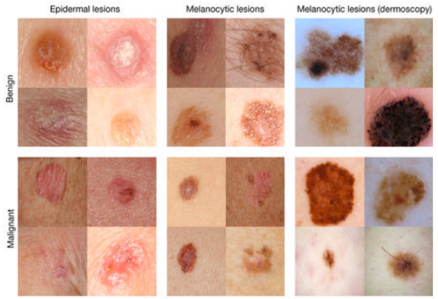
Building Scalable Framework and Environment of Reinforcement Learning

Yuandong Tian

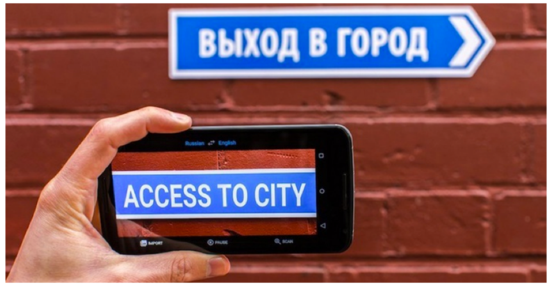
Facebook AI Research

AI works in a lot of situations

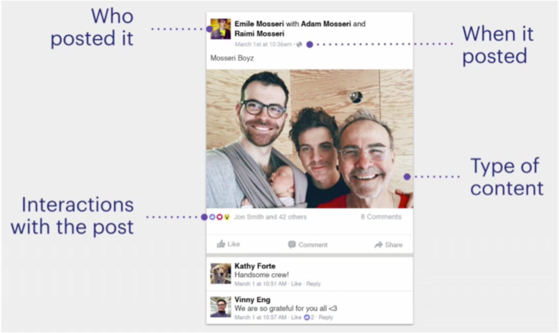
Object Recognition



Medical



Translation

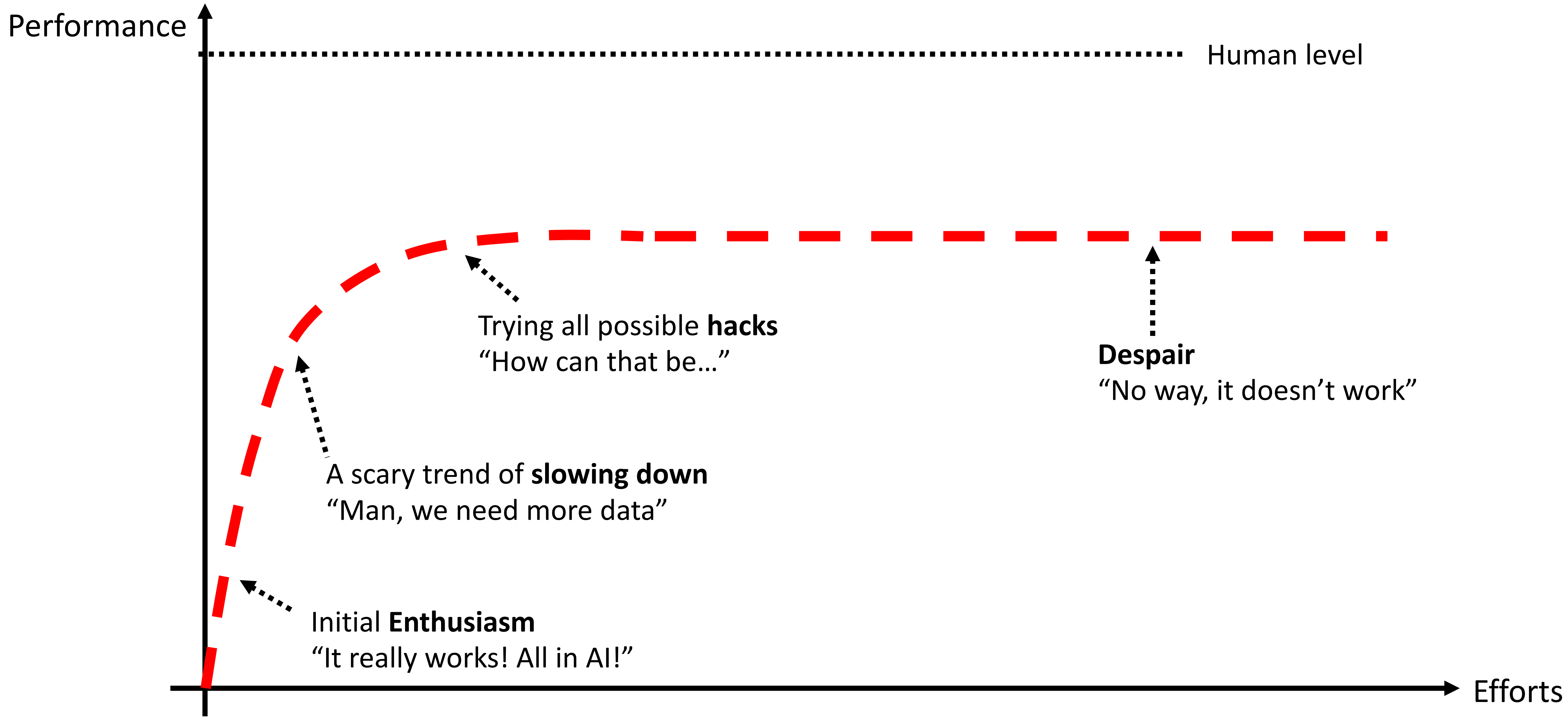


Personalization

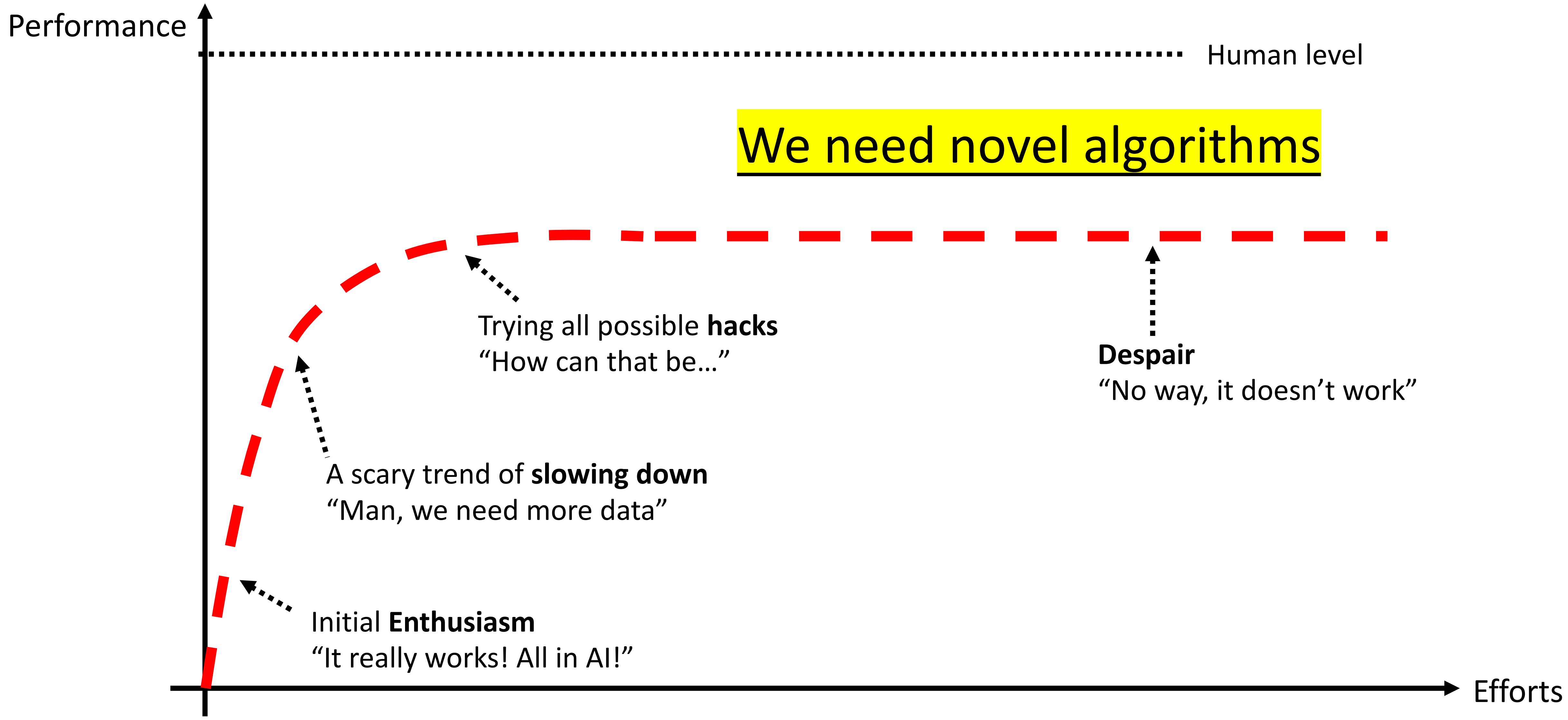
Surveillance

Speech Recognition

What AI still needs to improve

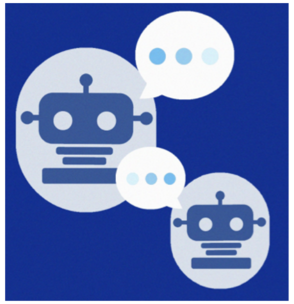


What AI still needs to improve



What AI still needs to improve

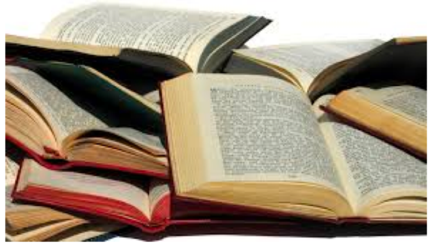
Question Answering



ChatBot

Common Sense

Program Induction



Text Generation

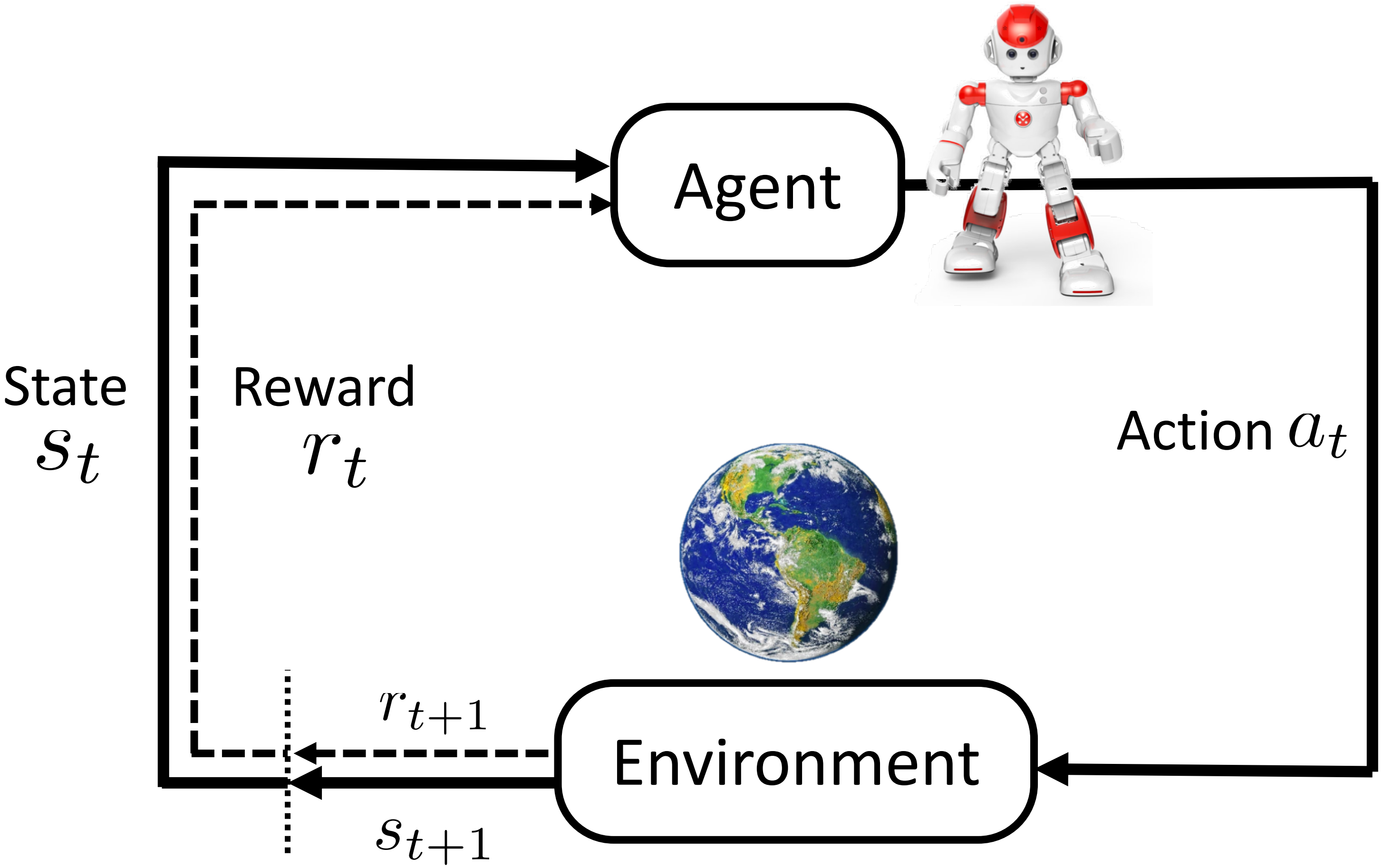
High-order Reasoning

Autonomous Driving

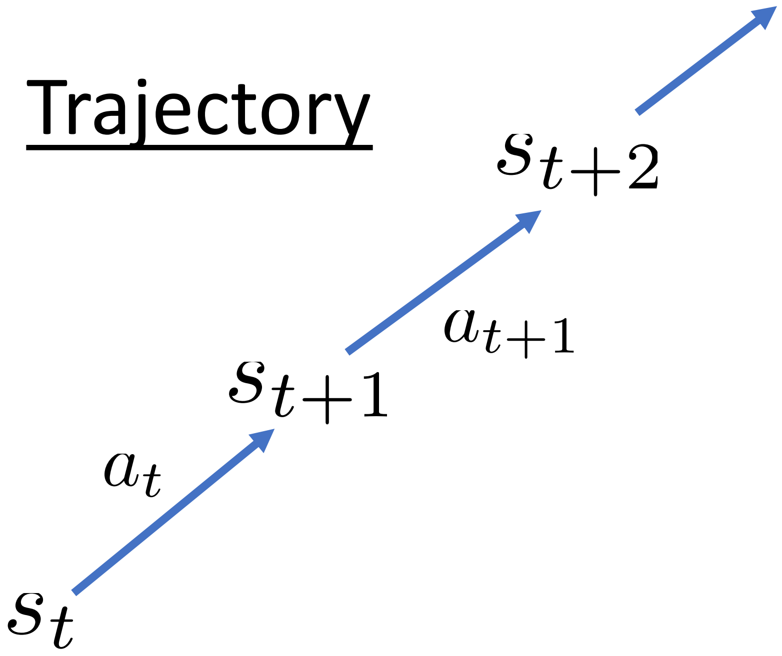
Home Robotics

Few supervised data
Complicated environments
Lots of Corner cases.

Reinforcement Learning



<u>Trajectory</u>



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

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

Game as a testbed of Reinforcement Learning

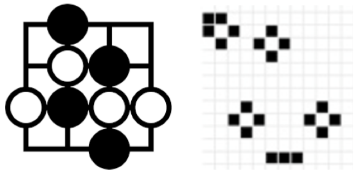
Infinite supply of *fully* labeled data

Controllable and replicable

Low cost per sample

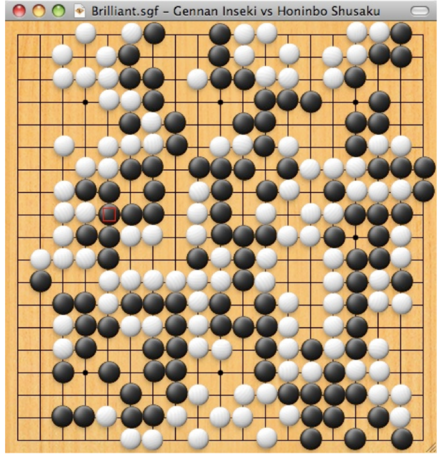
Faster than real-time

Less safety and ethical concerns



Complicated dynamics with simple rules.

Game as a testbed of Reinforcement Learning

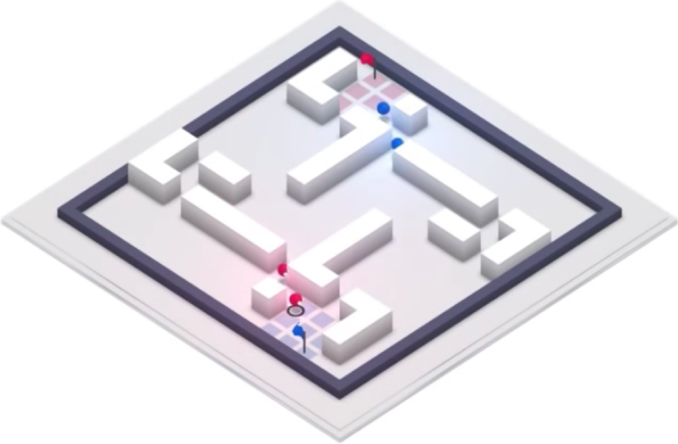


Go

Shogi

StarCraft II

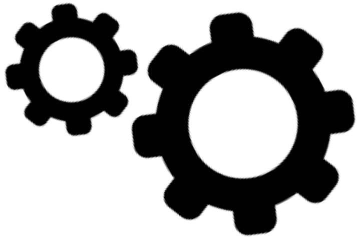
Chess



Quake 3

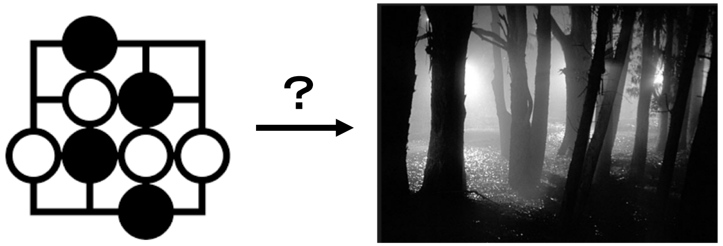
Dota 2

Game as a testbed of Reinforcement Learning



Need good simulator

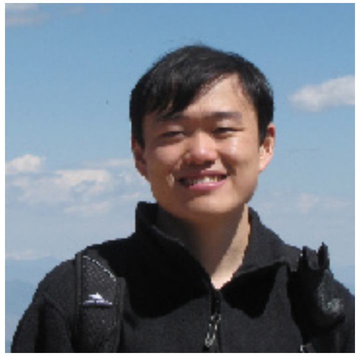
Require a lot of data/resources.



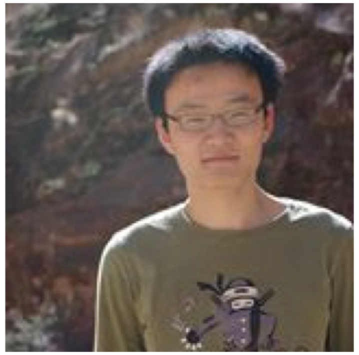
Sim2real issue

Applications?

