Hierarchical Reinforcement Learning in Minecraft

Francois Rossouw

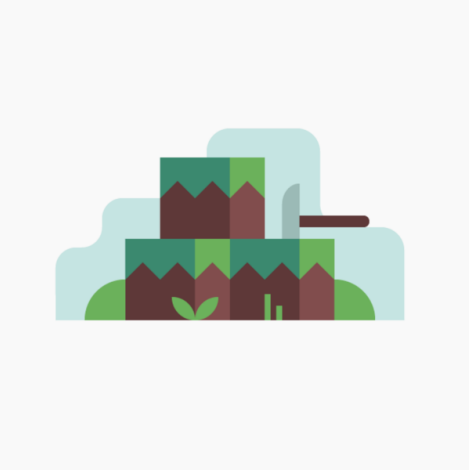
Supervisor: Prof. Herman Engelbrecht

Minecraft

- 3D, first-person game
- Survive in open world
- Worlds are:
  - Procedurally generated
  - Extremely large (60x60 million blocks)
  - Dynamic
  - Diverse
- Presents some challenges faced in the real world

MineRL 2019

- Goal: Obtain a diamond in Minecraft
- Sample-efficient & generalized machine learning
- Limitations:
  - 4 days training
  - 8 million environment interactions
  - *No human domain knowledge*
  - *No external data*
- Evaluate submissions with unseen texture pack & data



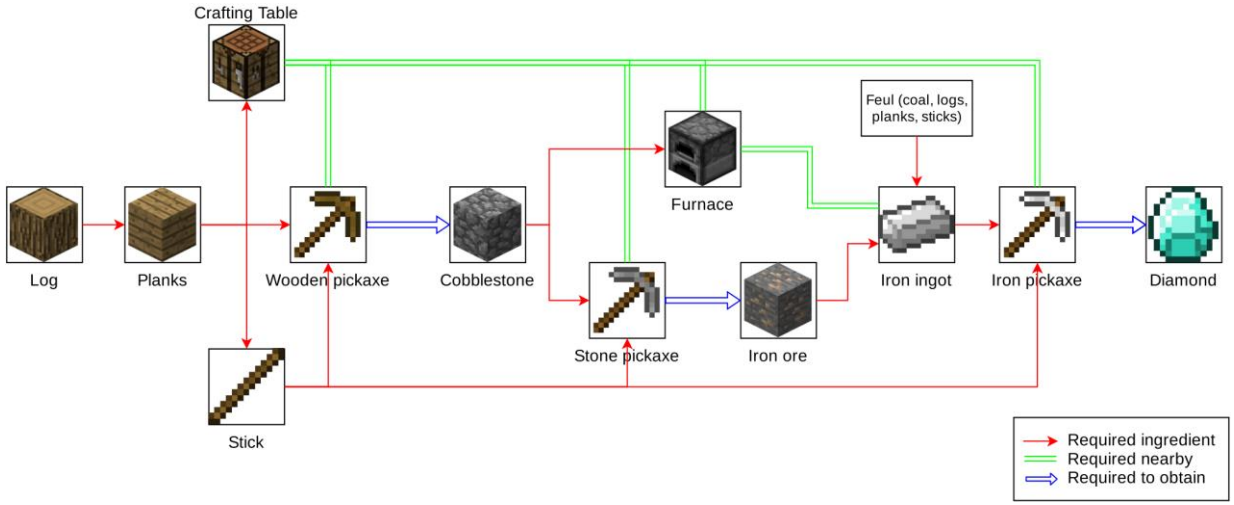
Treechop Environment

- Goal: Collect 64 logs
- Episode termination: 7 minutes or death
- Spawn location: A forest
- Equipped tool: Diamond axe

ObtainDiamond Environment

- Goal: Obtain one diamond
- Episode termination: 15 minutes or death
- Spawn location: Minecraft random spawn
- Equipped tool: None – need to craft tools

Item hierarchy to reach goal:



Observation Space

(What the agent sees)

- 64x64 RGB, Point of View (POV) image:

(Only for ObtainDiamond env)

- Vector of 3 components describing the item equipped.
- Vector representing 18 inventory items:

Coal	Cobblestone	Crafting table
Dirt	Furnace	Iron axe
Iron ingot	Iron ore	Iron pick-axe
Log	Planks	Stick
Stone	Stone axe	Stone pick-axe
Torch	Wooden axe	Wooden pick-axe

