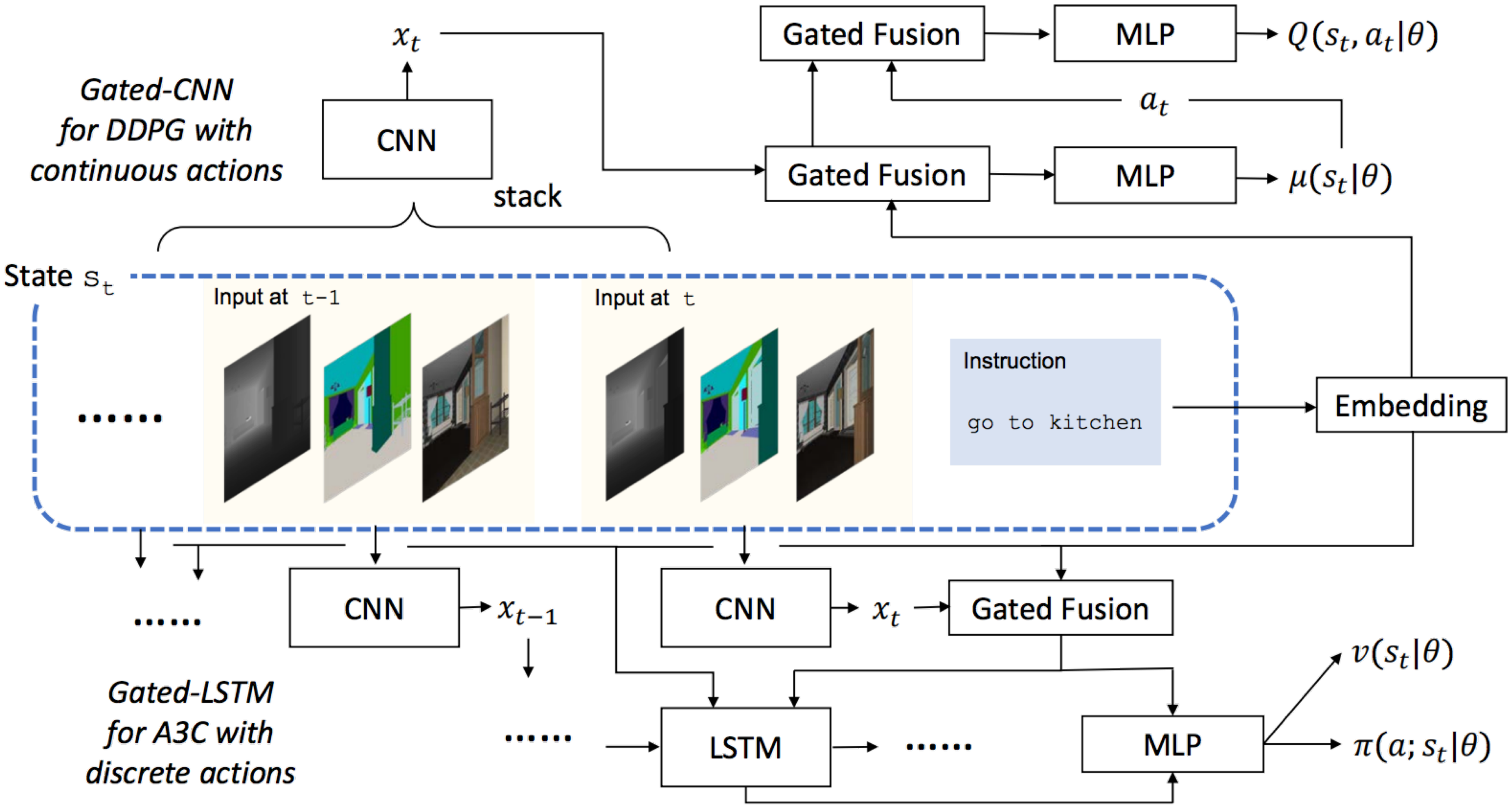
https://github.com/facebookresearch/House3D



