A Deep Hierarchical Approach to Lifelong Learning in Minecraft

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, Shie Mannor

Presented by **Yetian Wang**

June 13, 2018

Outline

- Introduction
  - Lifelong learning
  - Problem
  - Minecraft
- Background
  - RL, DQN, Double DQN
  - Skills, SMDP, Skill Policy, Policy Distillation
- Hierarchical Deep RL Network
  - Deep Skill Network
  - Deep Skill Module
    - DSN Array, The Distilled Multi Skill Network
  - H-DRLN
- Experiment
  - Training DSN
  - Training H-DRLN with DSN
  - Training H-DRLN with Deep Skill Module
- Results
- Conclusion
- Contribution and Future Work

Introduction

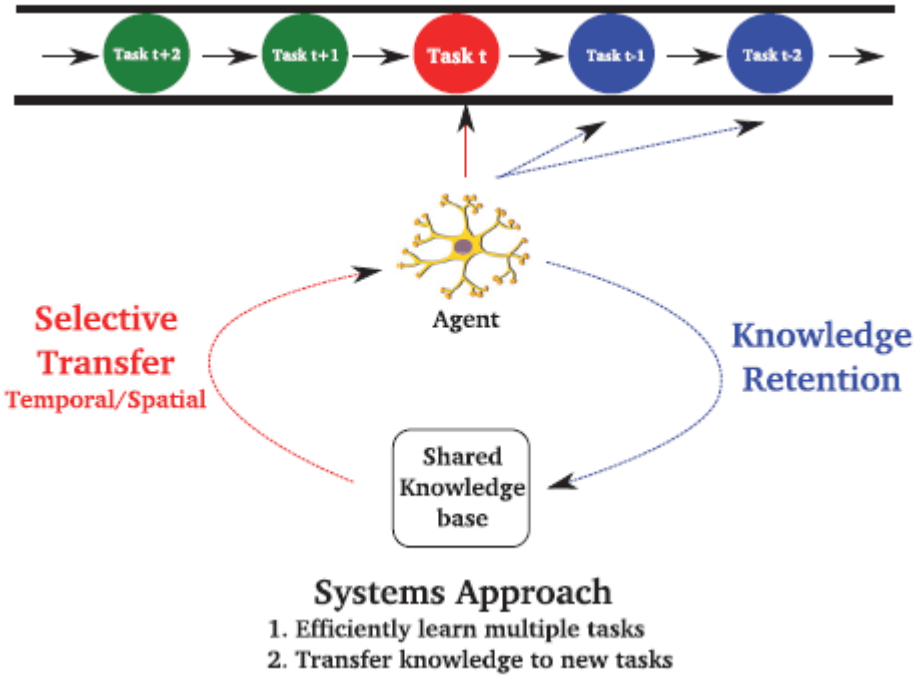
Lifelong learning, Problem, Minecraft

Lifelong Learning

Lifelong Learning is the continued learning of tasks, from one or more domains, over the course of a lifetime, by a lifelong learning system.

A lifelong learning system efficiently and effectively:

- Retains the knowledge it has learned
- Selectively transfers knowledge to learn new tasks
  - Select, reuse, and transfer past knowledge to solve new tasks
- System approach
  - ensures the effective and efficient interaction between (1) and (2)
  - Efficiently retain knowledge of multiple tasks and transfer to new tasks