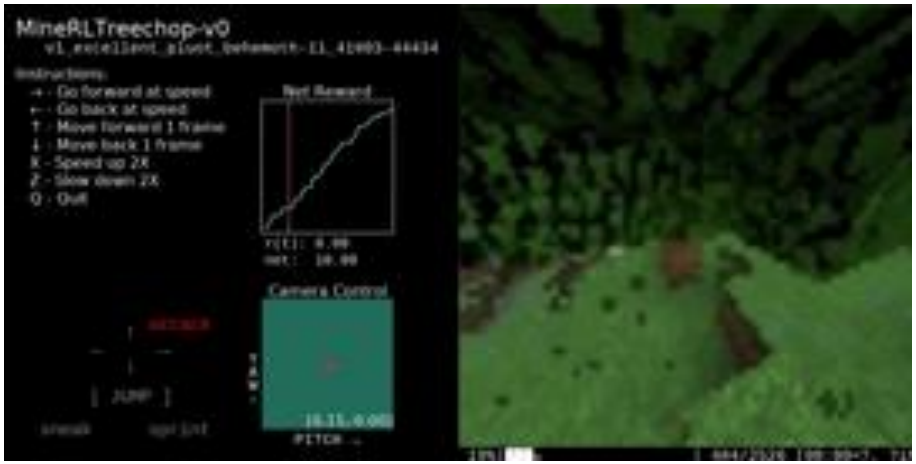
Action Space

(What the agent can do)

Movement	Crafting	Camera
Attack	Craft	Camera pitch delta
Back	Equip	Camera yaw delta
Forward	Nearby Craft	
Jump	Nearby Smelt	
Left	Place	
Right		
Sneak		
Sprint	*(Only Obtain environments)*	

MineRL Dataset

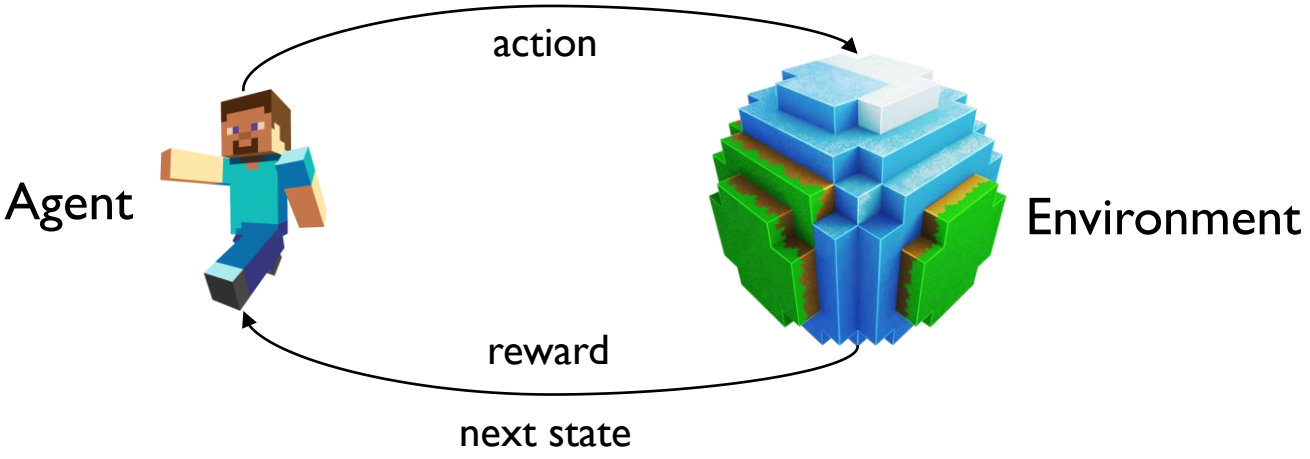
- Recordings of humans solving the MineRL environments
- Same actions & observations as the agent
- 60 million state-action pairs!
- Behavioural Cloning *(supervised learning from experts)*
- Dataset textures also altered for evaluation