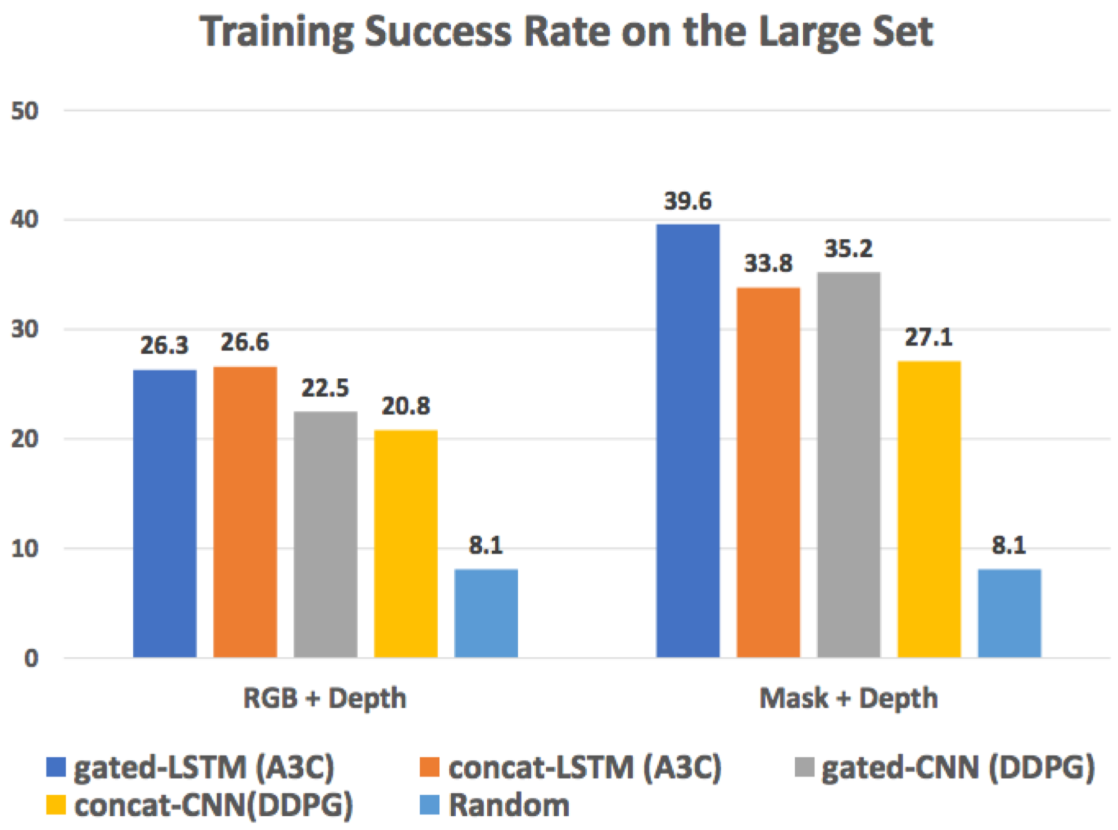
Comparison

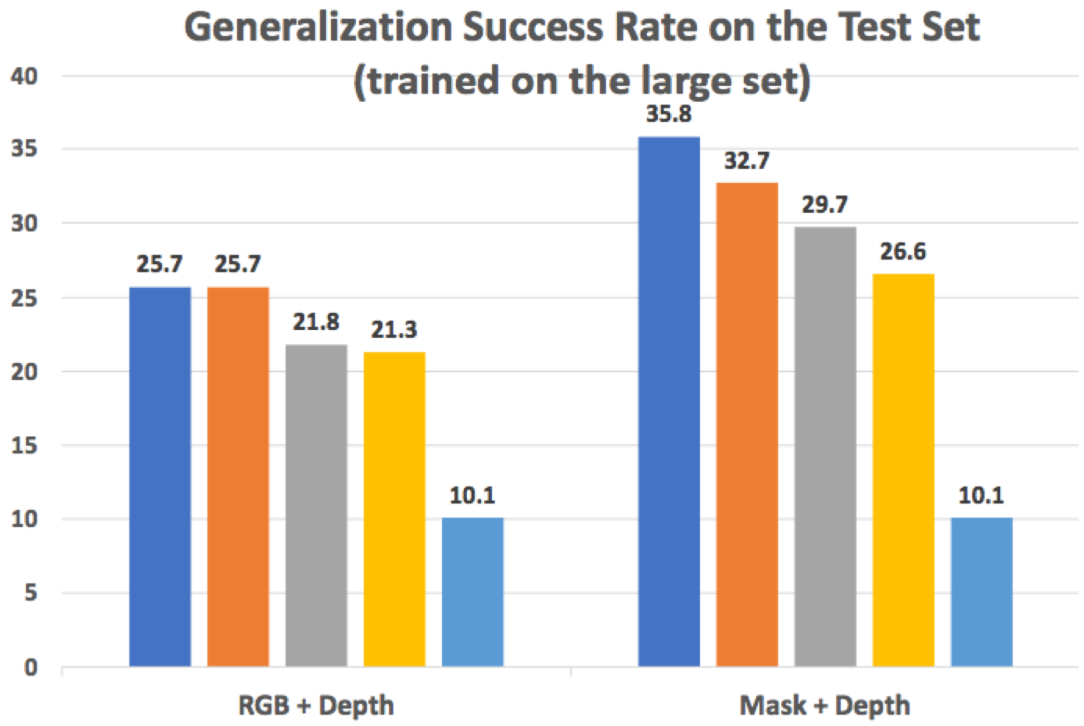
Environment	3D	Realistic	Large-scale	Fast-speed	Customizable
Atari (Bellemare et al., 2013)				●	
OpenAI Universe (Shi et al., 2017)		●	●		●
Malmo (Johnson et al., 2016)	●		●	●	●
DeepMind Lab (Beattie et al., 2016)	●			●	●
VizDoom (Kempka et al., 2016)	●			●	●
AI2-THOR (Zhu et al., 2017)	●	●		●	
House3D	●	●	●	●	●