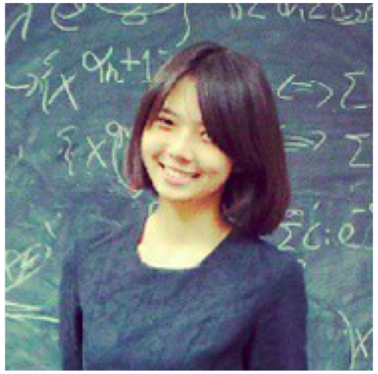
ELF: Extensive, Lightweight and Flexible Framework for Game Research



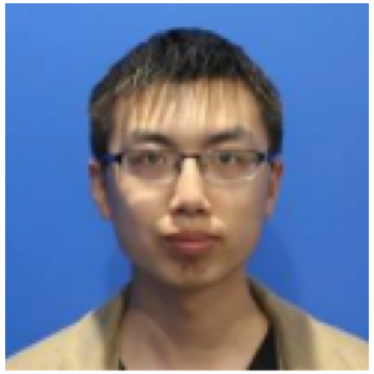
Yuandong Tian



Qucheng Gong



Wenling Shang



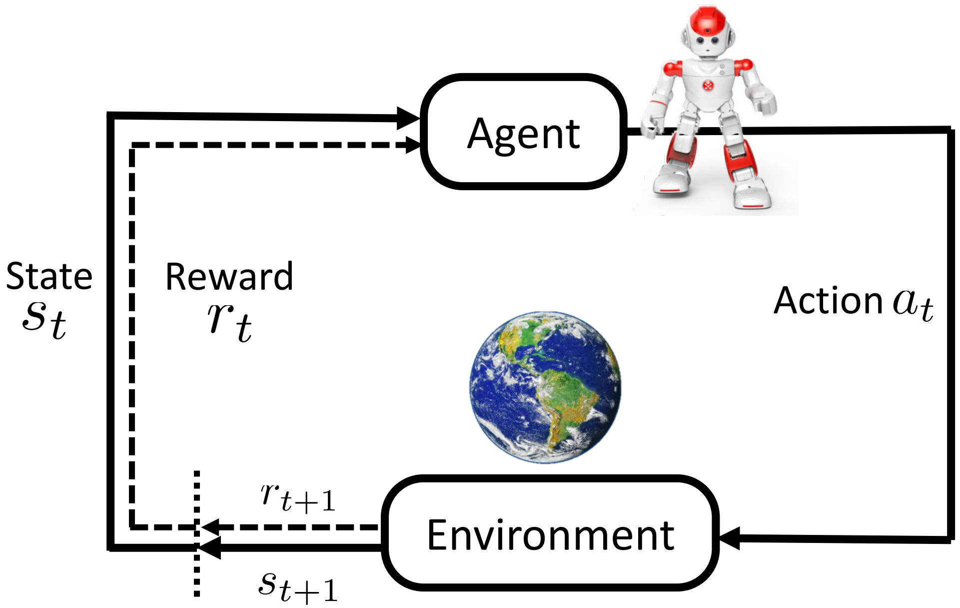
Yuxin Wu

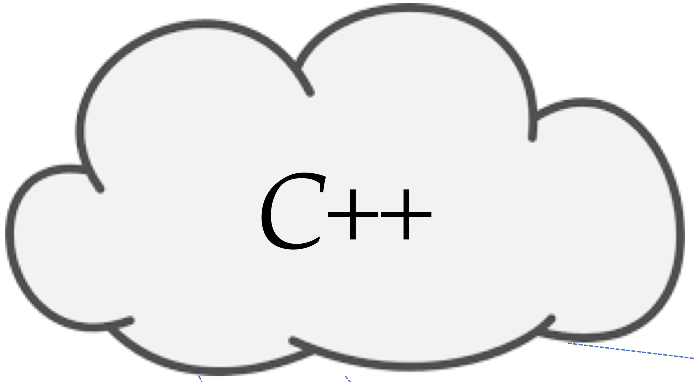
Larry Zitnick

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

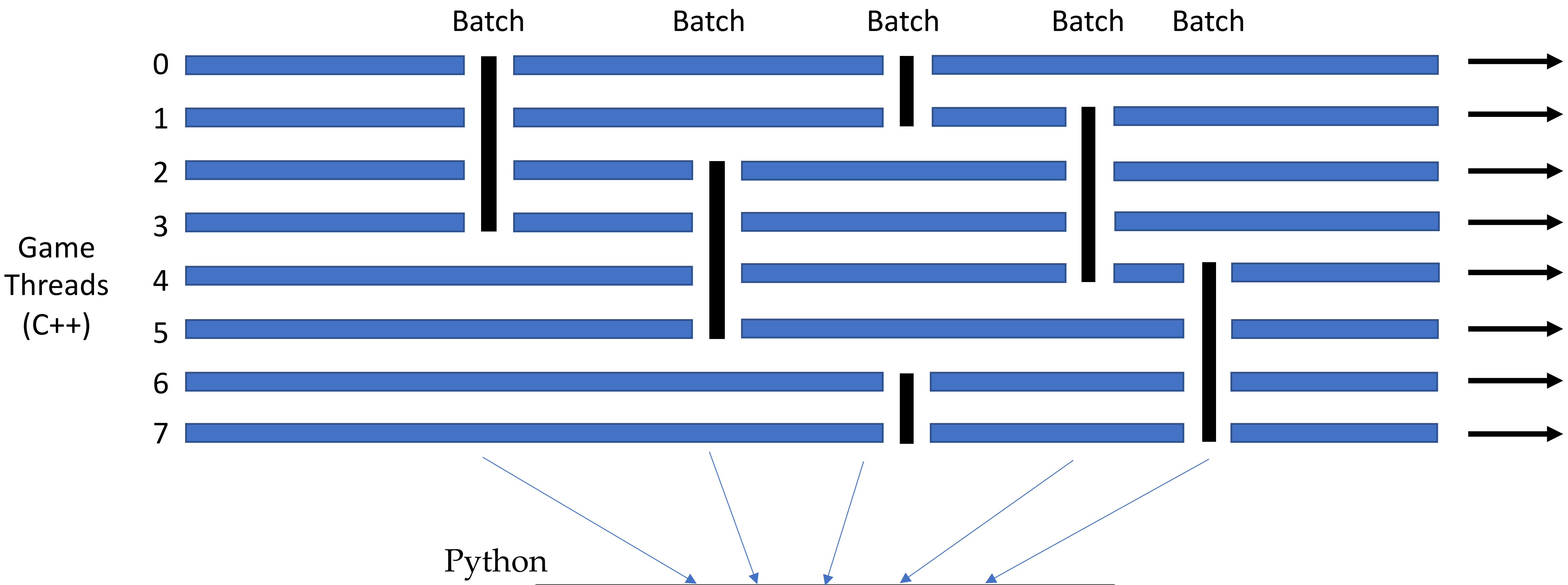




Python

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

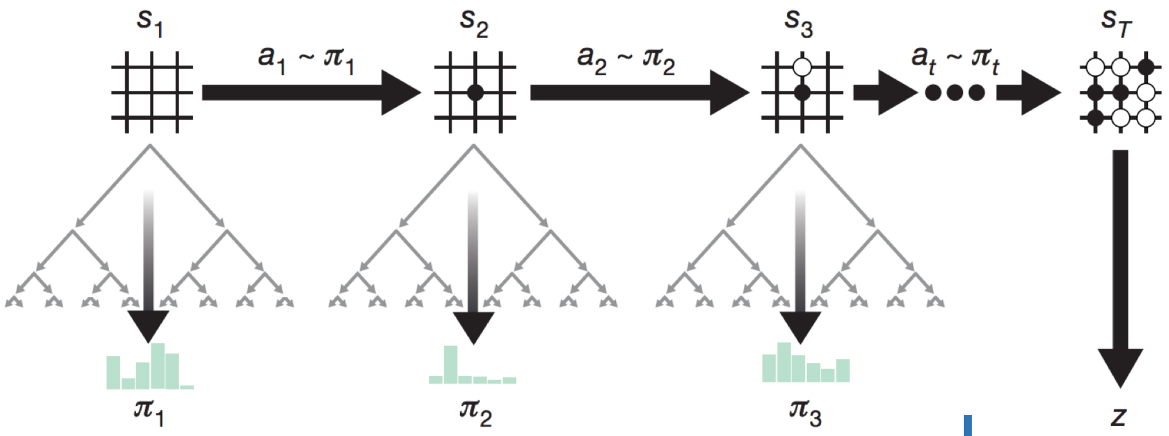
How ELF works



Python

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

Reimplementation of AlphaGo Zero



Generate Training data

Zero-human knowledge

Self-Replays

Update Models

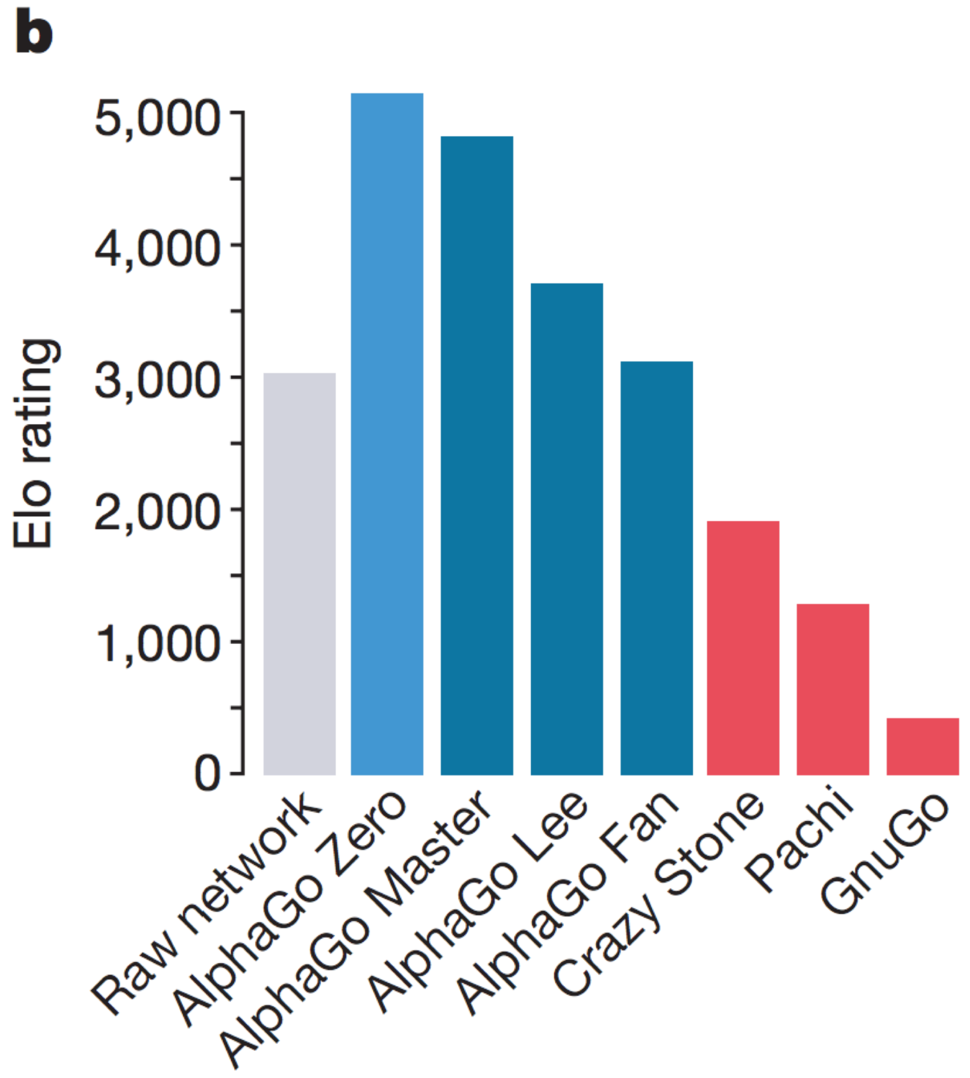
θ_i

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

AlphaGo Zero Strength

- 3 days version
  - 4.9M Games, 1600 rollouts/move
  - 20 block ResNet
  - Defeat AlphaGo Lee.

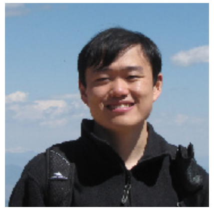
- 40 days version
  - 29M Games, 1600 rollouts/move
  - 40 blocks ResNet.
  - Defeat AlphaGo Master by 89:11



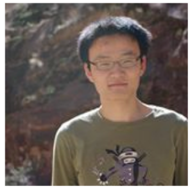
Demystifying AlphaGoZero/AlphaZero

- 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?

ELF OpenGo

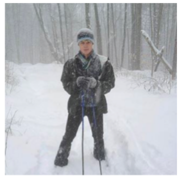

Yuandong Tian


Jerry Ma


Qucheng Gong

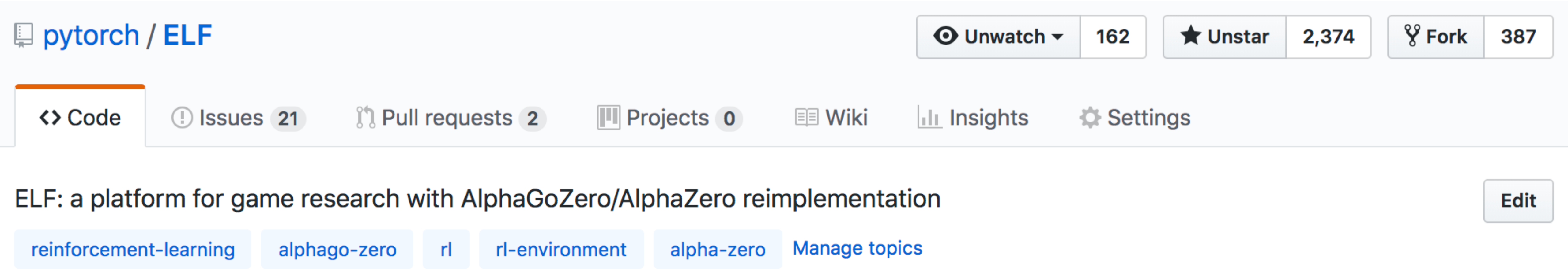

Shubho Sengupta


Zhuoyuan Chen


James Pinkerton


Larry Zitnick

- System can be trained with 2000 GPUs in 2 weeks (20 block version)
- Superhuman performance against professional players and strong bots.
- Abundant ablation analysis
- Decoupled design, code reusable for other games.



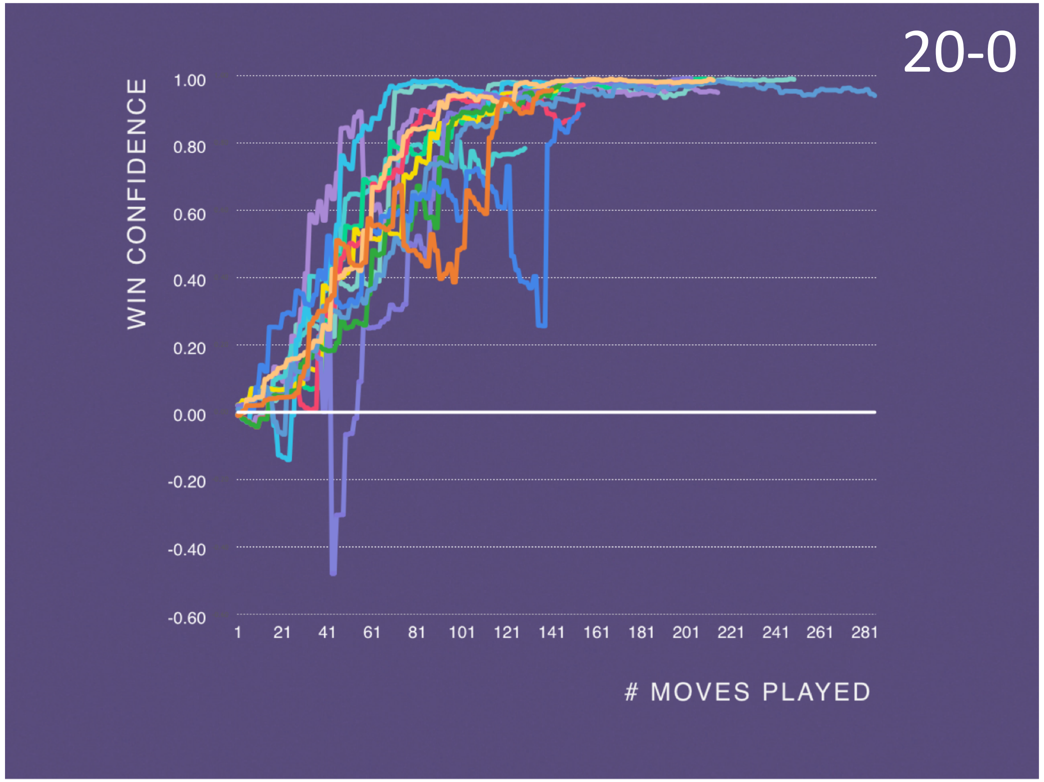
We open source the code and the pre-trained model for the Go and ML community

ELF OpenGo 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



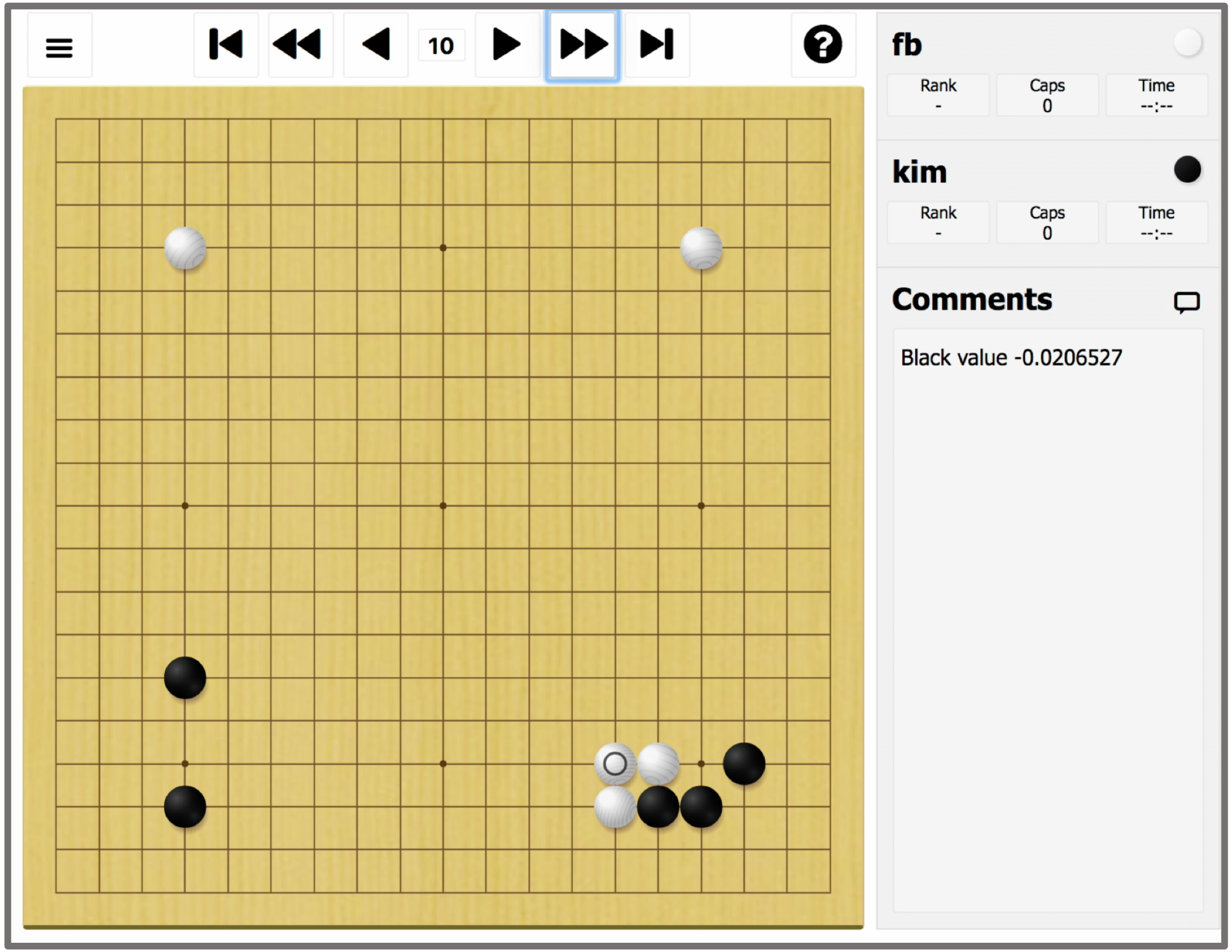
Vs professional players

Single GPU, 2k rollouts, 27-0 against Taiwanese pros.