Reinforcement Learning

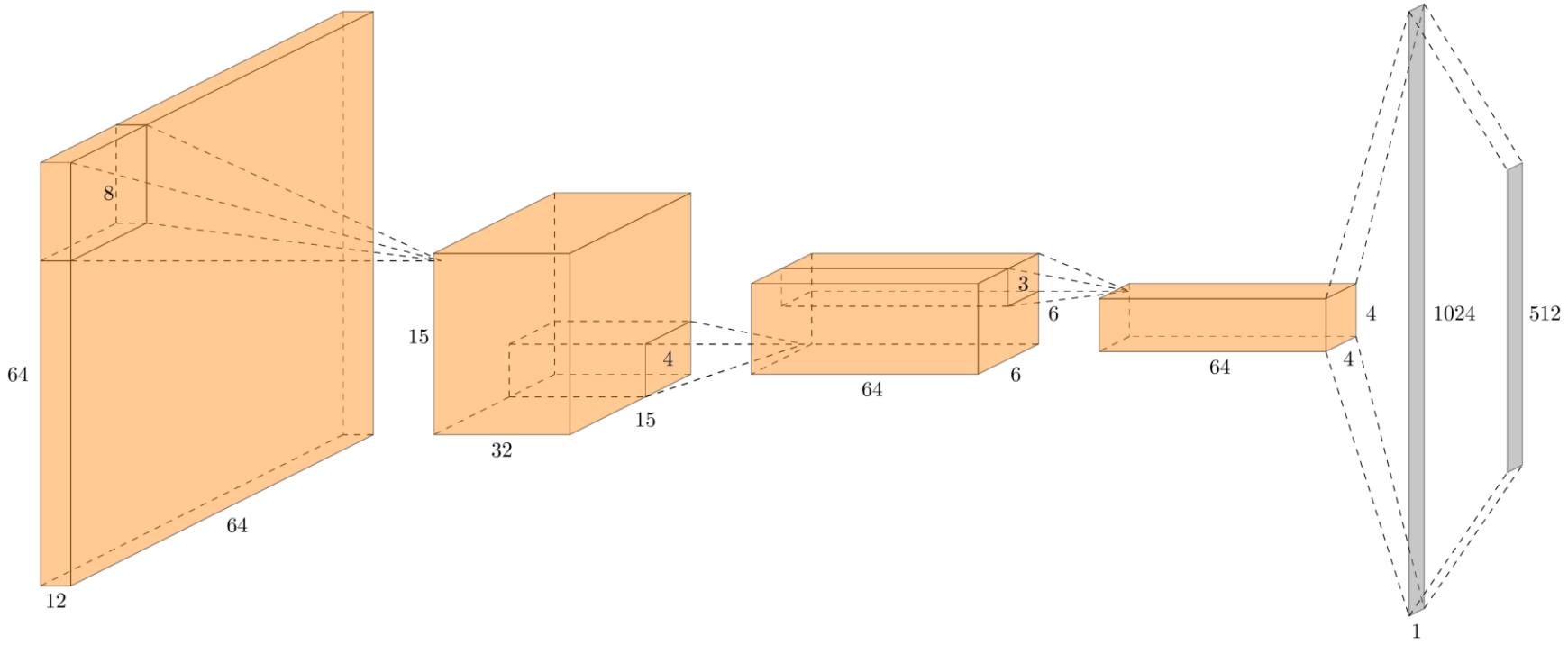
- Reward desired behaviour
- Agent learns to maximize rewards by interacting with the environment
- Slight bias toward immediate rewards
- Deep Q-Learning:

    Use a neural network to learn the value of each action in a state



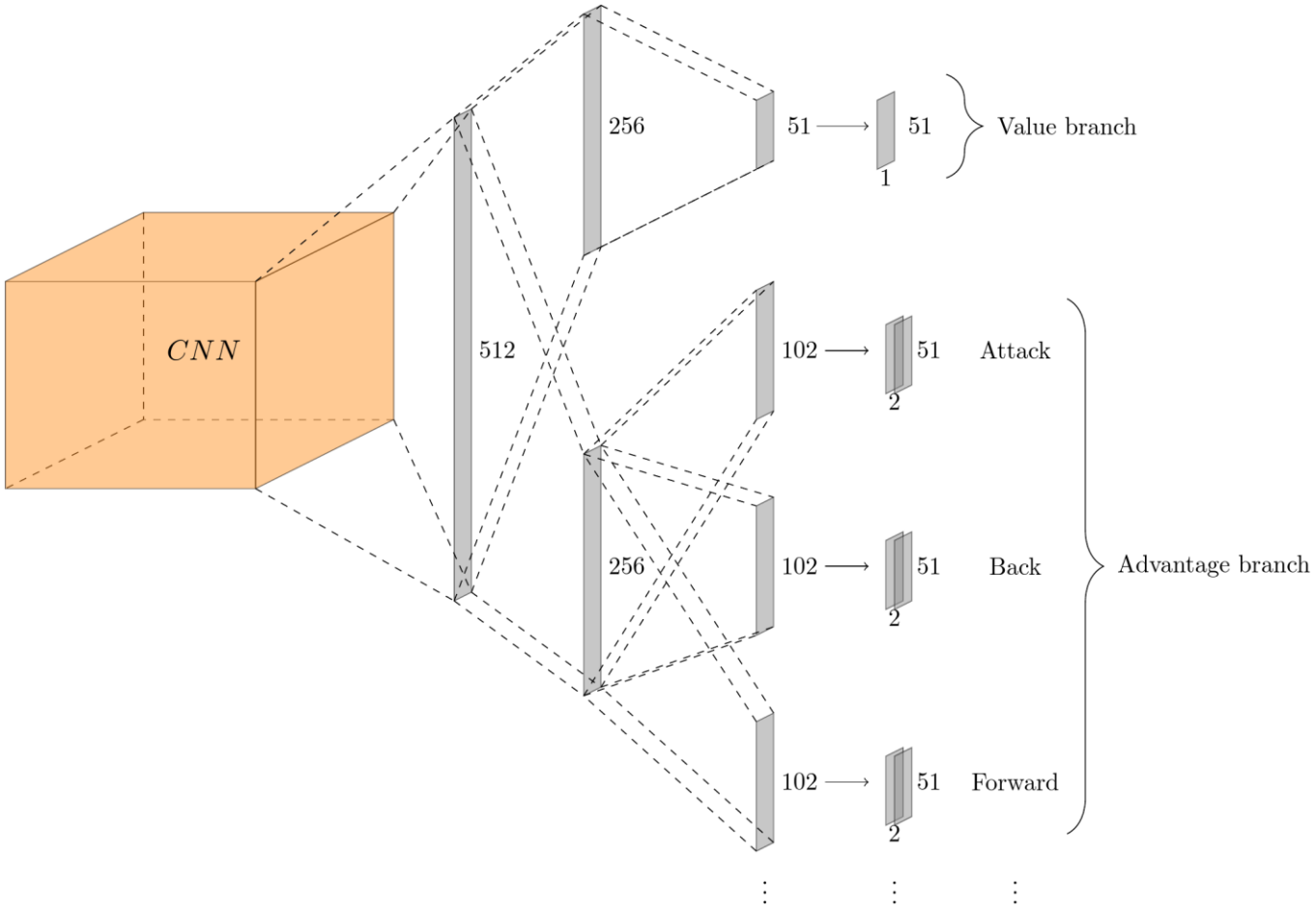
Network Architecture

Convolutional Neural Network block



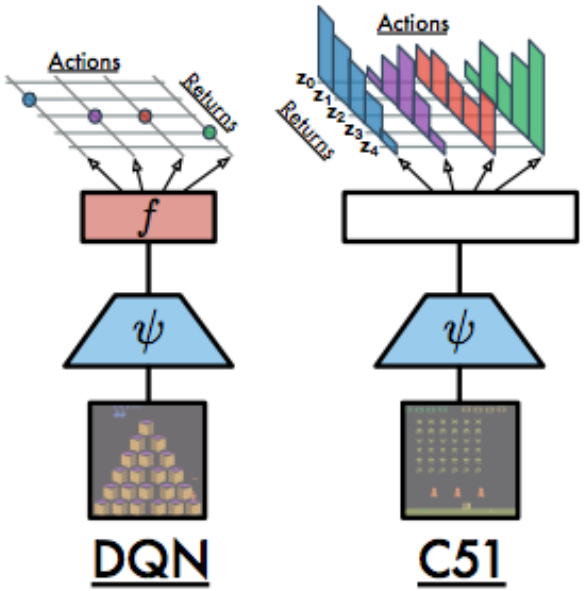
Network Architecture (cont.)

Network Head



Why 51?

- DQN does not account for uncertainty.
- Distributional RL does!
- Estimate the probability distribution for a range of returns.
- 51 atoms works well…
- Dubbed C51.