Architectures

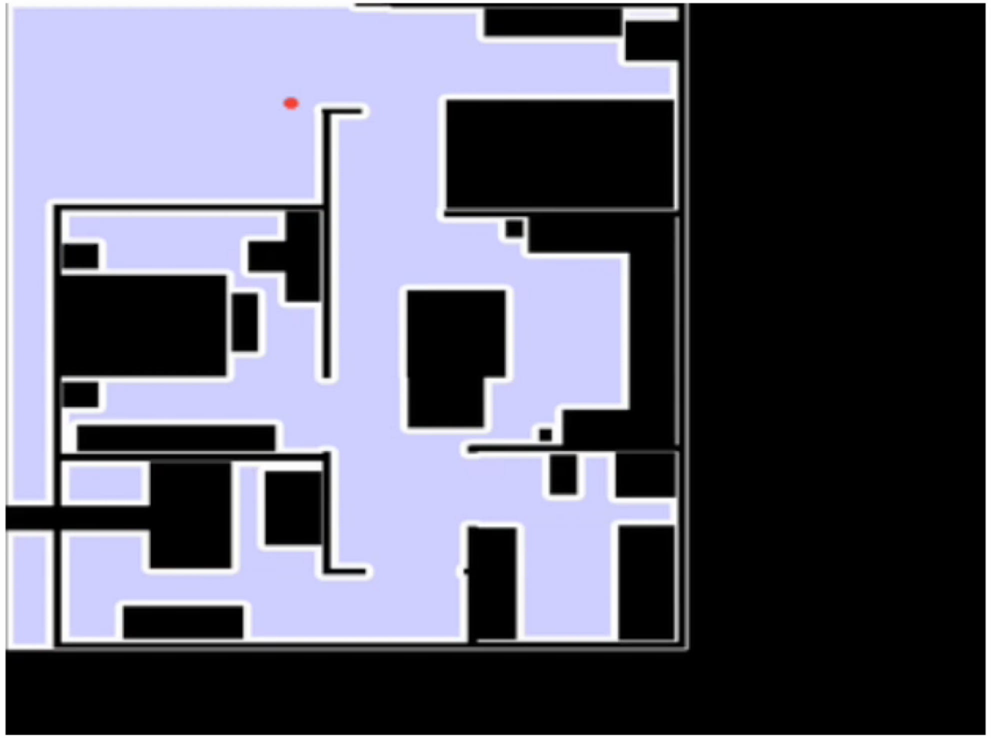


Generalization capability

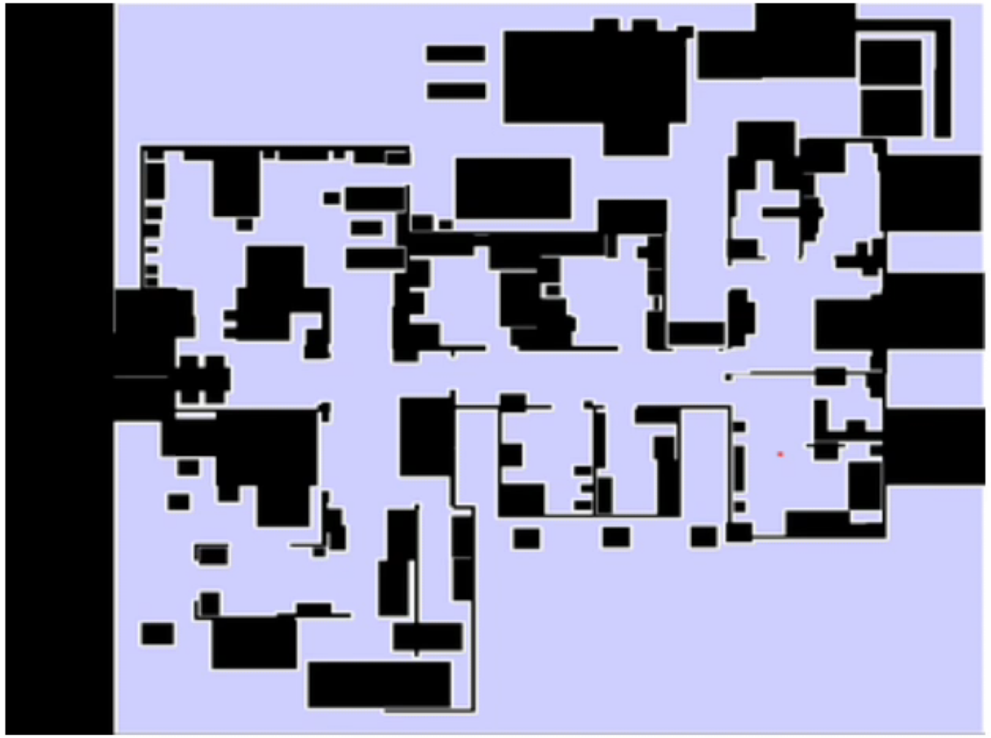




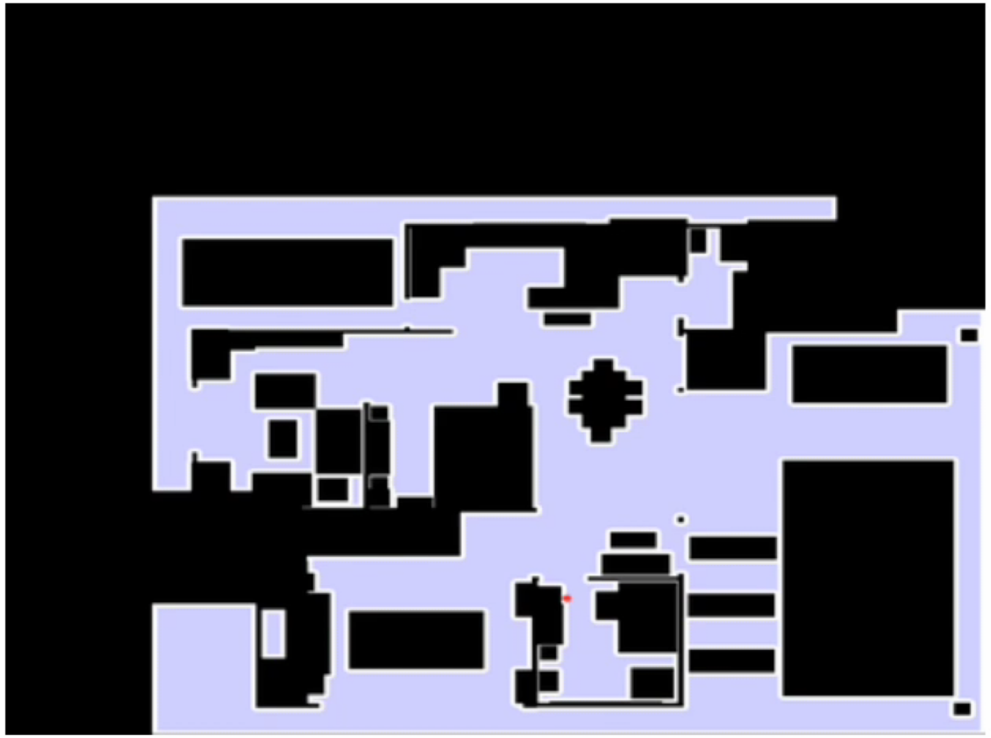
Target: Bathroom



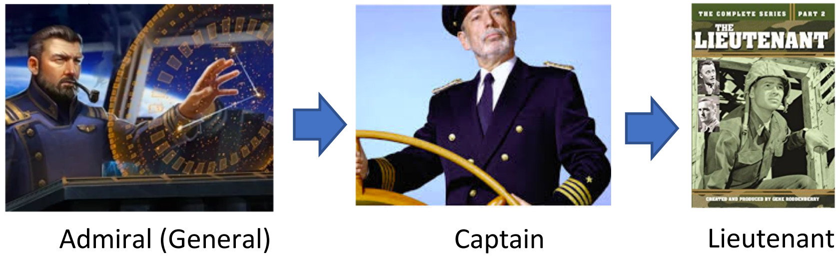
Target: Kitchen



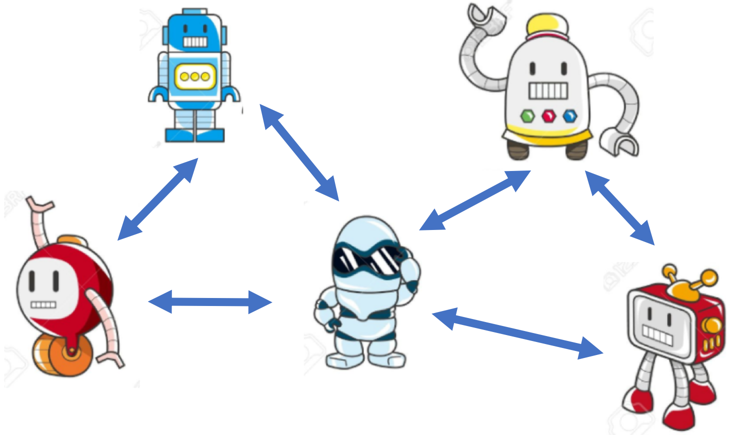
Target: Dining Room