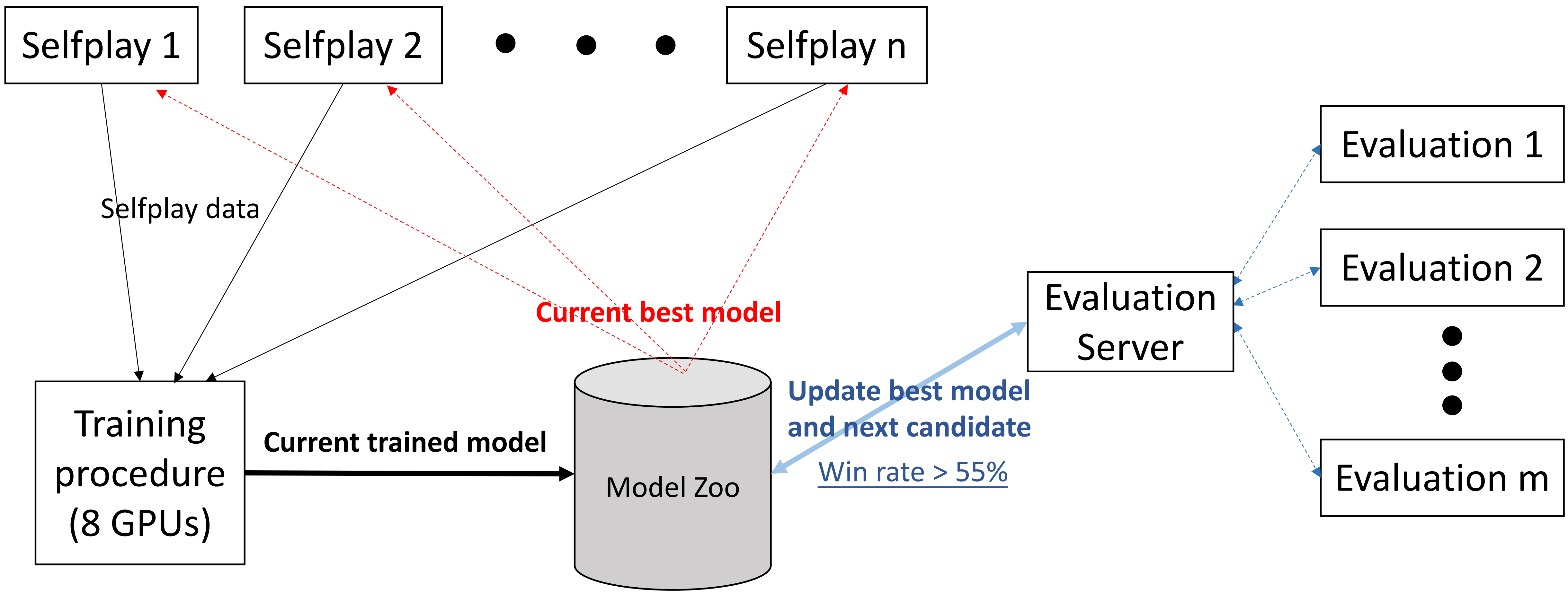
Vs strong bot (LeelaZero)

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

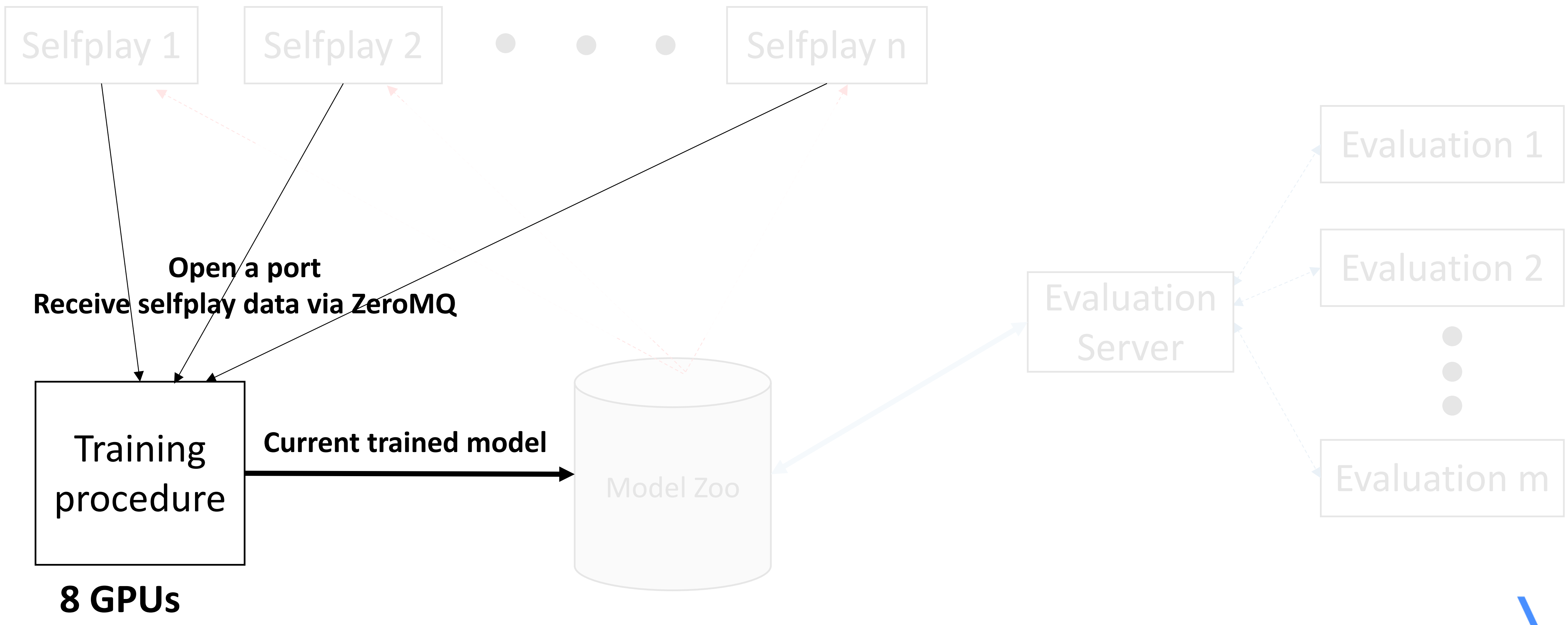
ELF OpenGo Sample Game



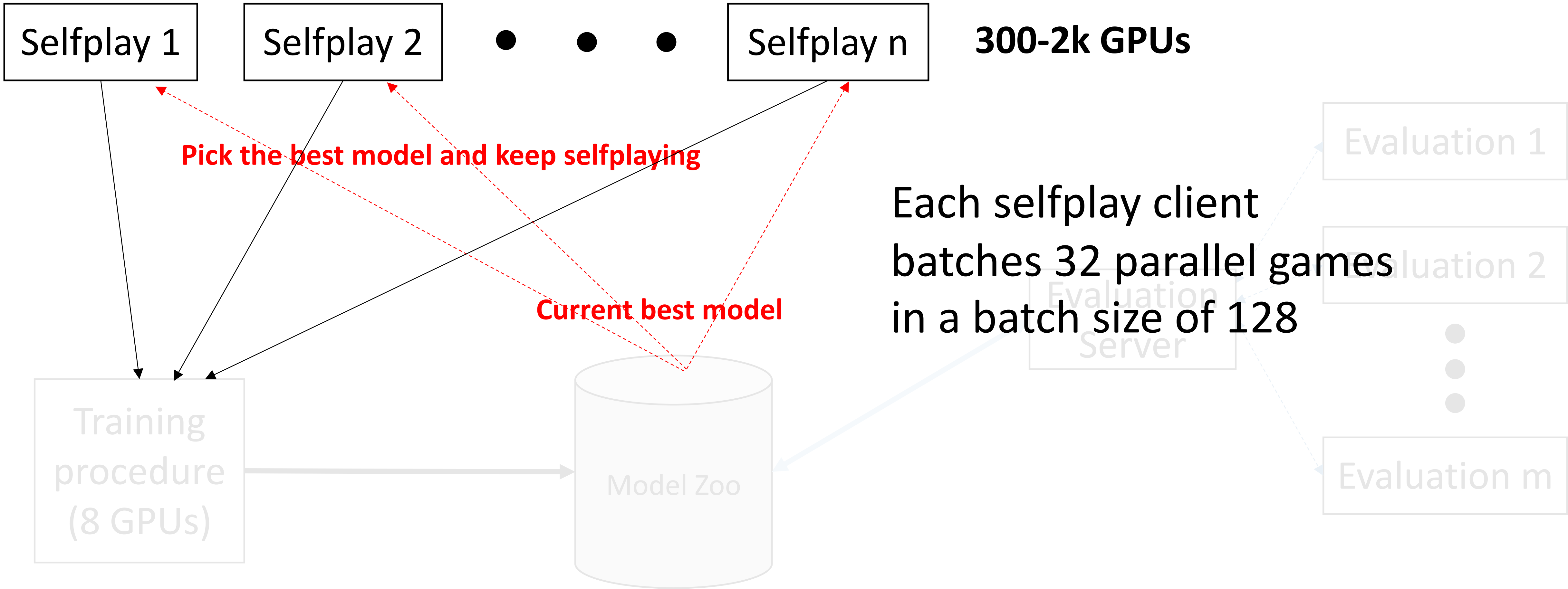
Distributed ELF (version 1)



Distributed System (version 1)

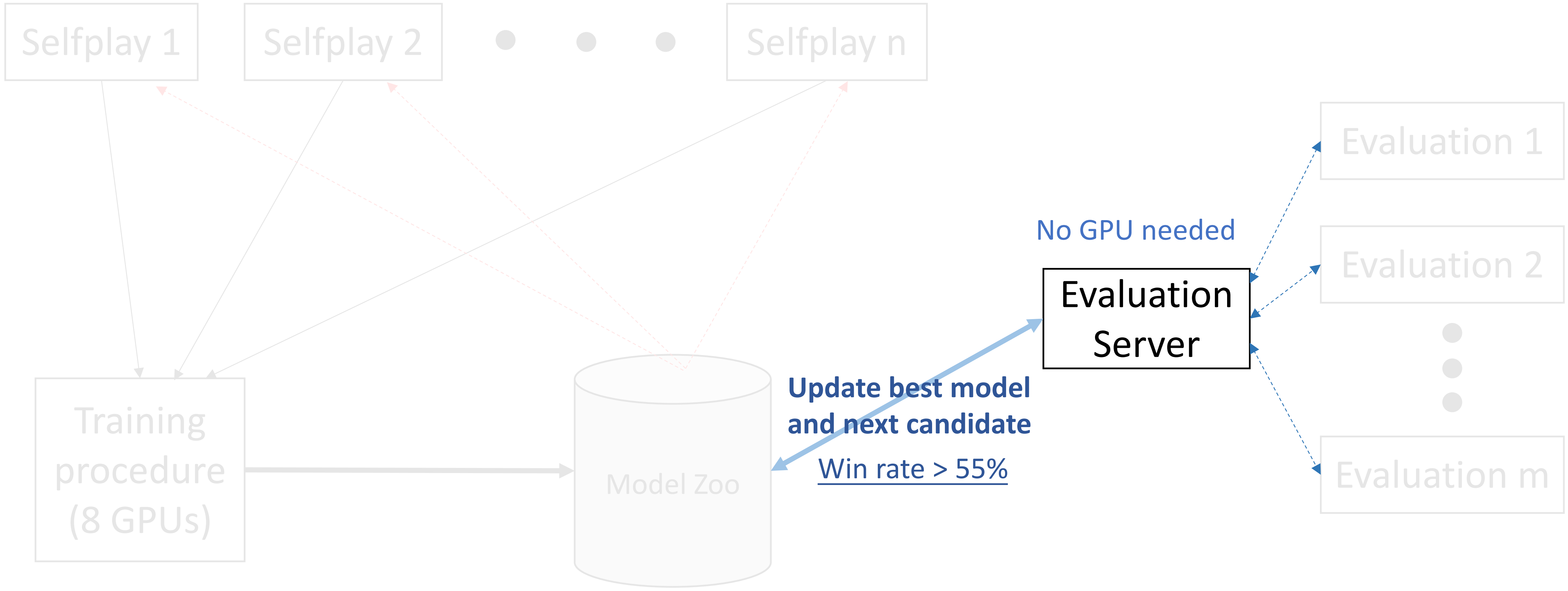


Distributed System (version 1)

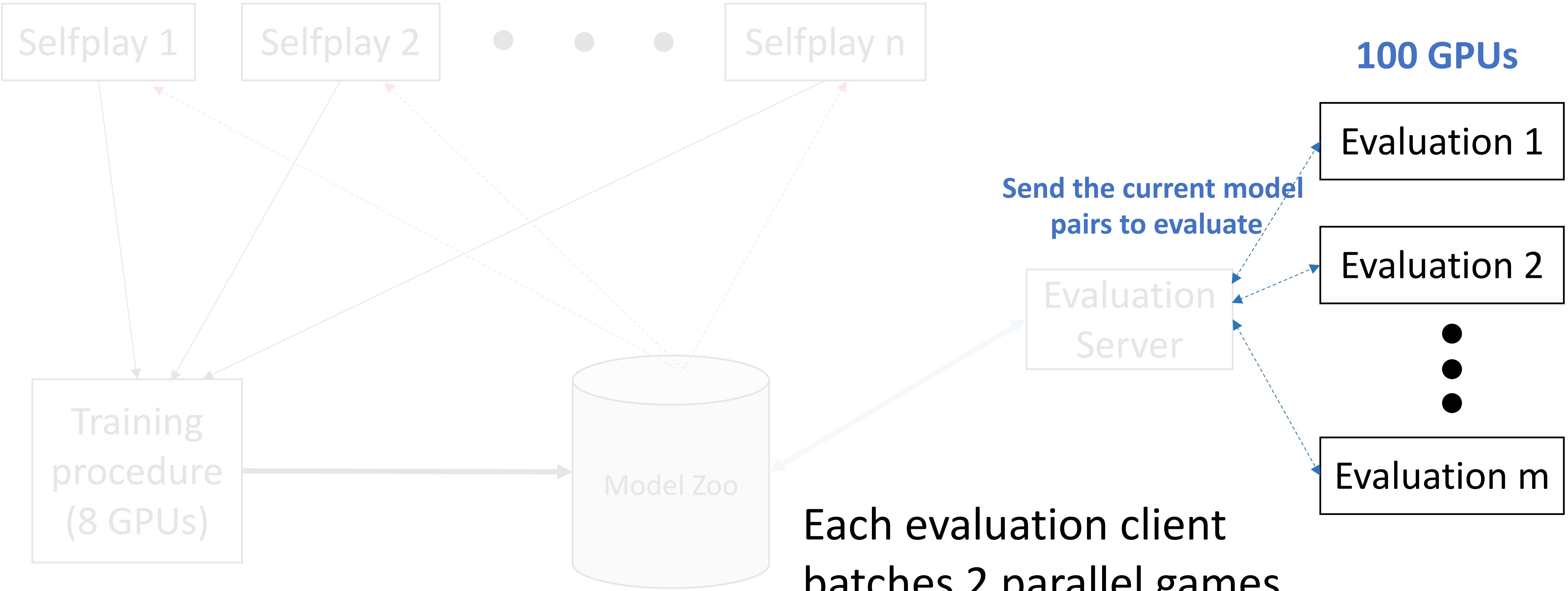


Each selfplay client batches 32 parallel games in a batch size of 128

Distributed System (version 1)

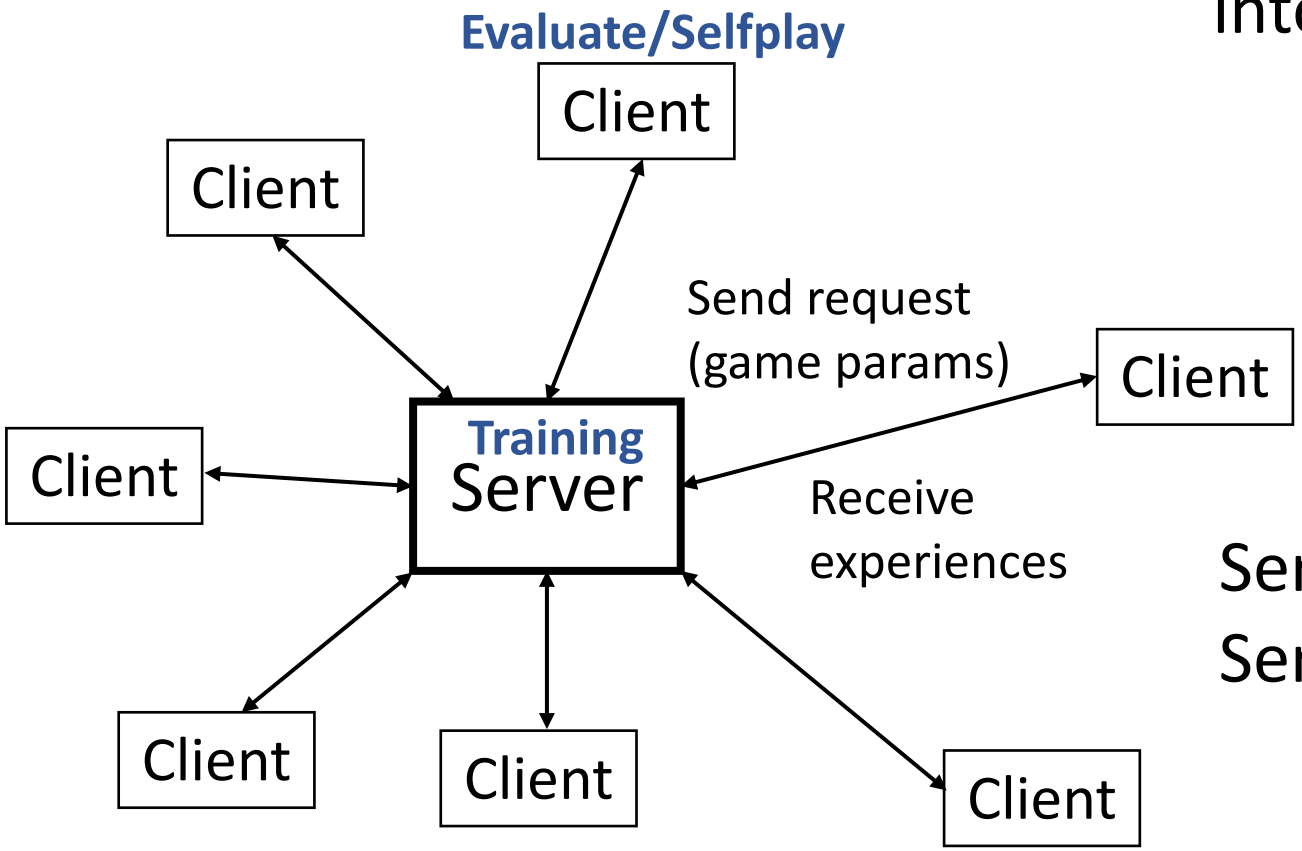


Distributed System (version 1)



Each evaluation client batches 2 parallel games

Distributed ELF (v2)

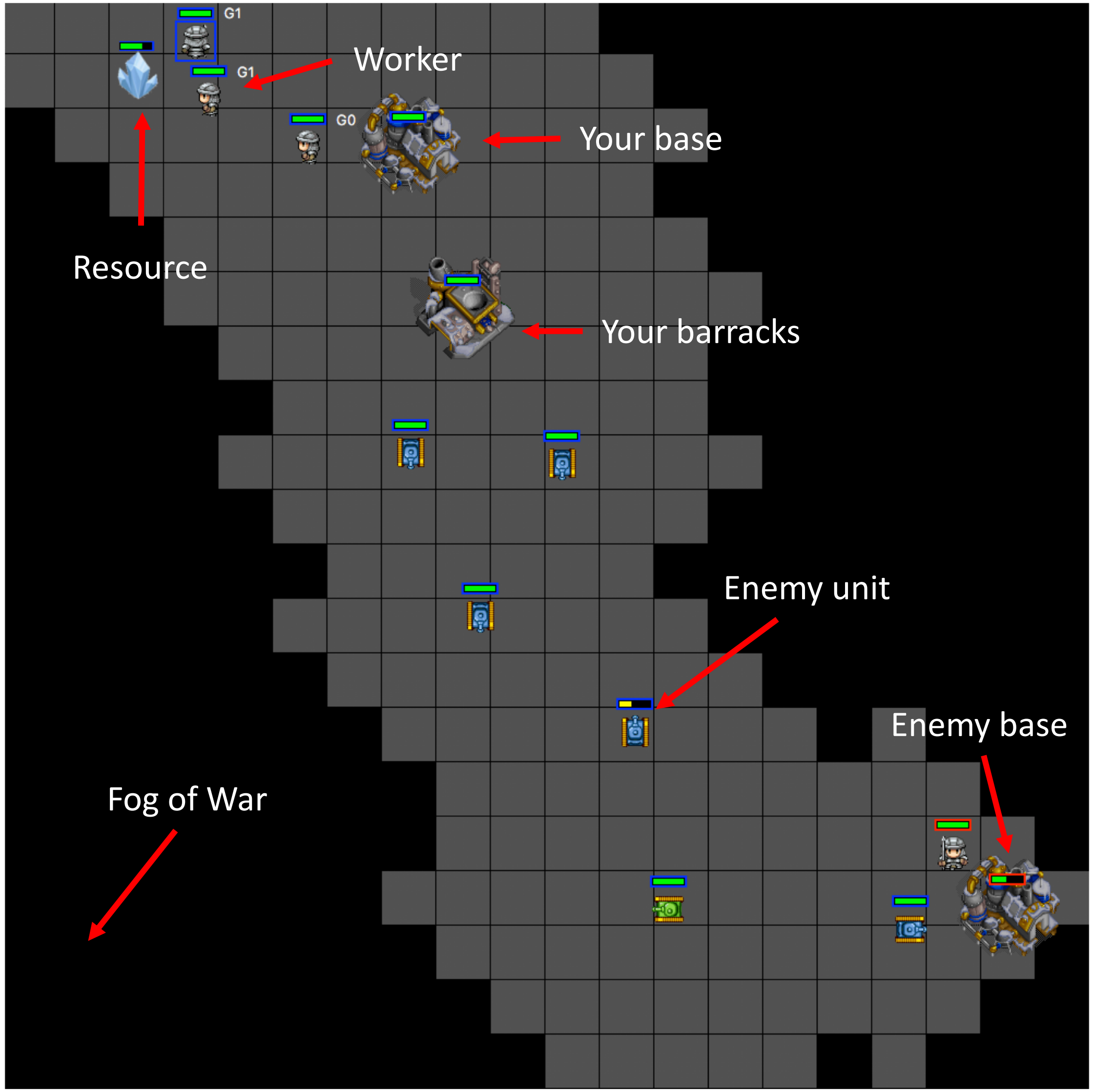


Putting AlphaGoZero and AlphaZero into the same framework

AlphaGoZero (more synchronization)
AlphaZero (less synchronization)

Server controls synchronization
Server also does training.