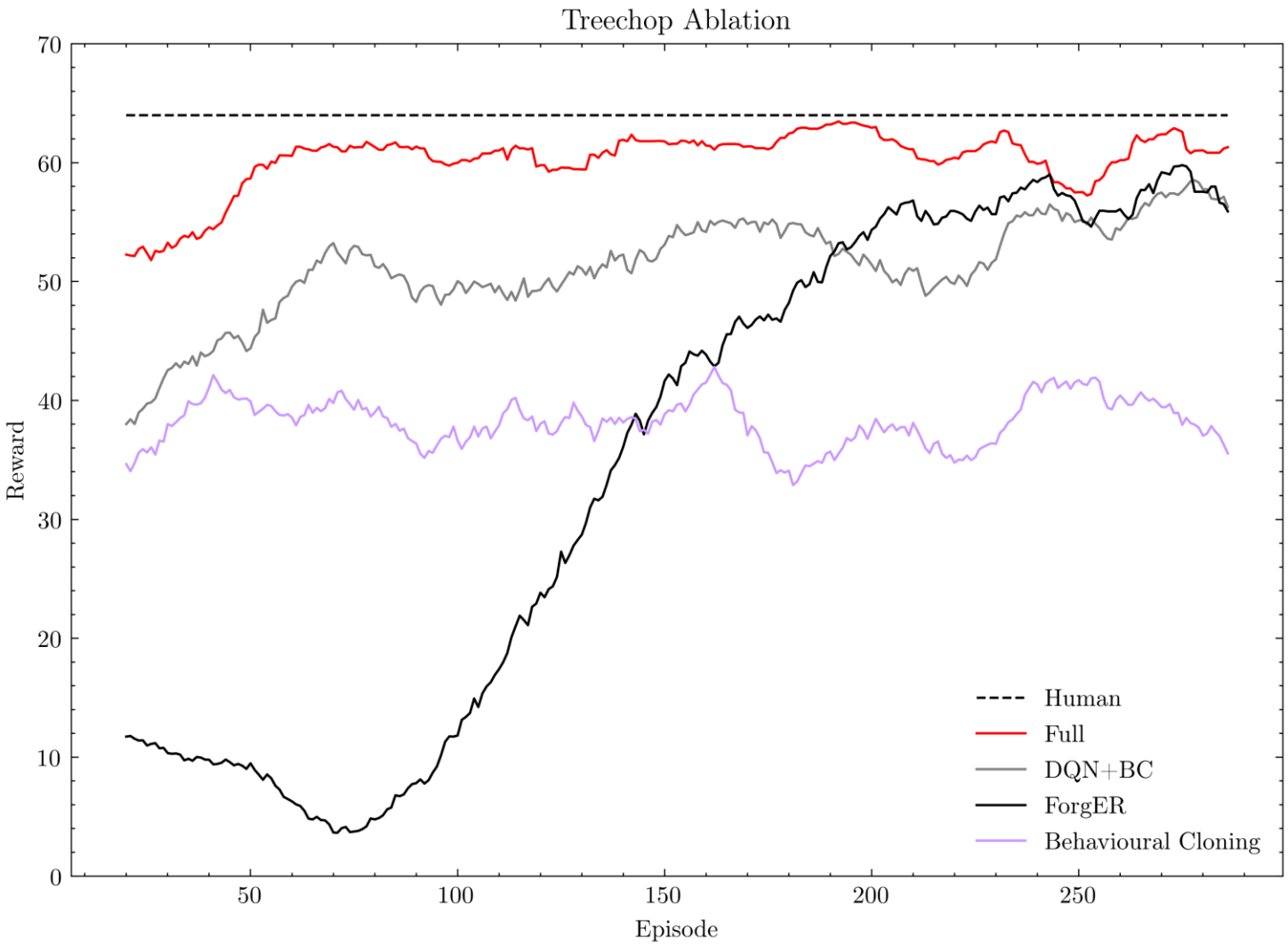


Dabney, W., Ostrovski, G., Silver, D. and Munos, R., 2018. Implicit quantile networks for distributional reinforcement learning. arXiv preprint arXiv:1806.06923.

But can it chop trees?