Future Directions

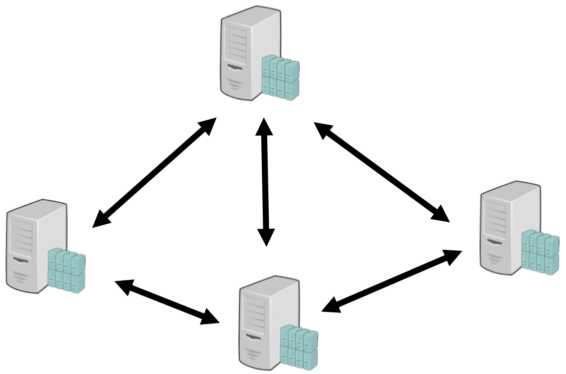


Hierarchical RL

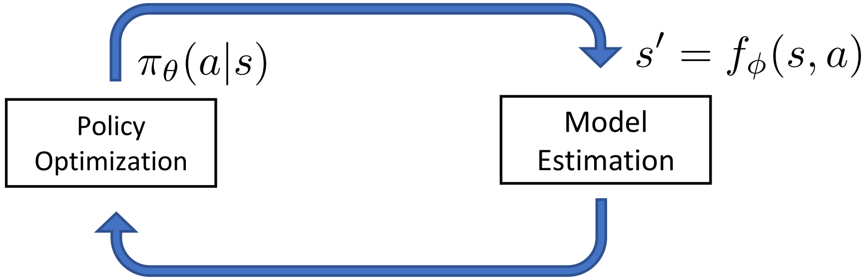


Multi-Agent

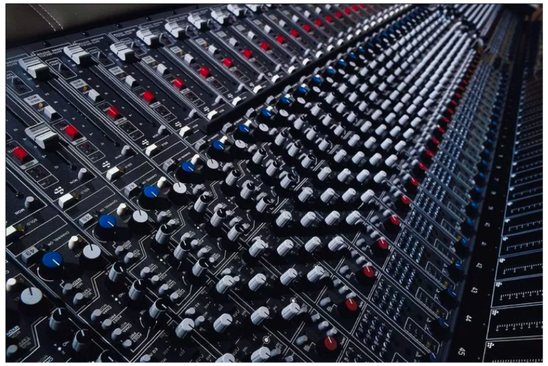
RL applications



RL Systems



Model-based RL



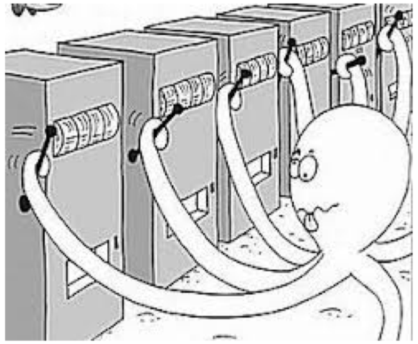
RL for Optimization