MiniRTS: A miniature RTS engine



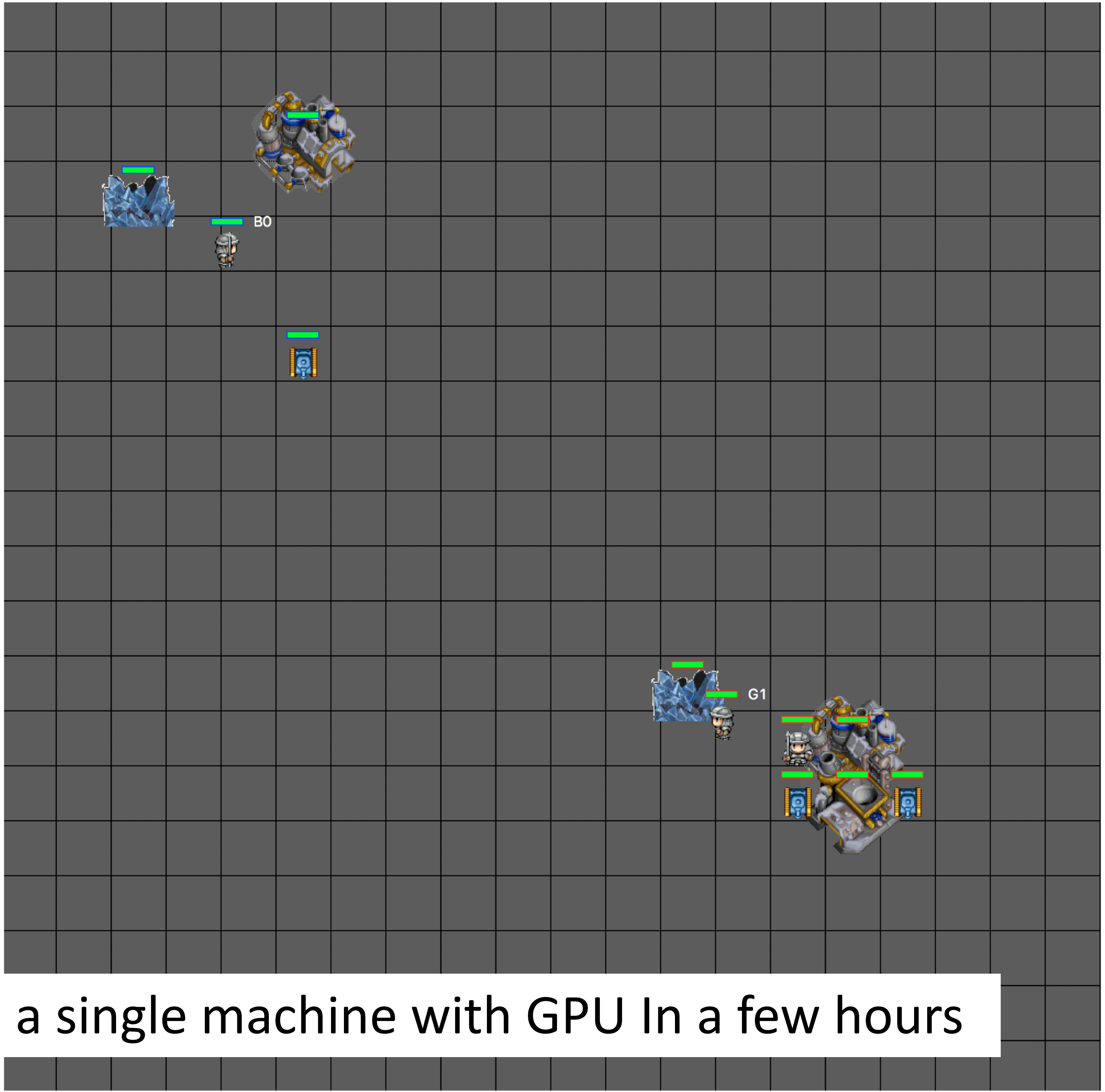
Platform	Frame per second
ALE	6,000
Open AI Universe	60
Malmo	120
DeepMind Lab	287*/866**
VizDoom	7,000
TorchCraft	2,000
MiniRTS	40,000

* Using CPU only　　** Using CPUs and GPU

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.

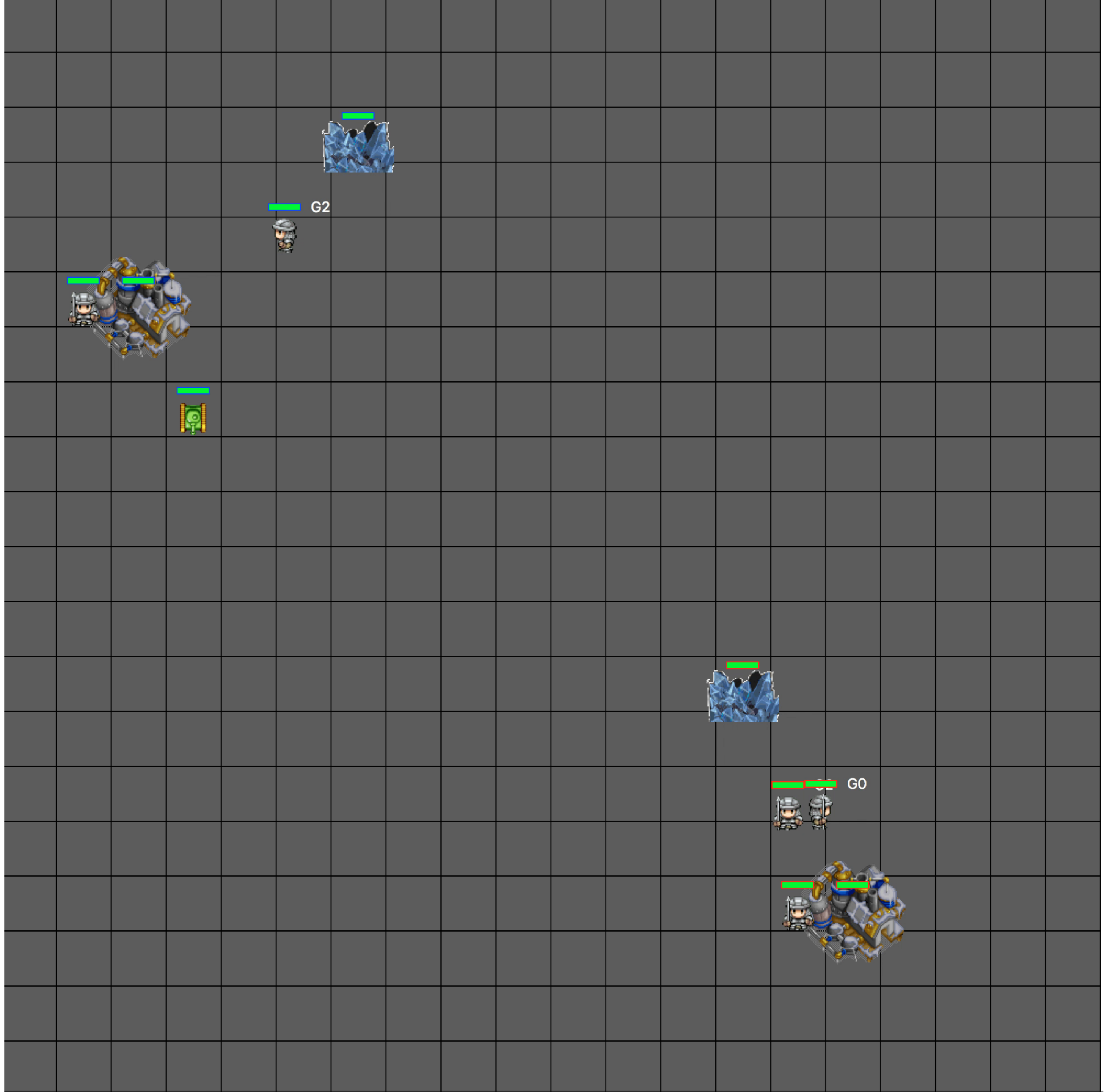
Trained AI



AI_SIMPLE

Trained with a single machine with GPU In a few hours

Trained AI

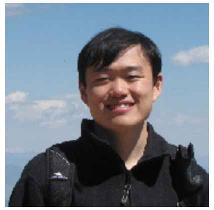


AI_SIMPLE

First Person Shooter (FPS) Game

Yuxin Wu, Yuandong Tian, ICLR 2017

Yuxin Wu



Yuandong Tian

Play the game from the raw image!

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

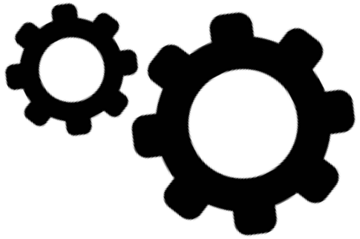
Videos:

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

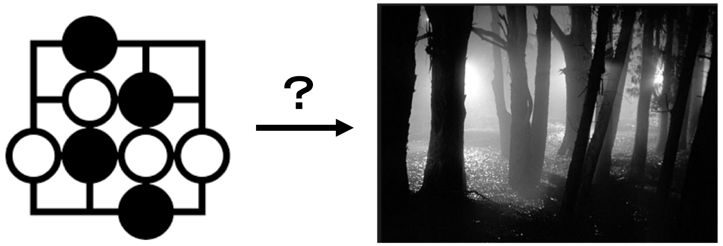
What’s Beyond Game for RL?

Game as a testbed of Reinforcement Learning



Need good simulator

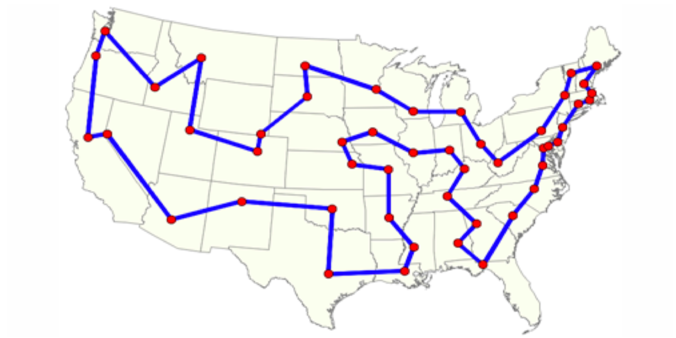
Require a lot of data/resources.



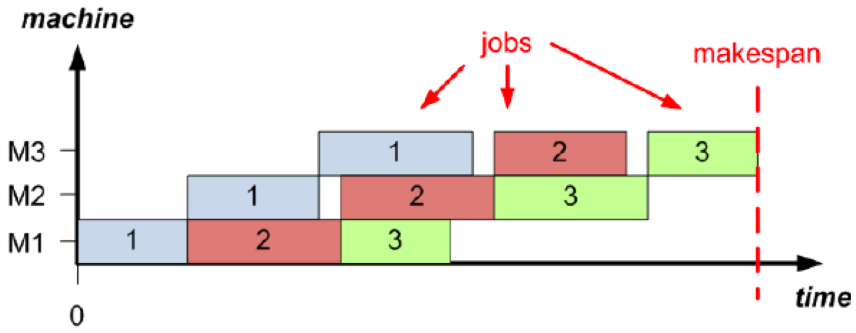
Sim2real issue

Applications?

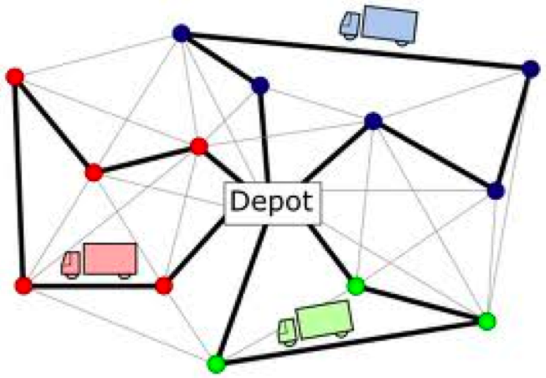
RL for optimization



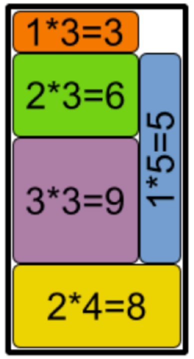
Travel Salesman Problem



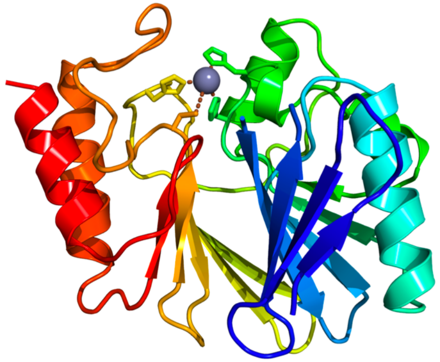
Job Scheduling



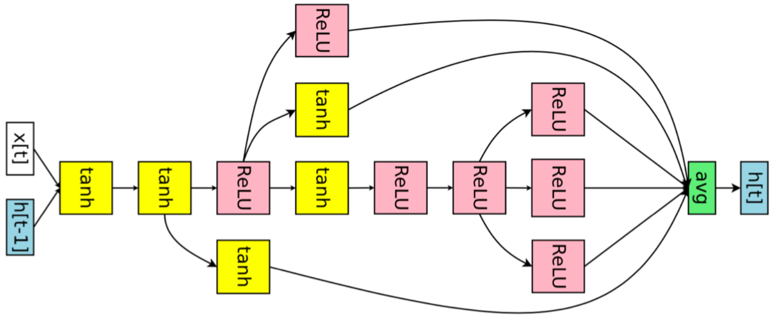
Vehicle Routing



Bin Packing



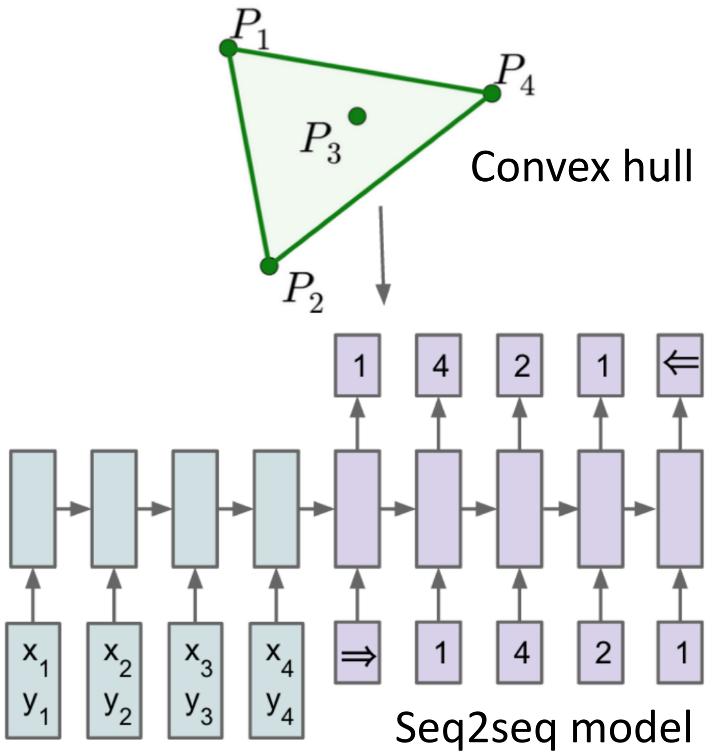
Protein Folding



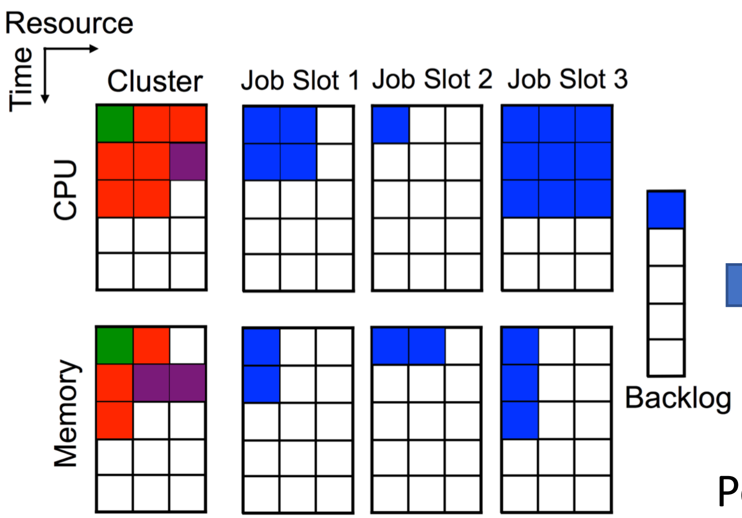
Model-Search

Non-differentiability

- Direct predicting combinatorial solutions.



[O. Vinyals. et al, Pointer Networks, NIPS 2015]

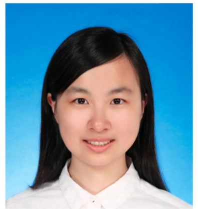
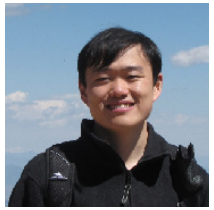


Schedule the job to *i*-th slot

Policy gradient

[H. Mao et al, Resource Management with Deep Reinforcement Learning, ACM Workshop on Hot Topics in Networks, 2016]

Local Rewriting Framework

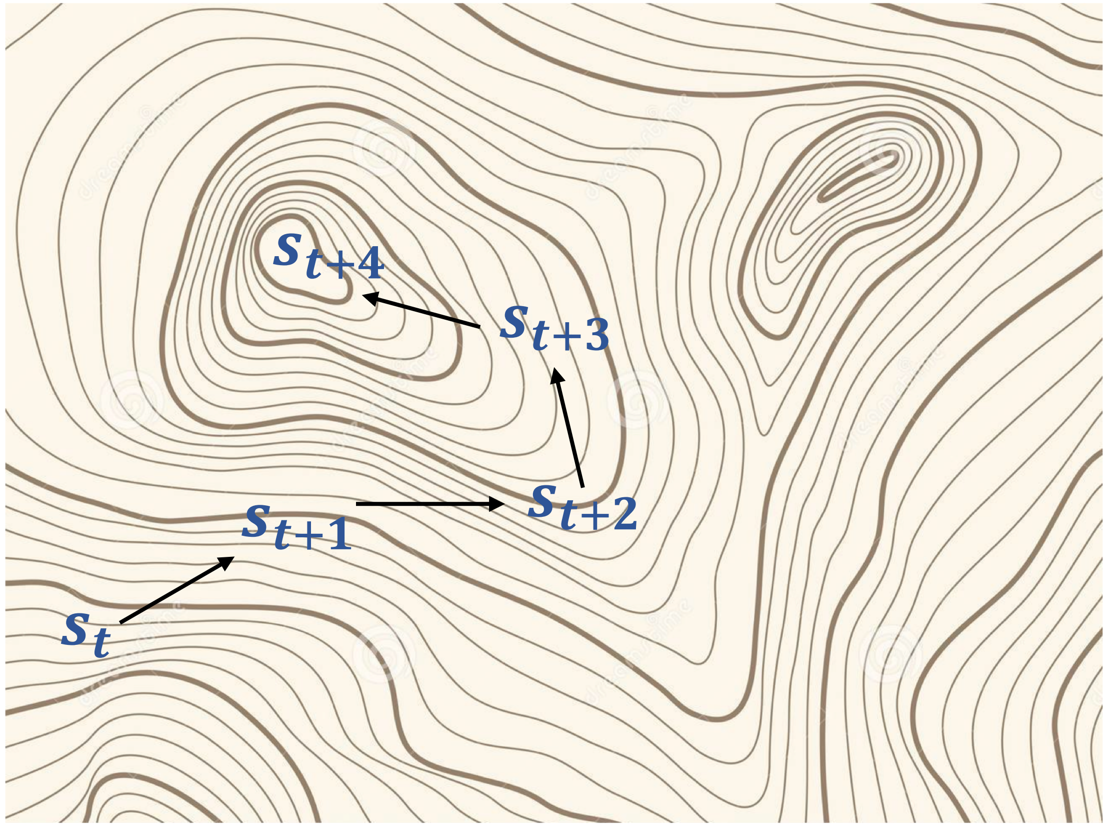



Xinyun Chen Yuandong Tian

A learned "gradient descent" that

starts from a feasible solution
iteratively converges to a good solution

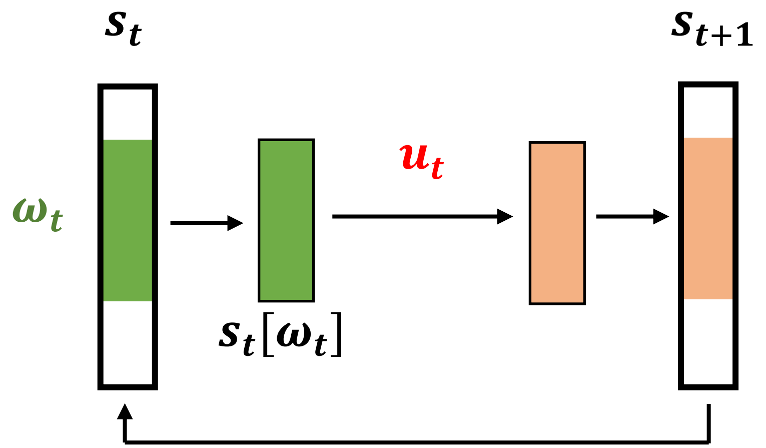
How to learn it?