… Yes, it can

Ablation on Treechop



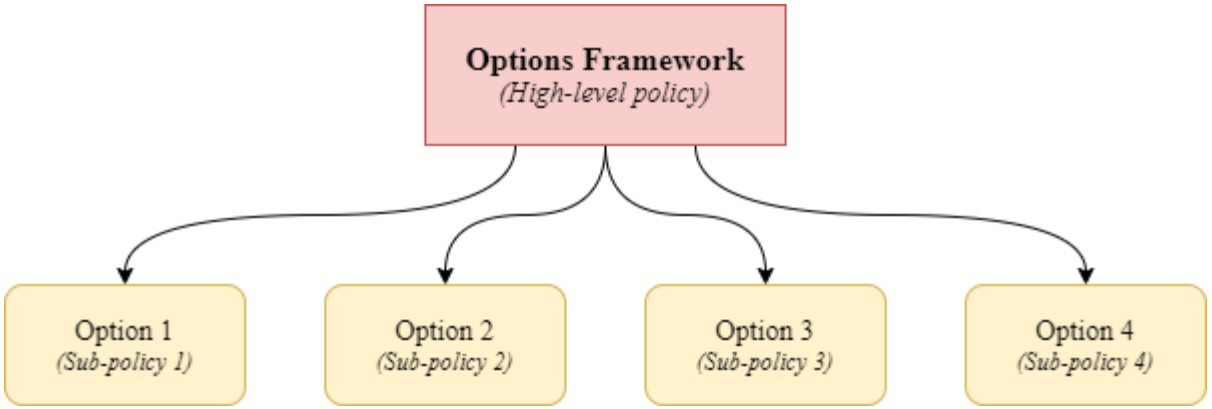
Hierarchical RL

Consider the following analogy:

- We want to build a robot to make tea
- Making tea involves many small tasks – some of which require other tasks to be completed
- For each of these small tasks, the agent needs to manipulate a robot body
- HRL splits these task into several sub-tasks
- Each of these sub-tasks are then solved by a sub-policy

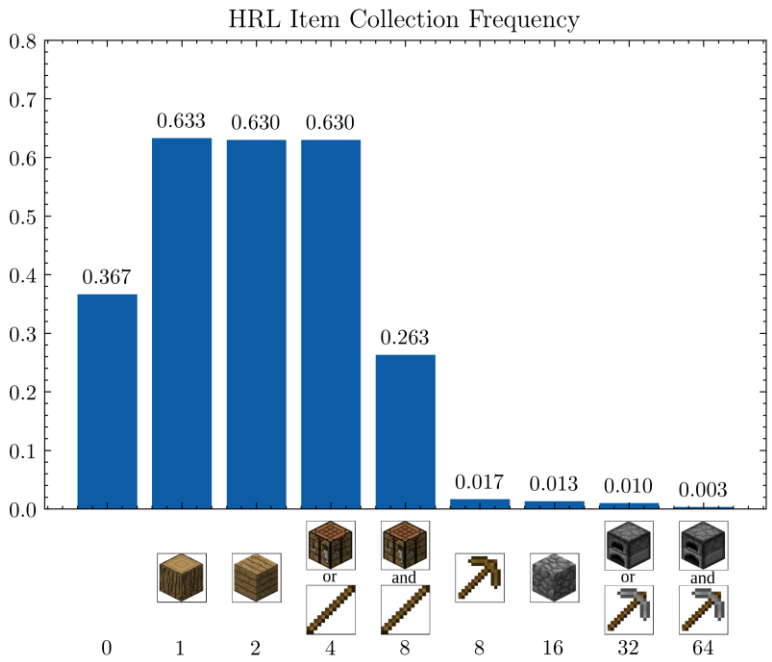
Options Framework

- Create high-level policy to choose from a set of low-level policies (options)
- Train high-level policy with dataset
- Train options with relevant observations
- Decide next option when inventory changes