How to do well in Reinforcement Learning?

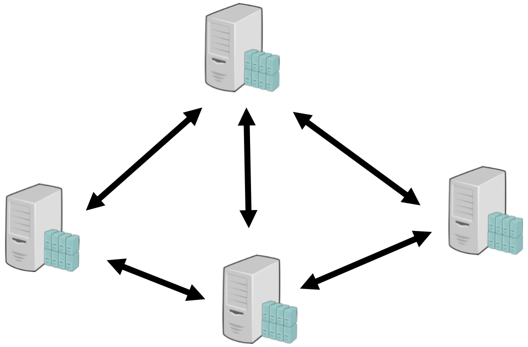
$$Q(s,a) \quad V^{\pi}(s)$$

$$V(s) \quad \pi(a|s) \quad Q^{\pi}(s,a)$$

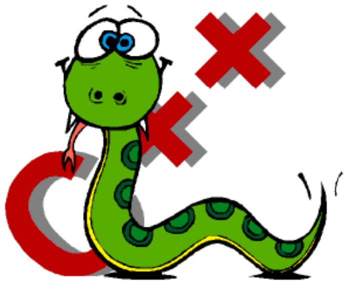
Strong math skills



Parameter tuning skills



Experience on
(distributed) systems



Strong coding skills

Thanks!