Local Rewriting Framework

Current State (i.e. Solution) s_t → Region-Picker $\omega_t \sim \pi_\omega(\cdot\,|s_t)$ → Rule-Picker $u_t \sim \pi_u(\cdot\,|s_t\,[\omega_t])$

$s_{t+1} = f(s_t, \omega_t, u_t)$ → s_t



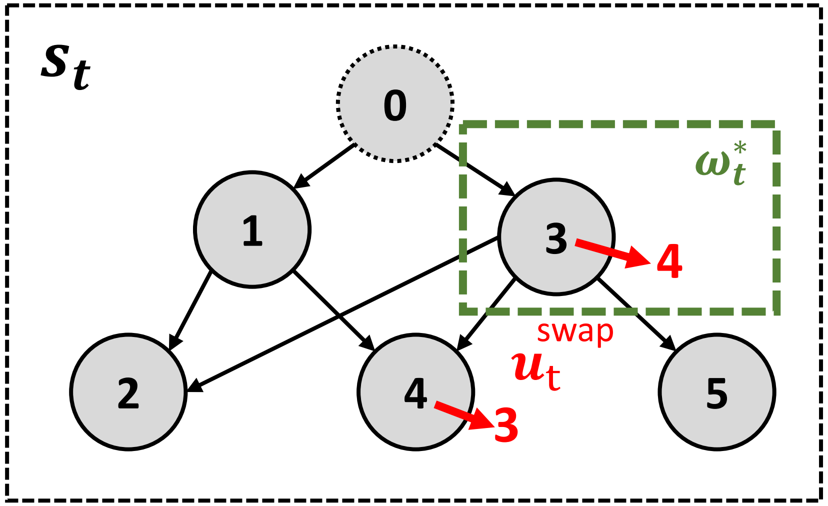
Q-Actor-Critic Training of two policies $\pi_\omega(\cdot\,|s_t)$ and $\pi_u(\cdot\,|s_t\,[\omega_t])$

$\pi_\omega(\cdot\,|s_t)$: Q-learning with soft policy $\pi_\omega(\omega_t|s_t;\theta) = \frac{\exp(Q(s_t,\omega_t;\theta))}{\sum_{\omega_t}\exp(Q(s_t,\omega_t;\theta))}$

$\pi_u(\cdot\,|s_t\,[\omega_t])$: Actor-Critic with learned Q $L_u(\phi) = -\sum_{t=0}^{T-1}\Delta(s_t,(\omega_t,u_t))\log\pi_u(u_t|s_t[\omega_t];\phi)$

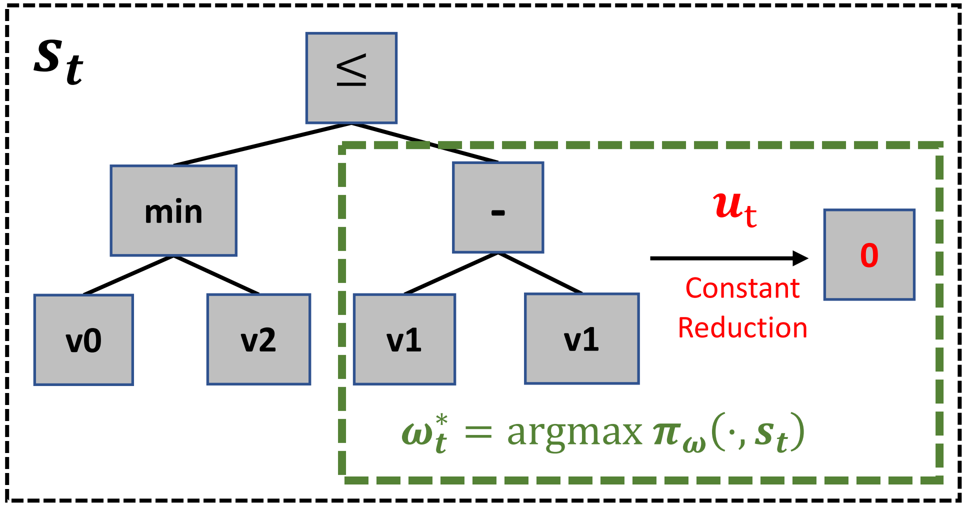
Applications

(a)



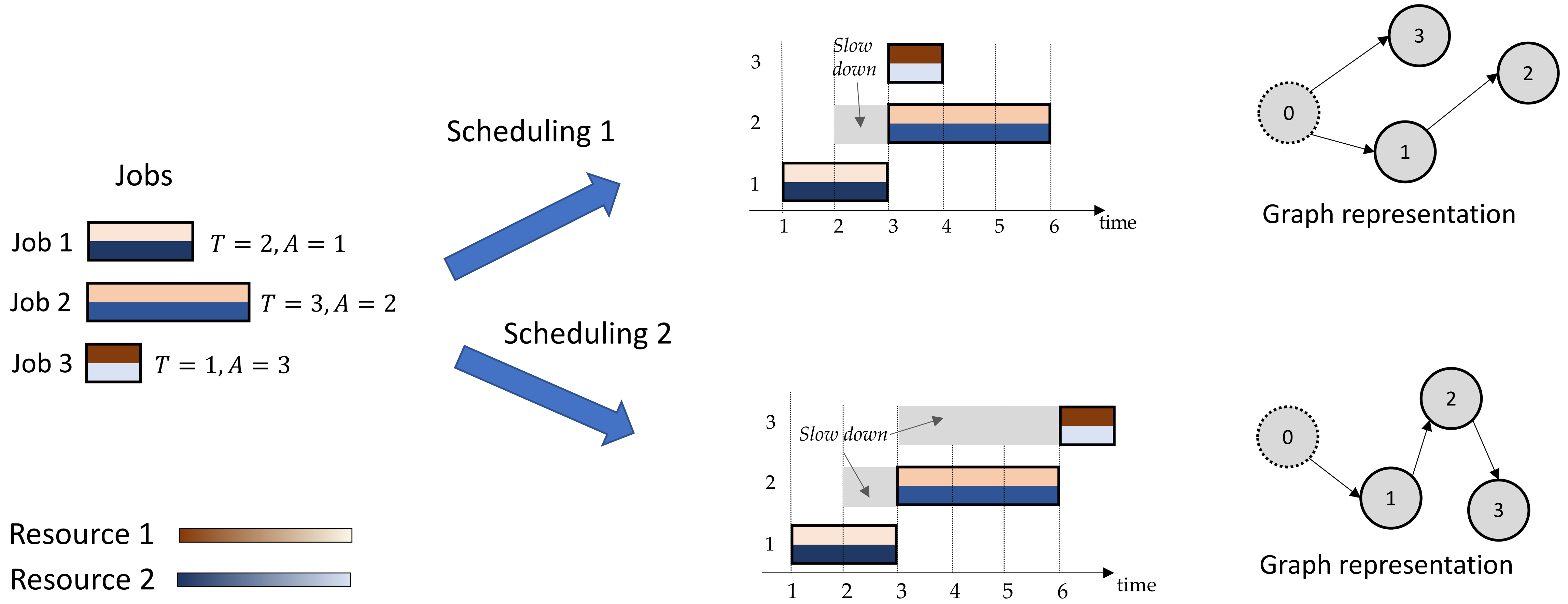
Online Job Scheduling

(b)

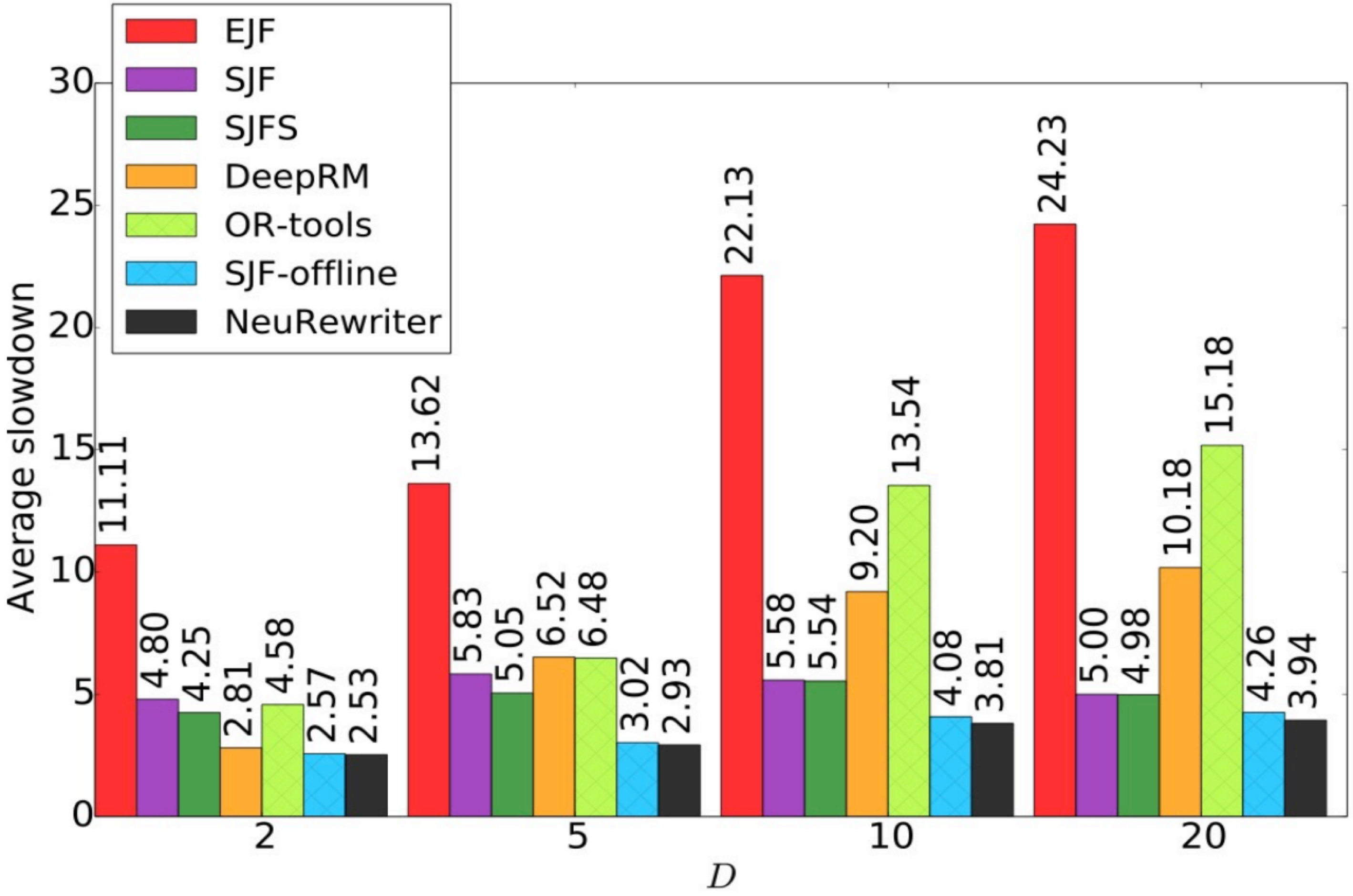


Expression Simplification

Online Job Scheduling



Online Job Scheduling



D: Number of resources

Baselines:

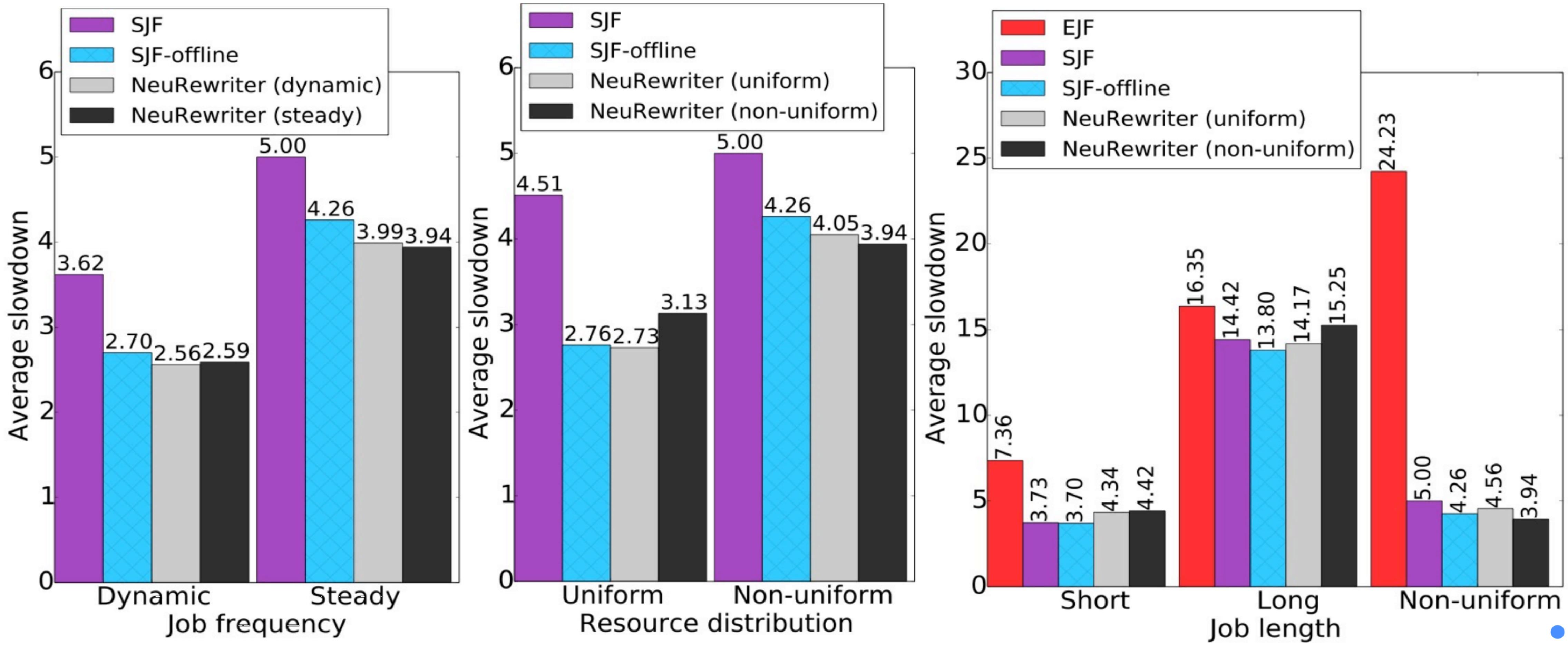
- Earliest Job First (EJF)
- Shortest Job First (SJF)
- Shortest First Search (SJFS)
- DeepRM

Offline baselines:

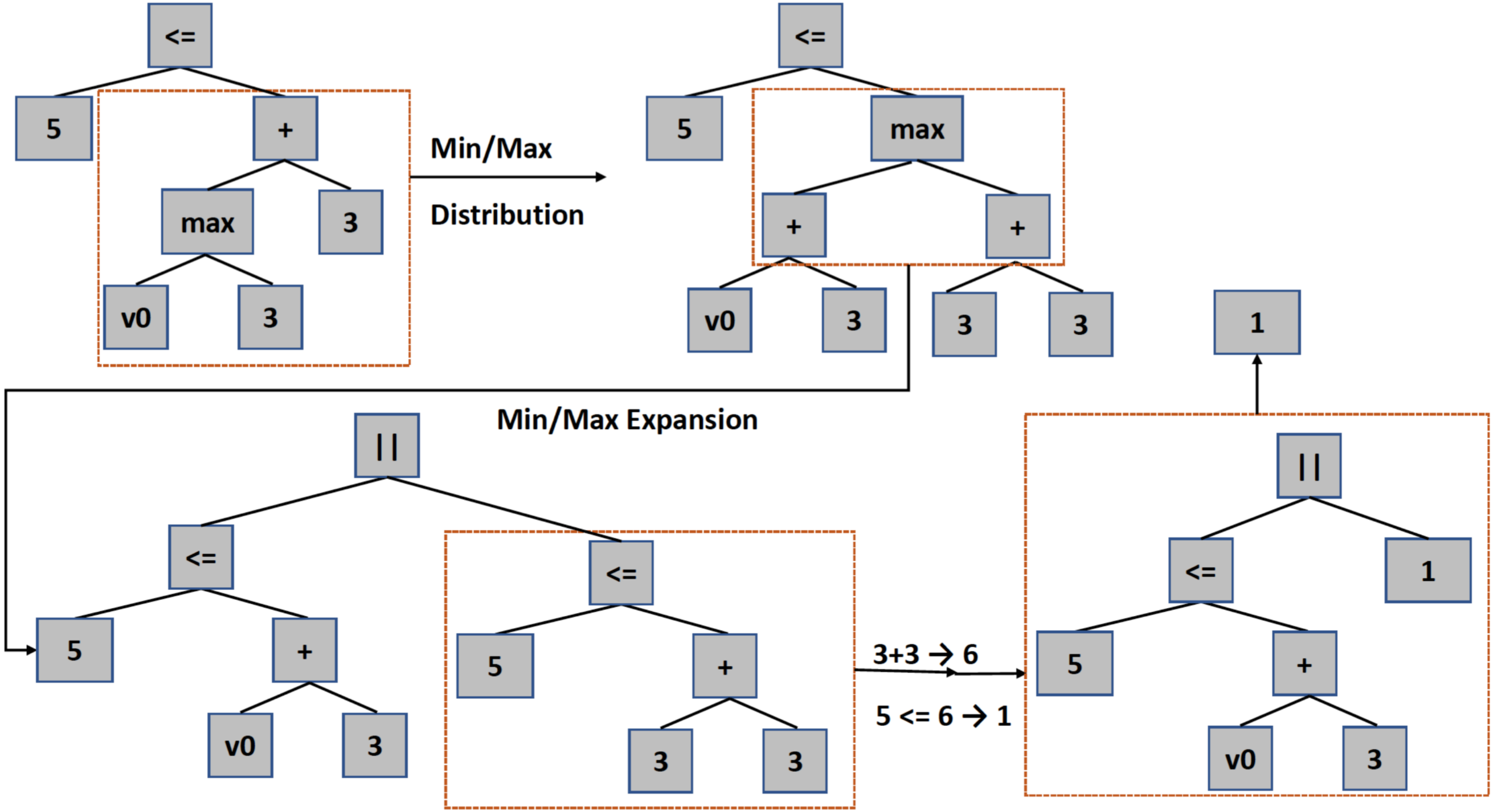
- Google OR-tools (OR-tools)
- SJF-offline