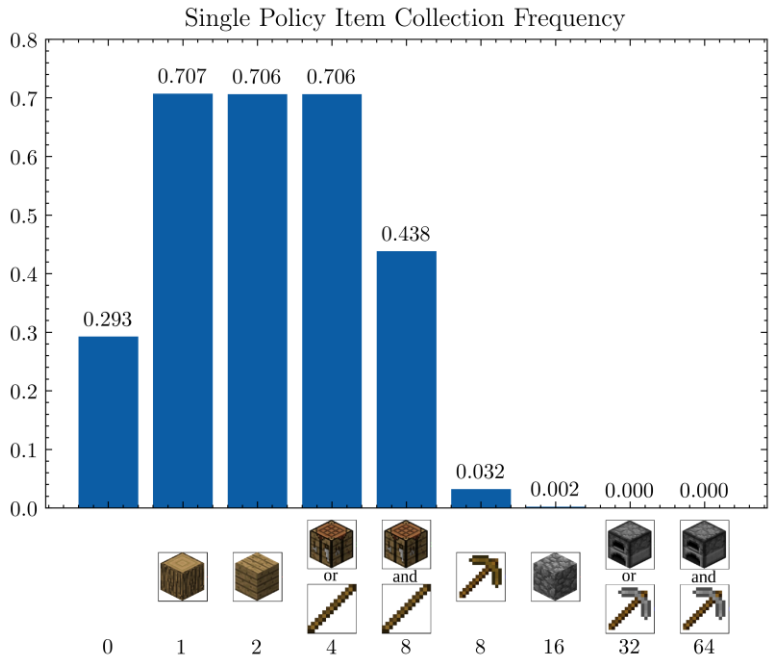


Can it obtain a Diamond?

Proportion of 300 episodes in which each item was collected



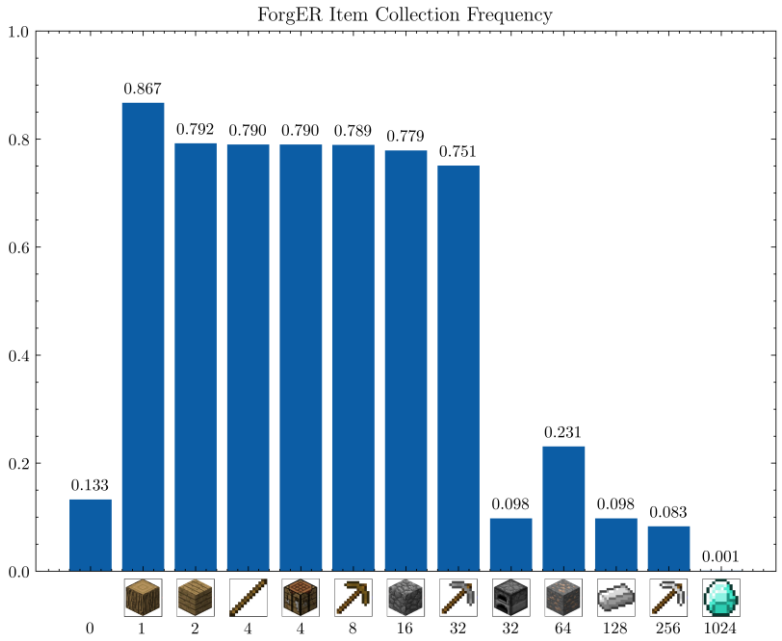
With Hierarchy



Without Hierarchy

MineRL 2019 Winner Result

Proportion of 1000 episodes in which each item was collected



ForgER: How does it work?

- Extract sample trajectory from experts
- Limit crafting actions to relevant sub-policies
- Switch sub-policy when item is collected

Lessons learned:

- Finding a diamond in Minecraft is possible.
- Currently requires some hand-crafting.

Conclusion

- Behavioural cloning provides good starting policy
- RL improves generalization
- HRL maintains knowledge for solving later tasks
- Finding a **generalized** solution is a difficult task