Online Job Scheduling: Ablation Study

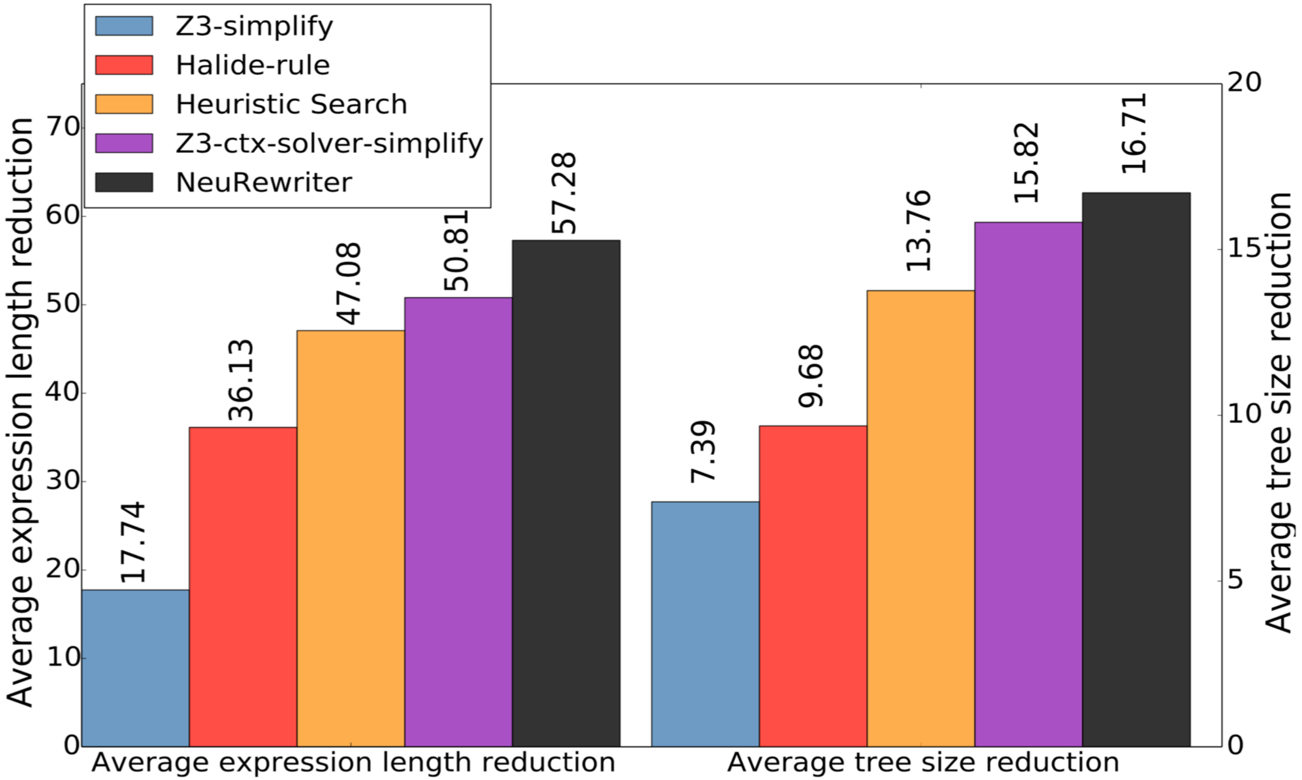
The learned model can generalize to different job distributions.



Expression Simplification



Expression Simplification



Baselines:

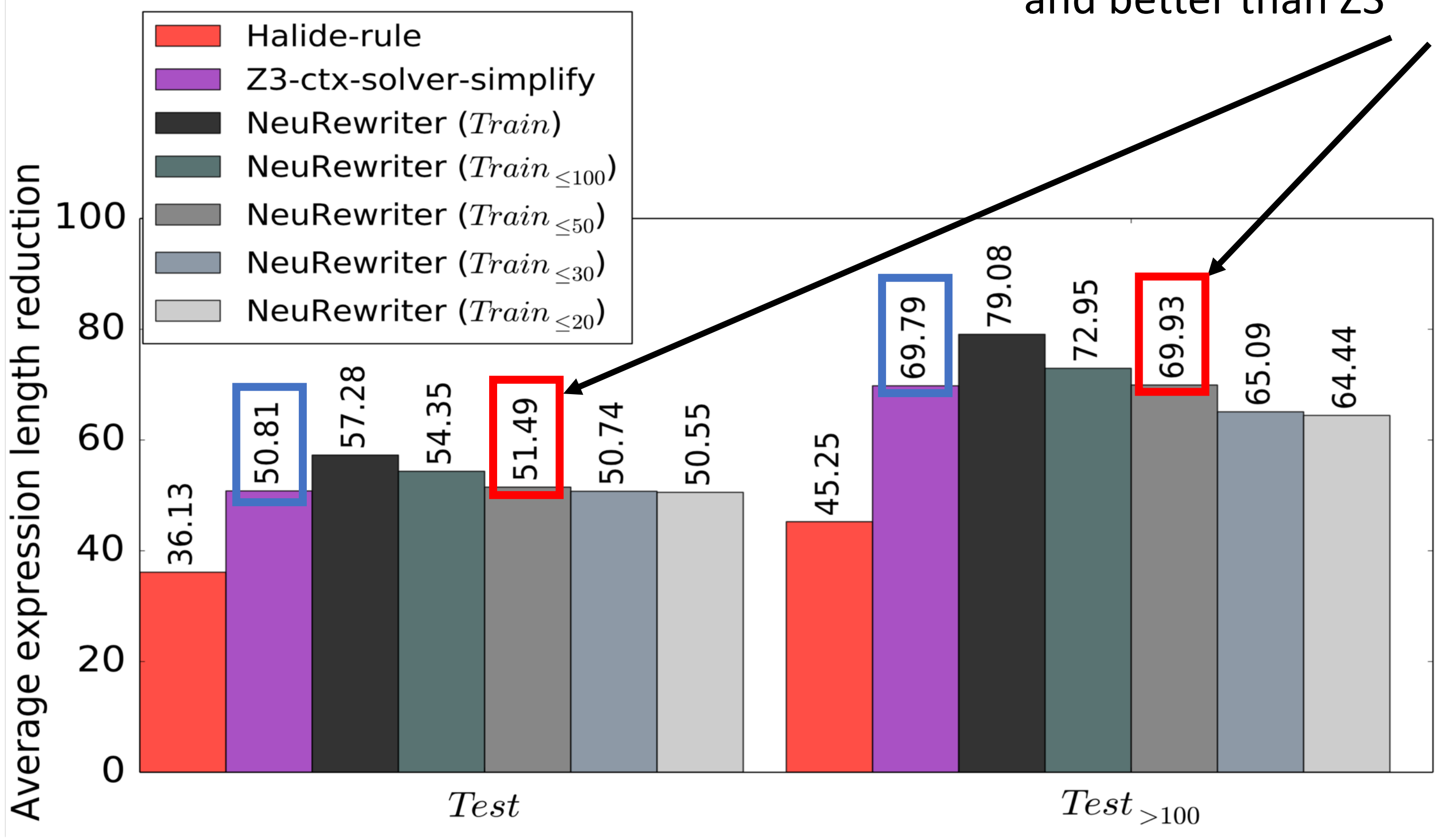
Z3-simplify
Z3-ctx-solver-simplify
Heuristic Search
Halide rules

Z3 is a state-of-the-art theorem prover.

Expression Simplification

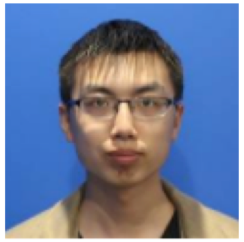
Transfer learning still works well.

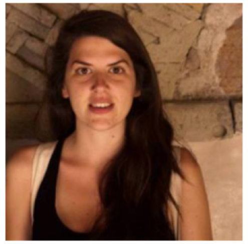
A model trained with expression length ≤ 50 has good performance on test set with expression length ≥ 100, and better than Z3

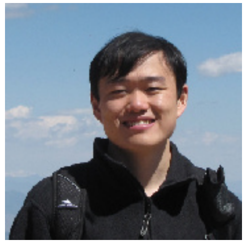


Navigation:
Build a good model


Yi Wu


Yuxin Wu

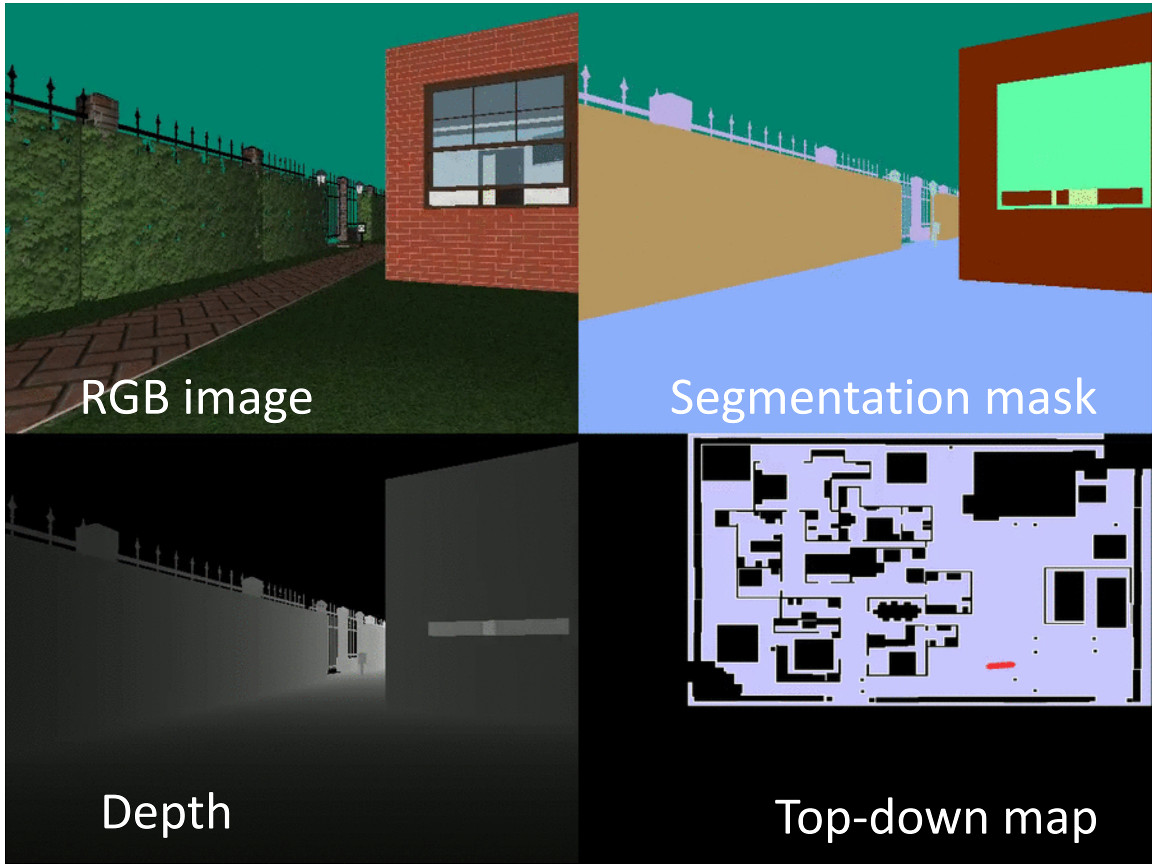

Georgia Gkioxari


Yuandong Tian

How to plan the trajectory
in **unknown** environments?

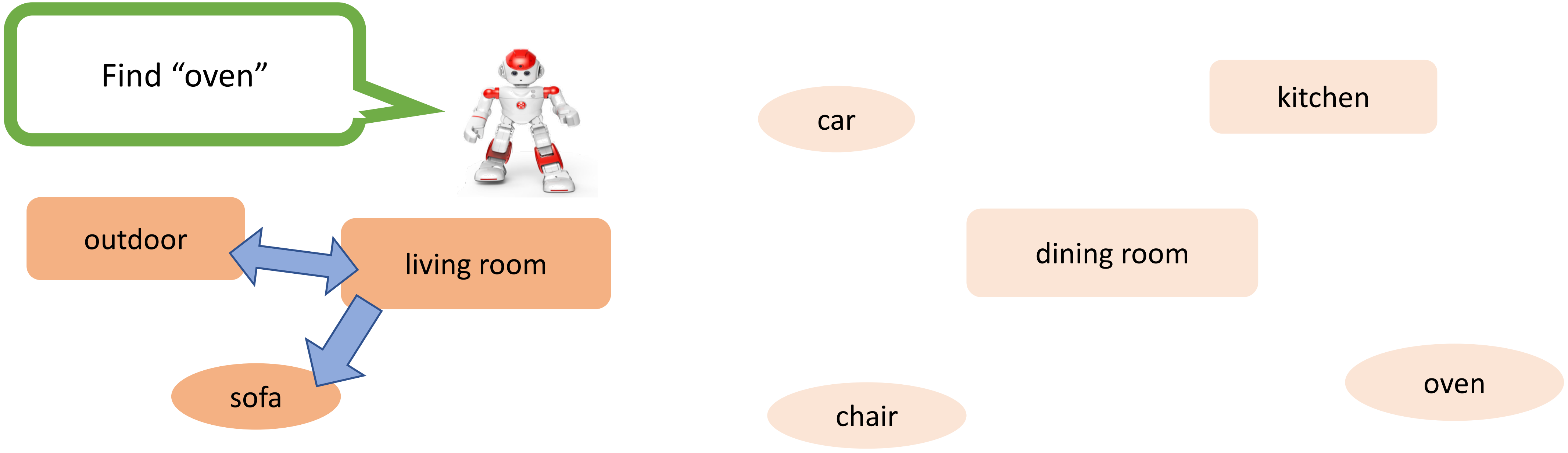
House3D

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



https://github.com/facebookresearch/House3D

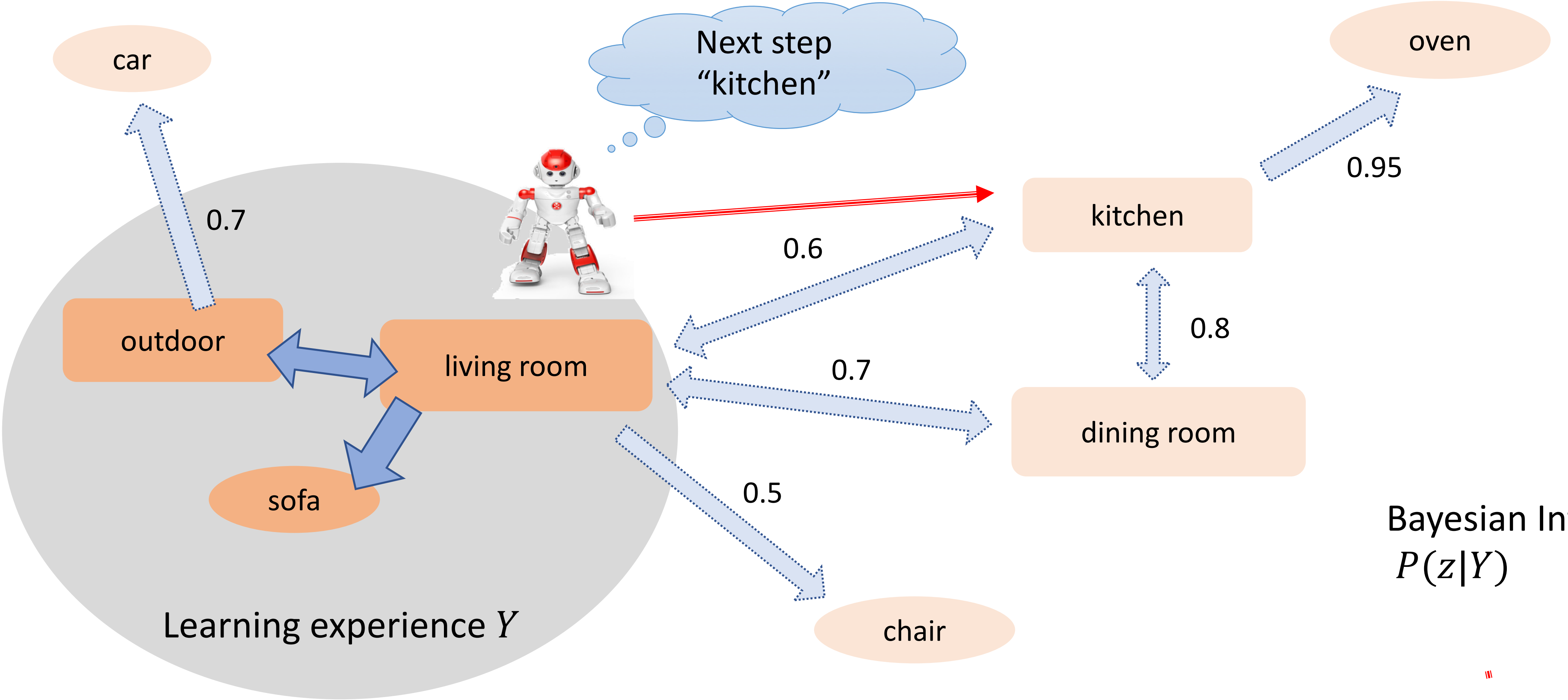
Build a semantic model



Incomplete model of the environment

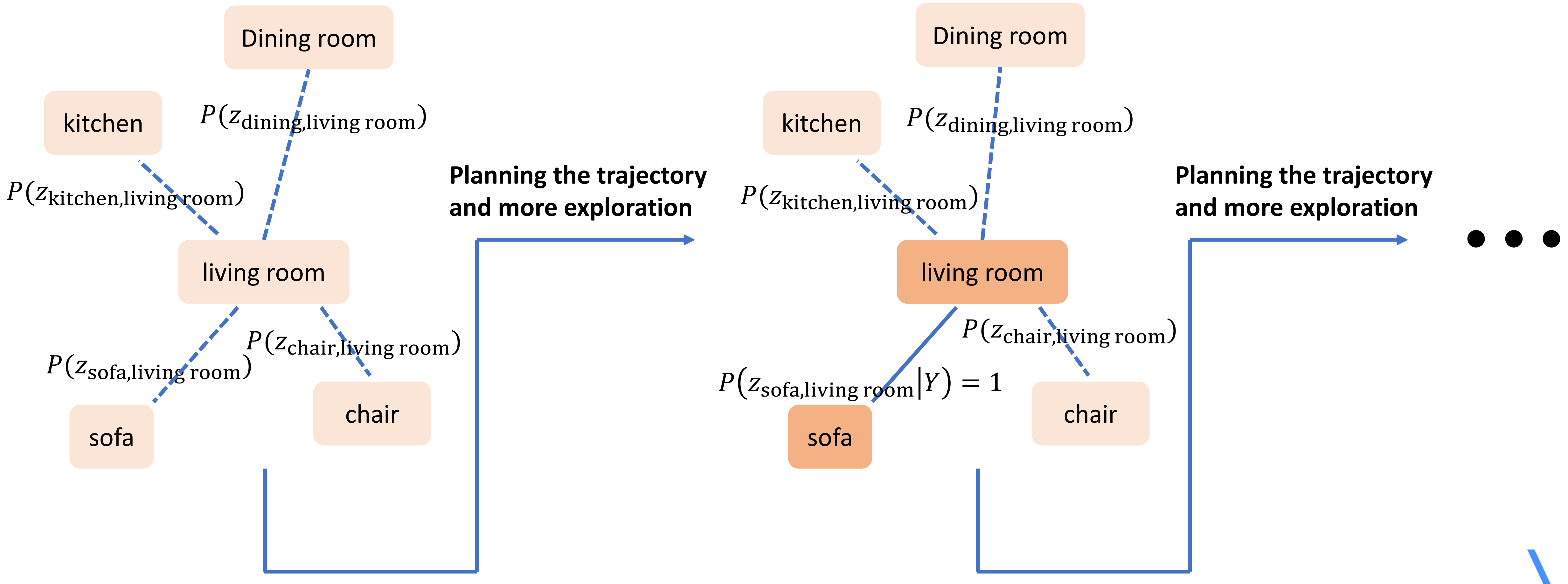
[Y. Wu et al, Learning and Planning with a Semantic Model, submitted to ICLR 2019]

Build a semantic model

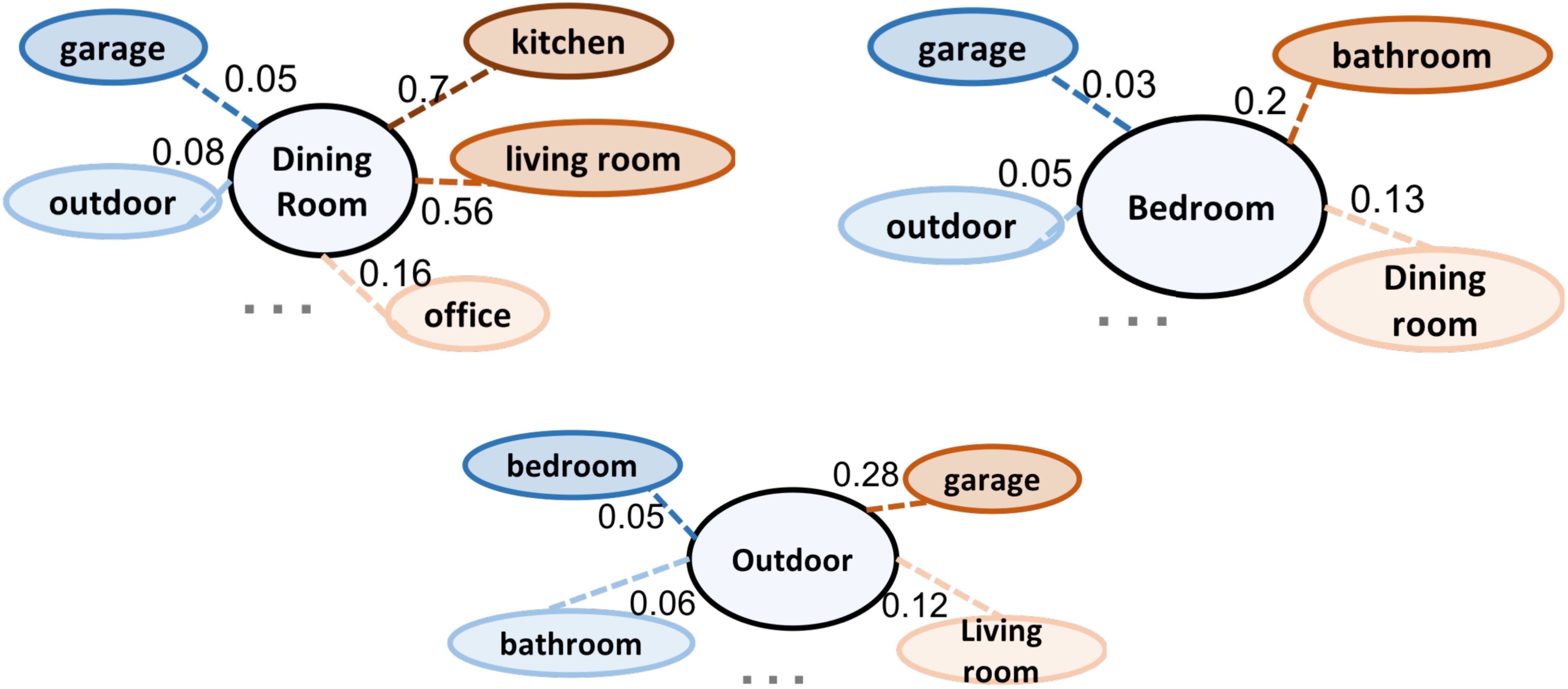


Bayesian Inference
$P(z|Y)$

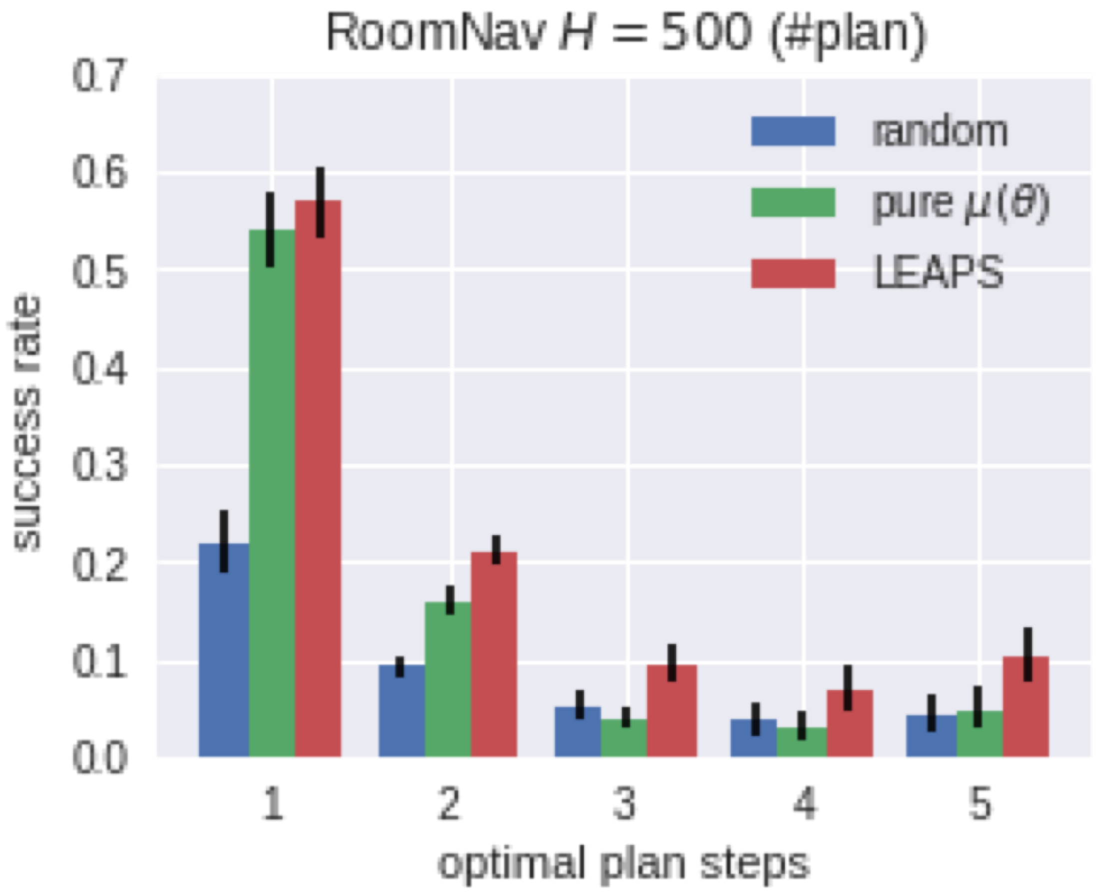
LEAPS
LEArning and Planning with a Semantic model

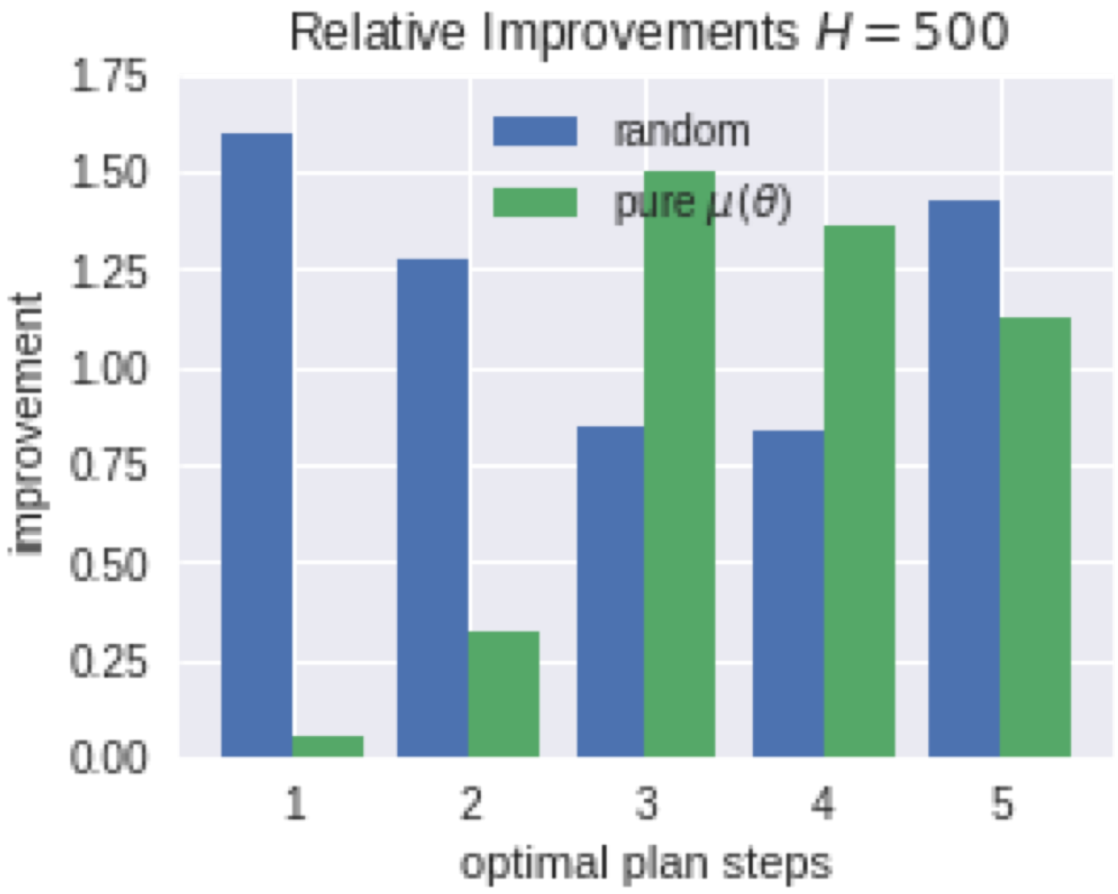


Learning the Prior between Different Rooms



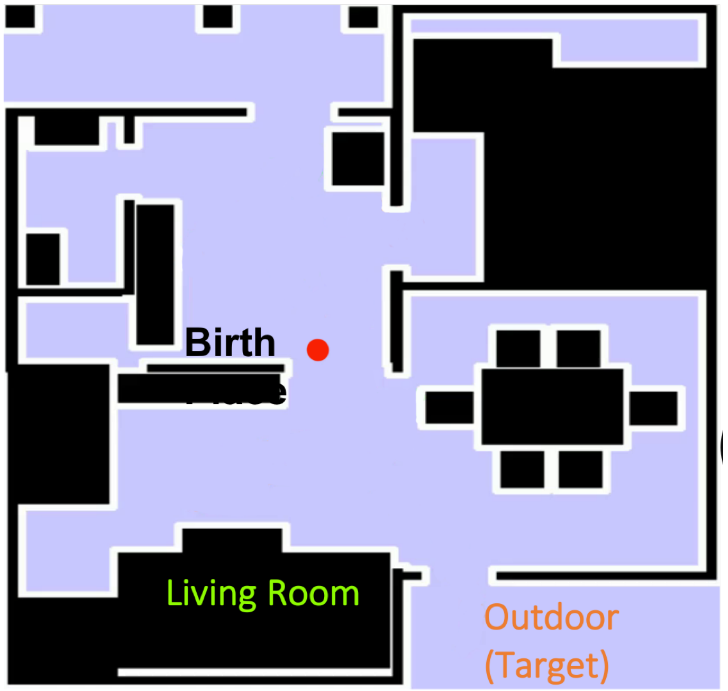
Test Performance on ConceptNav

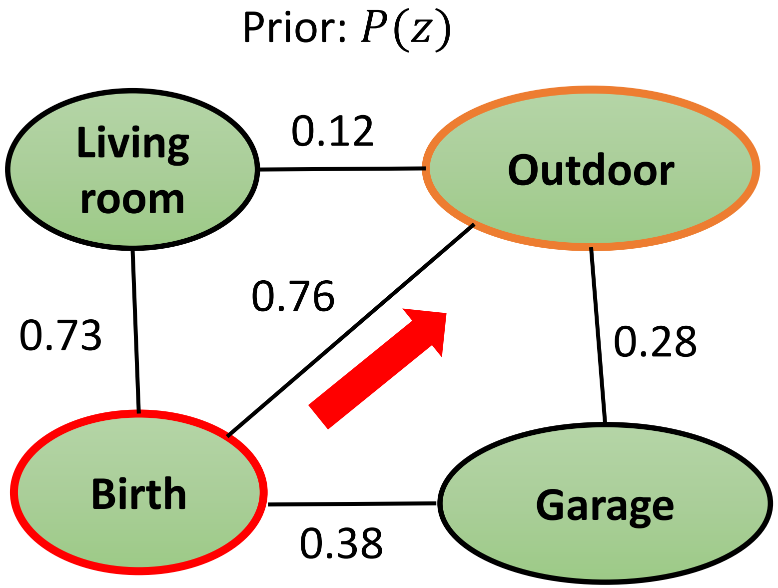




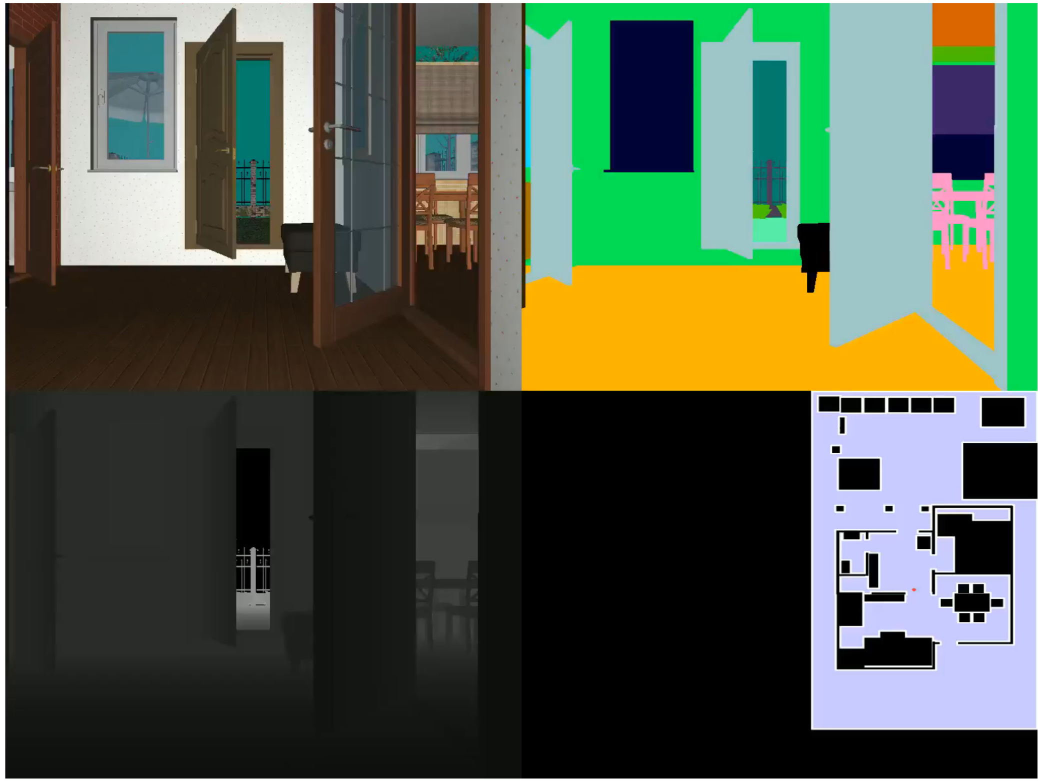
Case Study

- A case study
  - **Go to “outdoor”**



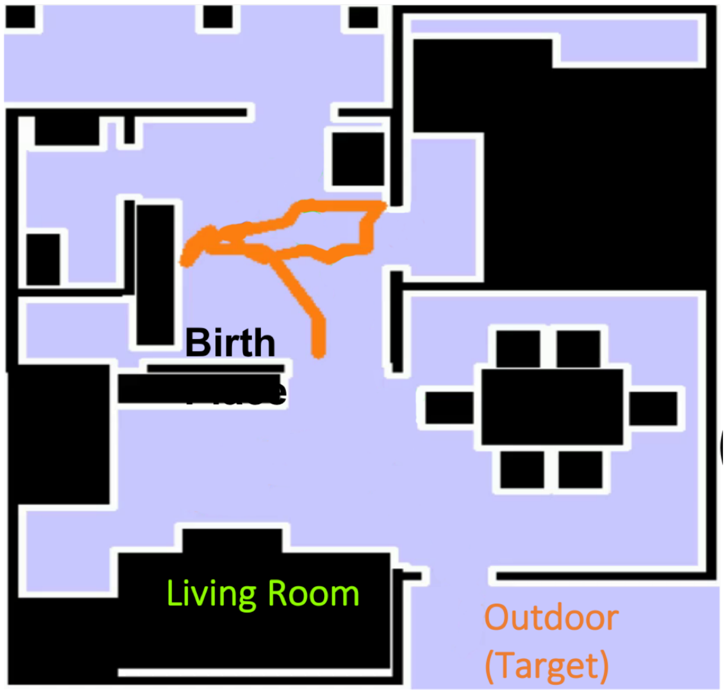


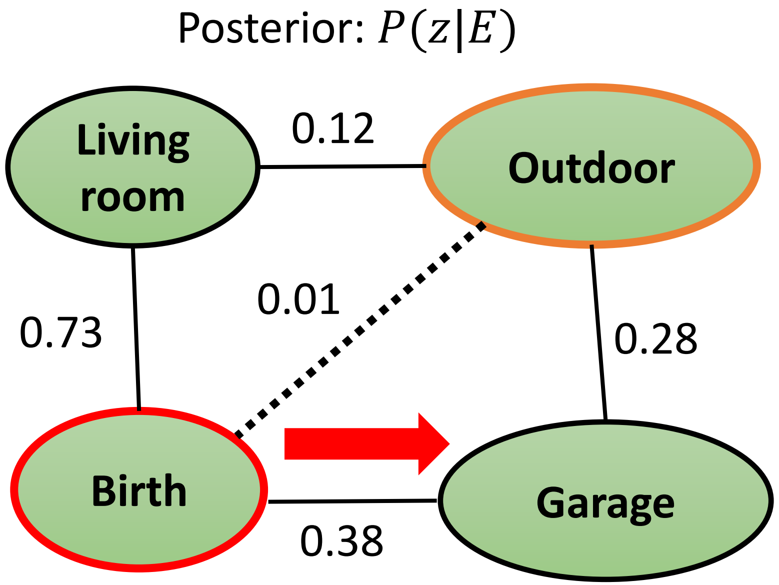
Sub-Goal: Outdoor



Case Study

- A case study
  - **Go to "outdoor"**

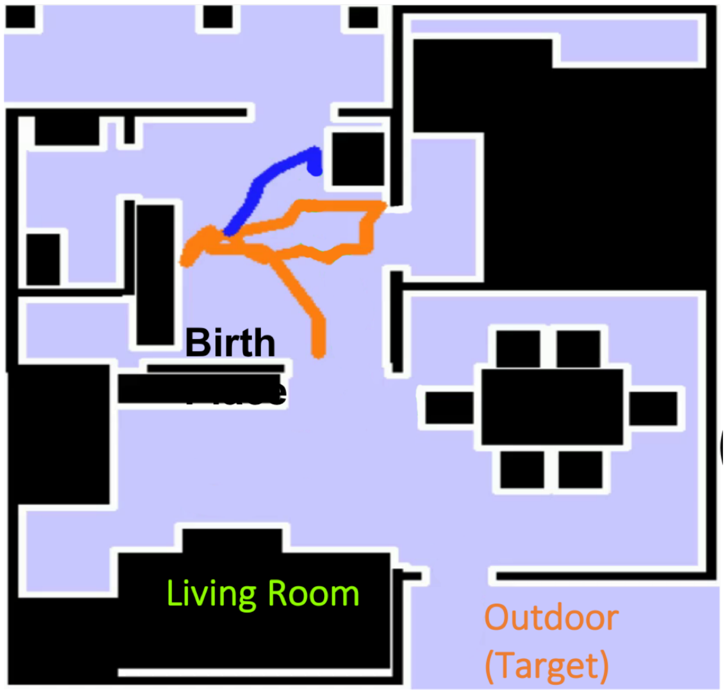




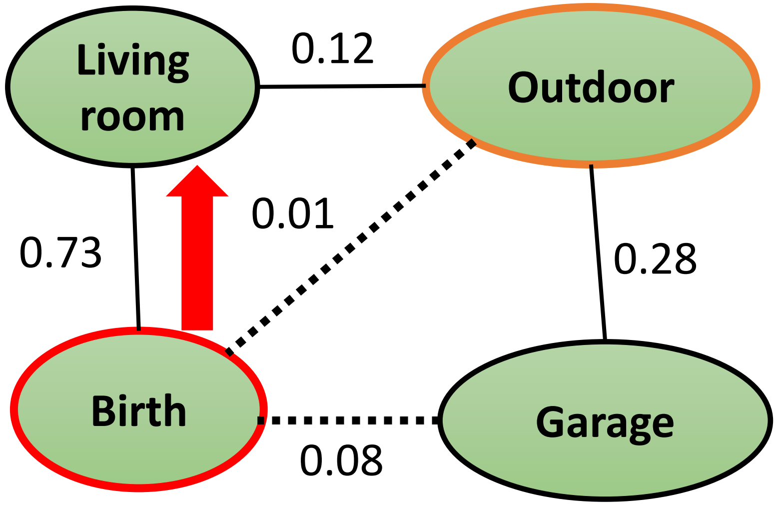
Sub-Goal: Outdoor

Case Study

- A case study
  - Go to “outdoor”