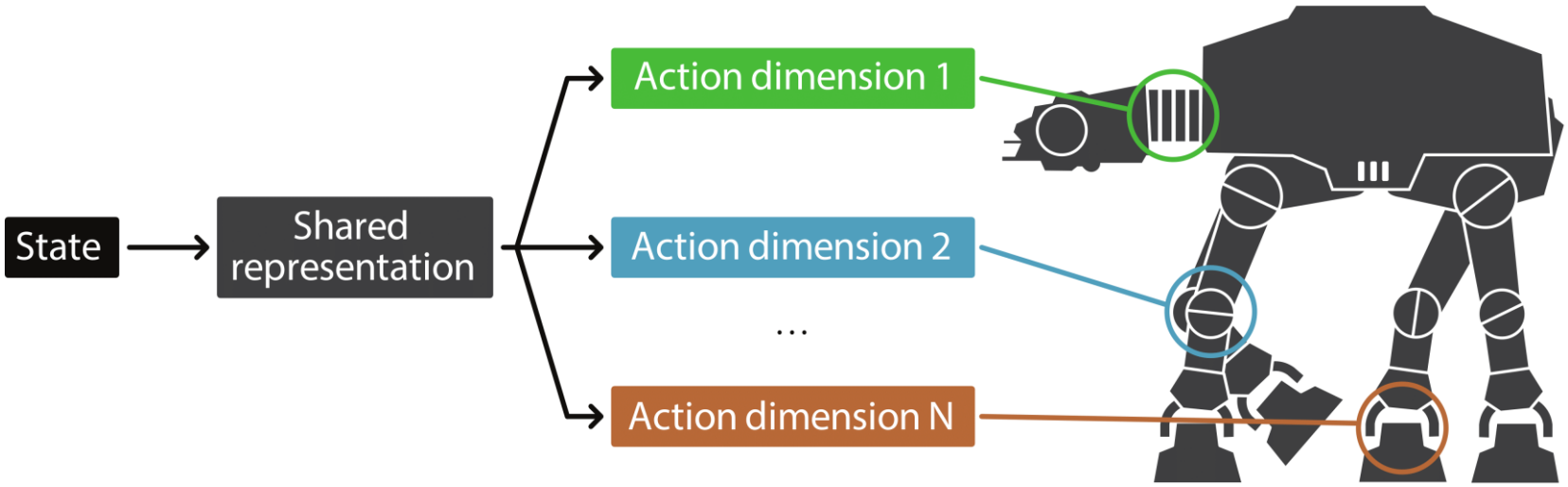
Future Work

- Select sub-policies better
  - Maybe IL+RL?
- Bayesian NNs to avoid being forced to select sub-policy.
- Improved distributional DQN methods
  - IQN or FQF?

Thank You • Dankie • Enkosi

Branching DQN

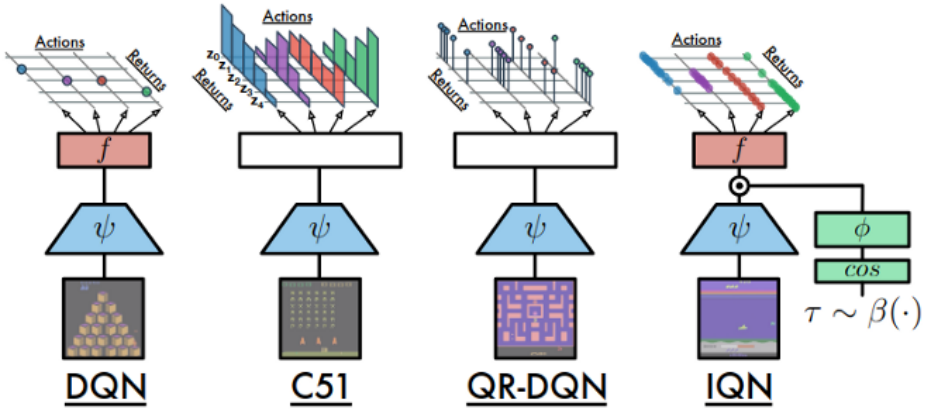
Problem: DQN can only select one action per step.
Solution: Branch for each action.



Tavakoli, A., Pardo, F. and Kormushev, P. (2017). Action branching architectures for deep reinforcement learning. CoRR, vol. abs/1711.08946. 1711.08946.

Distributional DQN methods

- Quantile Regression **(QR-DQN)**
- Implicit Quantile Networks **(IQN)**
- Fully Parameterized Quantile Function **(FQF)**



Dabney, W., Ostrovski, G., Silver, D. and Munos, R., 2018. Implicit quantile networks for distributional reinforcement learning. arXiv preprint arXiv:1806.06923.

	Mean	Median	>Human	>DQN
DQN	221%	79%	24	0
PRIOR.	580%	124%	39	48
C51	701%	178%	40	50
RAINBOW	1213%	227%	42	52
QR-DQN	902%	193%	41	54
IQN	1112%	218%	39	54
FQF	**1426%**	**272%**	**44**	**54**

Yang, D., Zhao, L., Lin, Z., Qin, T., Bian, J. and Liu, T.Y., 2019. Fully parameterized quantile function for distributional reinforcement learning. In Advances in Neural Information Processing Systems (pp. 6193-6202).

Obtain Diamond video