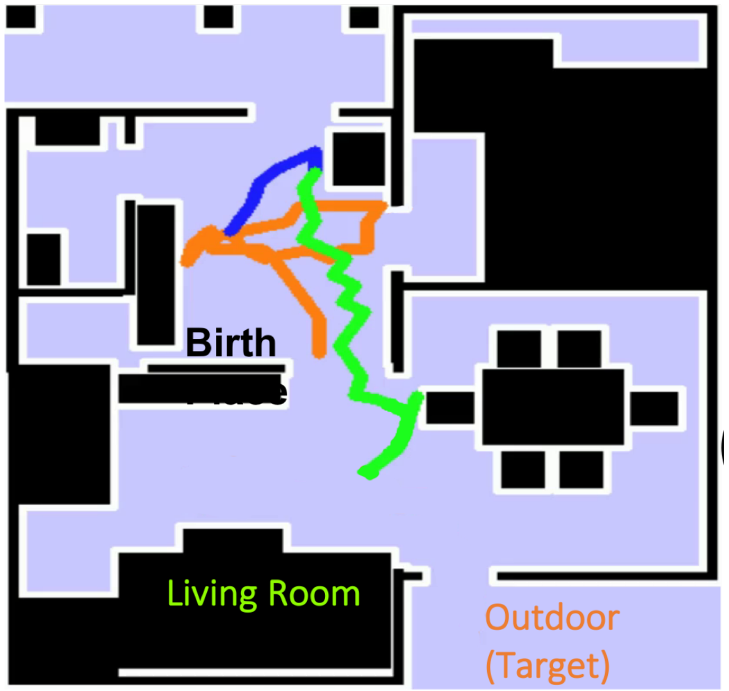
Posterior: $P(z|E)$

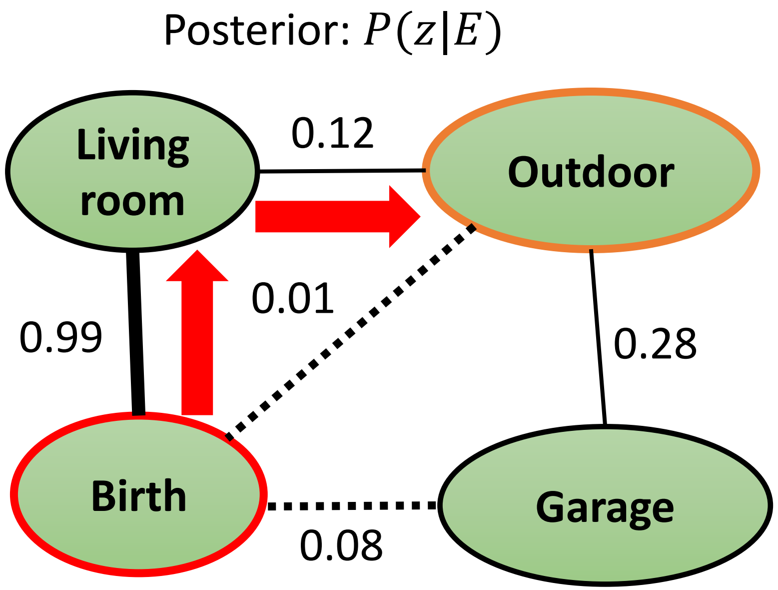


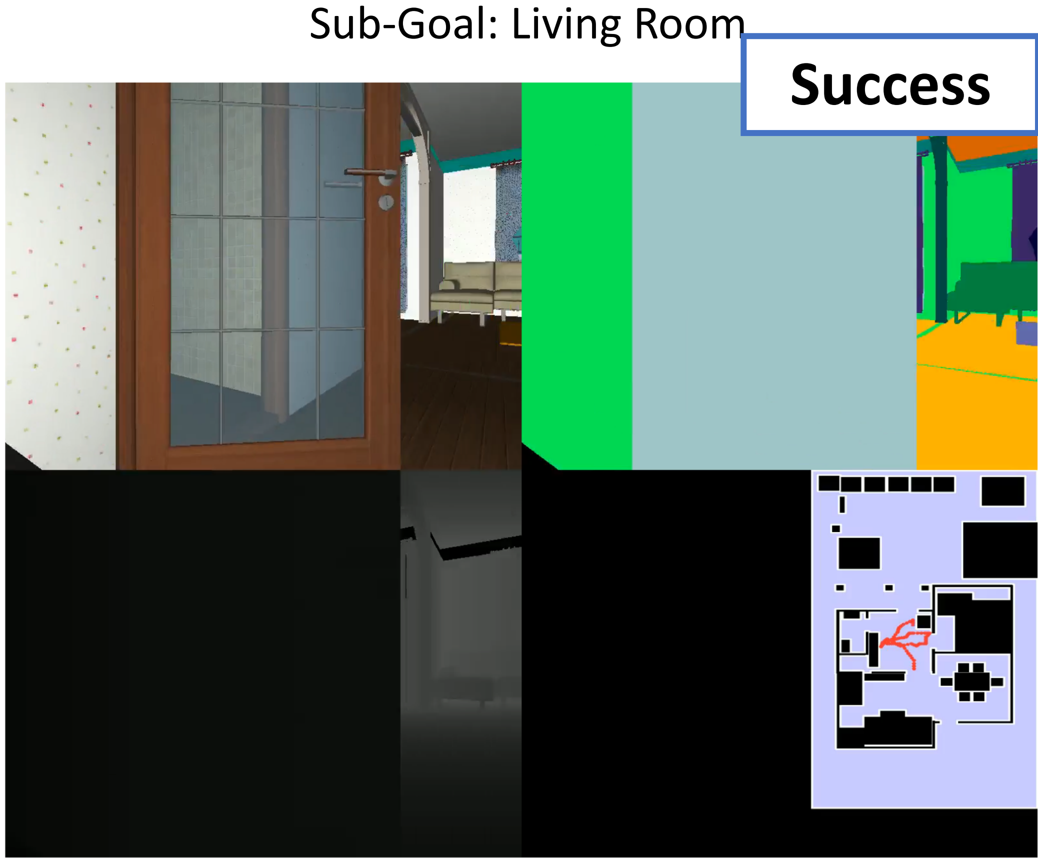
Sub-Goal: Garage

Case Study

- A case study
  - Go to “outdoor”

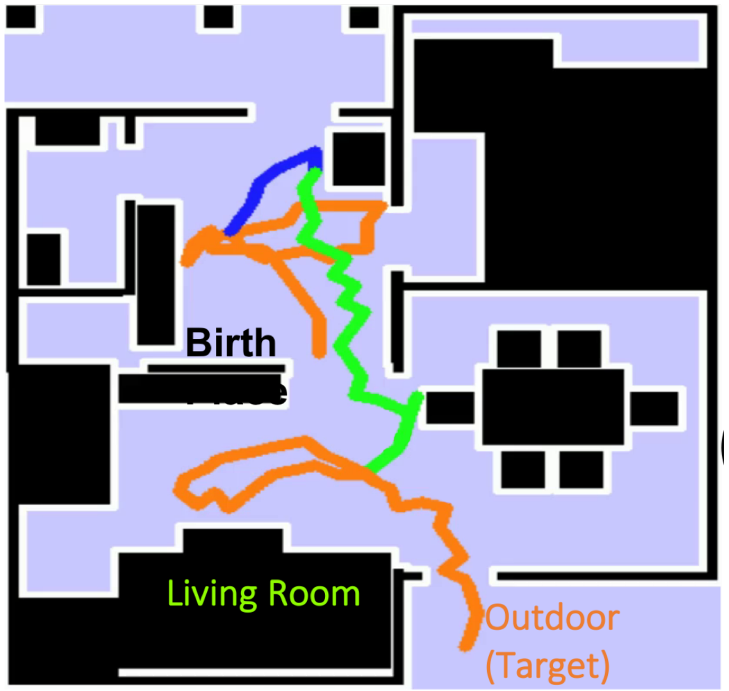


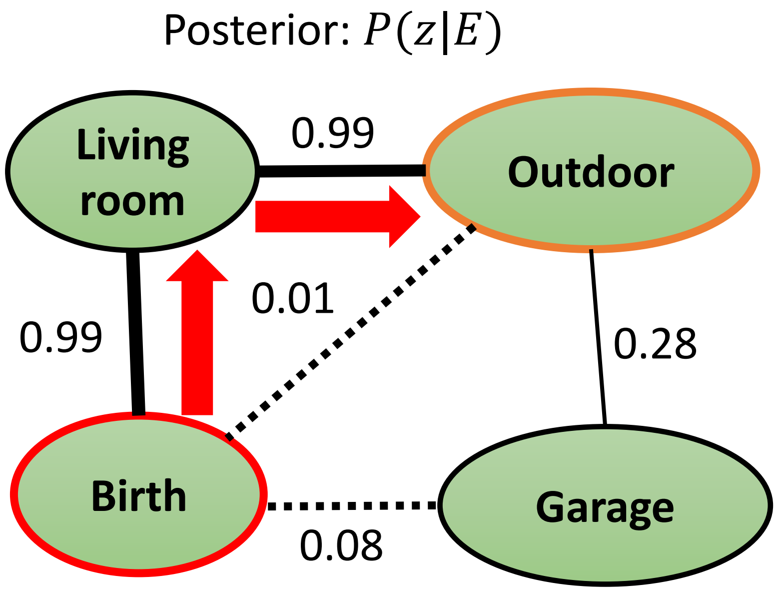


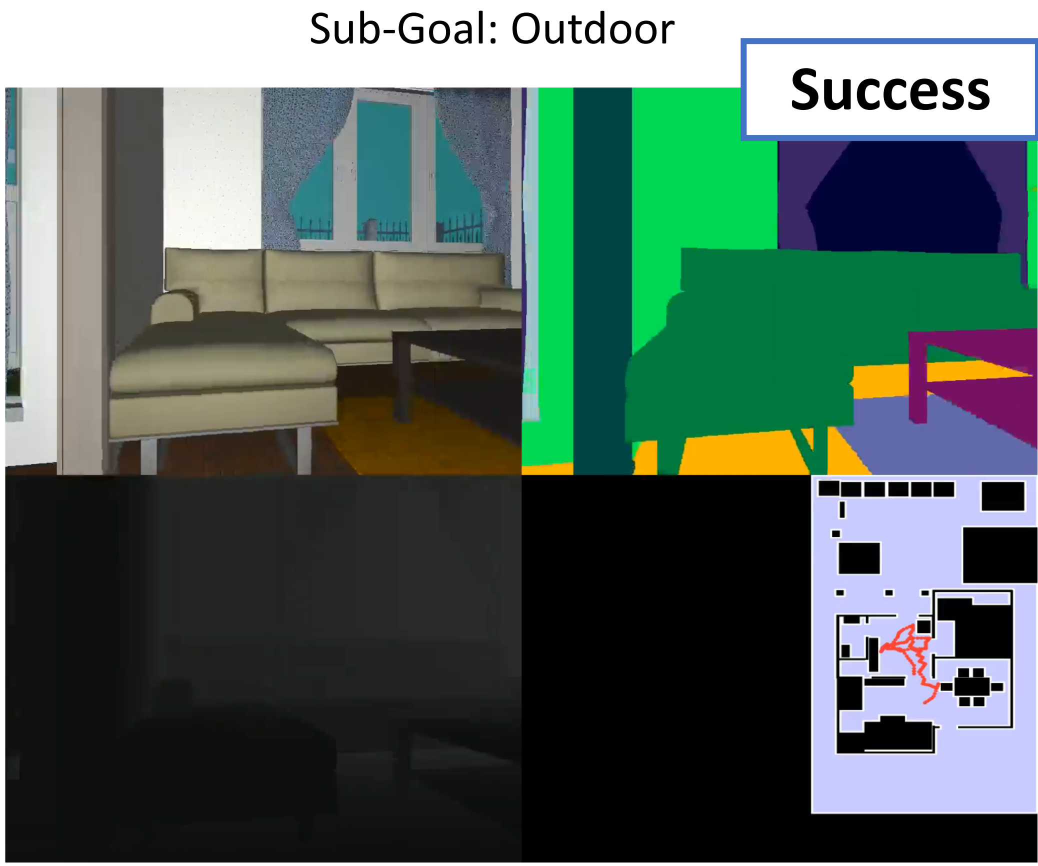


Case Study

- A case study
  - Go to "outdoor"





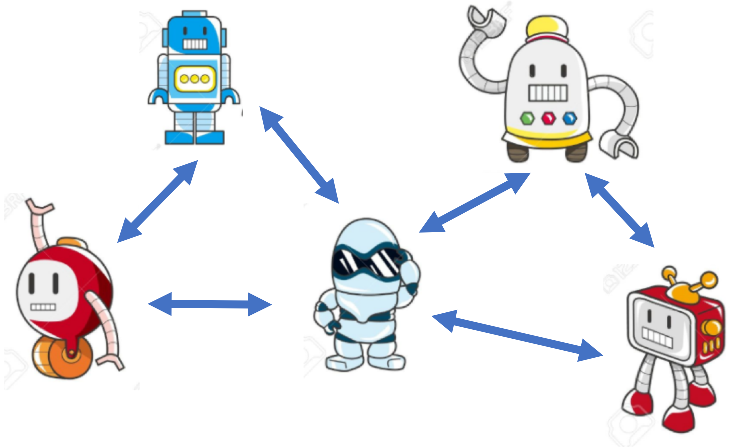


opt. plan-steps	1	2	3	4	5	overall
Horizon $H = 300$						
random	20.5 / 15.9	6.9 / 16.7	3.8 / 10.7	1.6 / 4.2	3.0 / 8.8	7.2 / 13.6
pure $\mu(\theta)$	49.4 / 47.6	11.8 / 27.6	2.0 / 4.8	2.6 / **10.8**	4.2 / 13.2	13.1 / 22.9
aug.$\mu_S(\theta_s)$	47.8 / 45.3	11.4 / 23.1	3.0 / 7.8	3.4 / 8.1	4.4 / 11.2	13.0 / 20.5
RNN control.	52.7 / 45.2	13.6 / 23.6	3.4 / 9.6	3.4 / 10.2	6.0 / 17.6	14.9 / 21.9
LEAPS	**53.4 / 58.4**	**15.6 / 31.5**	**4.5 / 12.5**	**3.6** / 6.6	**7.0 / 18.0**	**16.4 / 27.9**
Horizon $H = 500$						
random	21.9 / 16.9	9.3 / 18.3	5.2 / 12.1	3.6 / 6.1	4.2 / 9.9	9.1 / 15.1
pure $\mu(\theta)$	54.0 / 57.5	15.9 / 25.6	3.8 / 7.7	2.8 / 6.4	4.8 / 8.6	16.2 / 22.9
aug.$\mu_S(\theta_s)$	54.1 / 51.8	15.5 / 26.5	4.6 / 8.2	3.0 / **11.8**	4.6 / 12.5	16.1 / 23.5
RNN control.	**57.4** / 43.8	20.2 / 28.0	7.2 / 14.6	4.2 / 8.0	9.0 / 16.0	19.9 / 24.6
LEAPS	57.2 / **61.9**	**21.5 / 34.4**	**10.0 / 14.8**	**6.4** / 11.6	**12.0 / 23.5**	**21.6 / 31.1**
Horizon $H = 1000$						
random	24.3 / 17.6	13.5 / 20.3	9.1 / 14.3	8.0 / 9.3	7.0 / 11.5	13.0 / 17.0
pure $\mu(\theta)$	60.8 / **58.4**	23.3 / 29.5	7.6 / 8.8	8.2 / 12.9	11.0 / 17.2	22.5 / 26.5
aug.$\mu_S(\theta_s)$	61.3 / 50.1	23.0 / 26.2	9.4 / 12.0	5.8 / 9.6	9.0 / 13.6	22.4 / 23.8
RNN control.	**66.7** / 49.0	30.1 / 31.5	13.8 / 15.4	9.0 / 10.0	14.0 / 20.8	28.2 / 27.7
LEAPS	66.4 / **58.4**	**31.9 / 40.5**	**15.0 / 18.3**	**11.4 / 17.0**	**15.4 / 27.1**	**29.7 / 35.2**

Future Directions

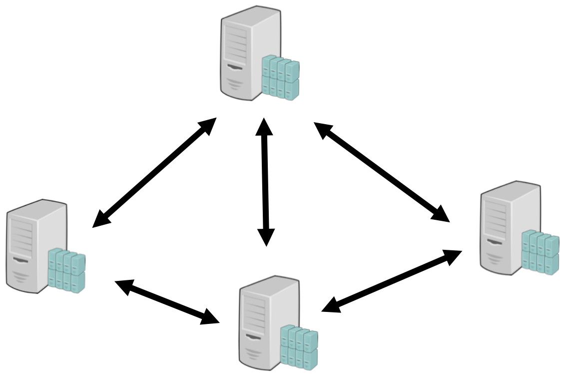
Admiral (General) Captain Lieutenant

Hierarchical RL

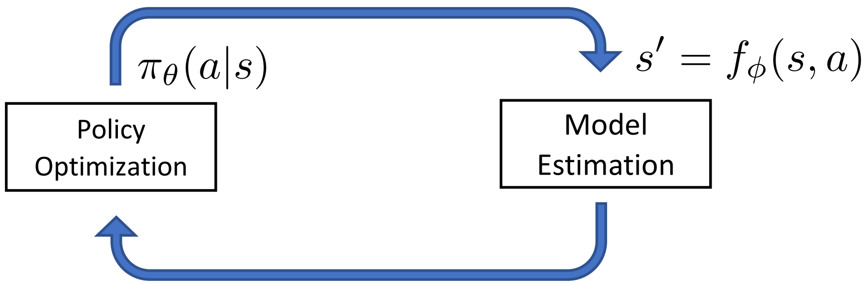


Multi-Agent

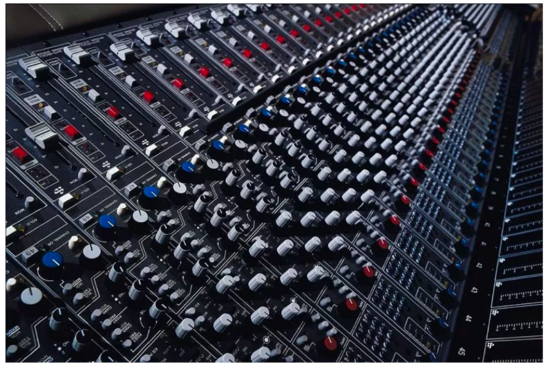
RL applications



RL Systems



Model-based RL



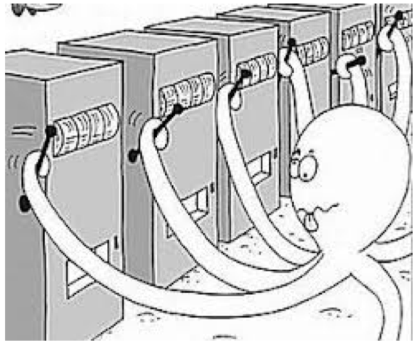
RL for Optimization

How to do well in Reinforcement Learning?

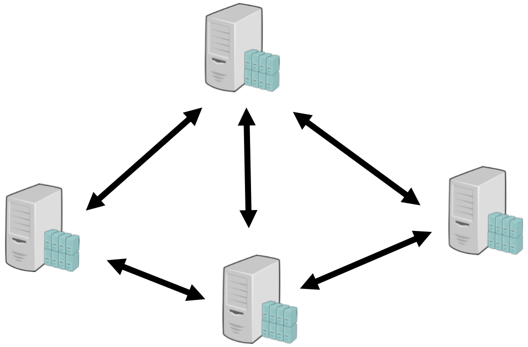
$Q(s,a)$ $V^{\pi}(s)$

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

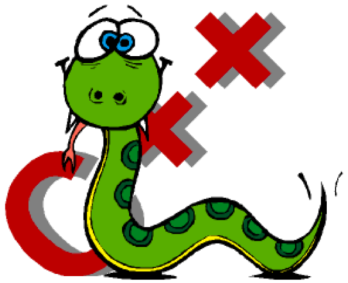
Strong math skills



Parameter tuning skills



Experience on
(distributed) systems



Strong coding skills

Thanks!