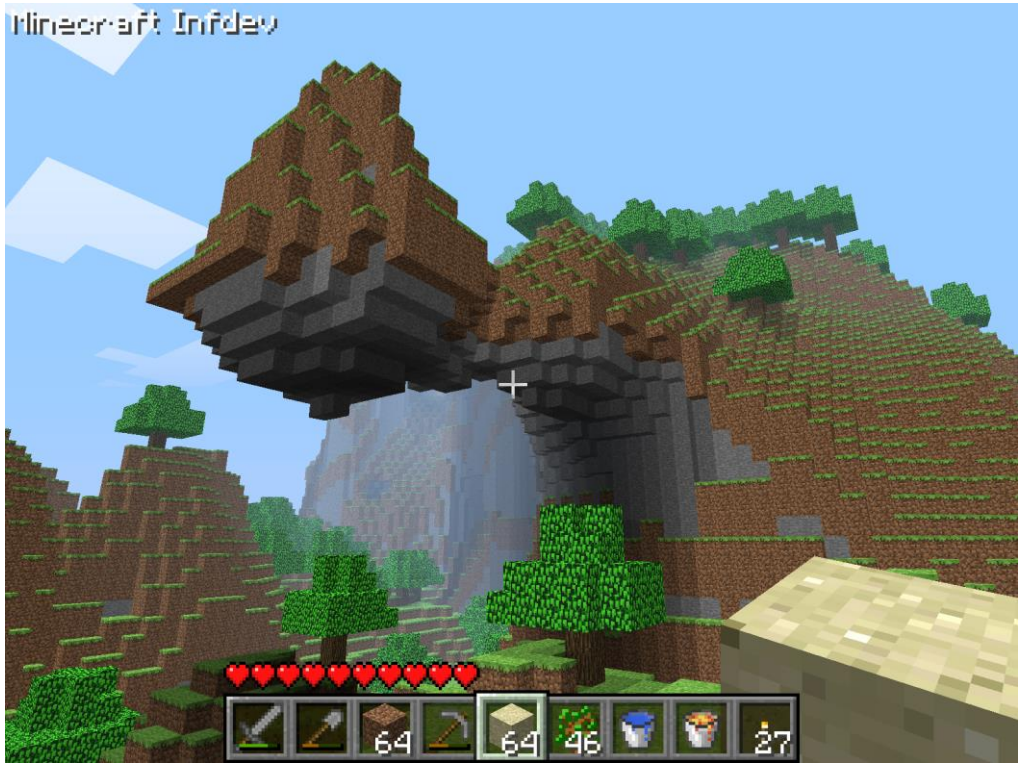
Lifelong Learning Problem

- Dimension
  - Difficult to model and solve tasks when state and action spaces increase
- Planning
  - Potential infinite time horizon
- Efficiency
  - Retaining and reusing knowledge learned

- Minecraft
  - Unsolved high dimensional lifelong learning problem

Minecraft

- Pixelized sandbox crafting-survival game
- Every pixel can be transformed into materials or gadgets/parts
- 2nd best selling video game of all-time
- Bought by Microsoft for $2.5 billion



Tasks in Minecraft

- Solve sub-problems
- Skill Hierarchies
  - Building a wooden house
  - Cutting tree – get wood – make boards, etc.
- Skills can be reused
  - Build a city
  - Start from building a house

- In order to solve Minecraft, we need to:
  - Learn Skill
  - Learn a controller
    - When to use and reuse skill
  - Efficiently accumulate reused skills

Background

DDQN, Skill, Skill Policy

Deep Q Networks

- Deep Q Networks (DQN)
  - Optimize Q function
  - Minimize error
- Experience Replay (ER)
  - Replay buffer
  - Stores agent’s experience at each timestep t
  - Minimize loss function
  - Two separate Q networks
  - Sync the target networks after n steps
- Double DQN (DDQN)
  - Prevents overly optimistic estimates of value functions
  - Select action from current Q network
  - Evaluate with target network

Skill and Skill Policy

- A skill $\sigma =< I, \pi, \beta >$
  - $I \subseteq S$ – Subset of states where skills can be initiated
  - π – Intra-skill policy
  - β – a function of s and t, termination probability
- Semi-MDP
  - Produces a skill policy from $< S, \sum, P, R, \gamma >$
- Skill policy
  - Mapping between state and distribution over set of skills
  - Q function with skills
- Policy Distillation
  - Distillation
    - Transfer knowledge from a teacher model to a student model
    - Distill ensemble models into a single model
  - Learn from multiple teachers, i.e., multiple policies

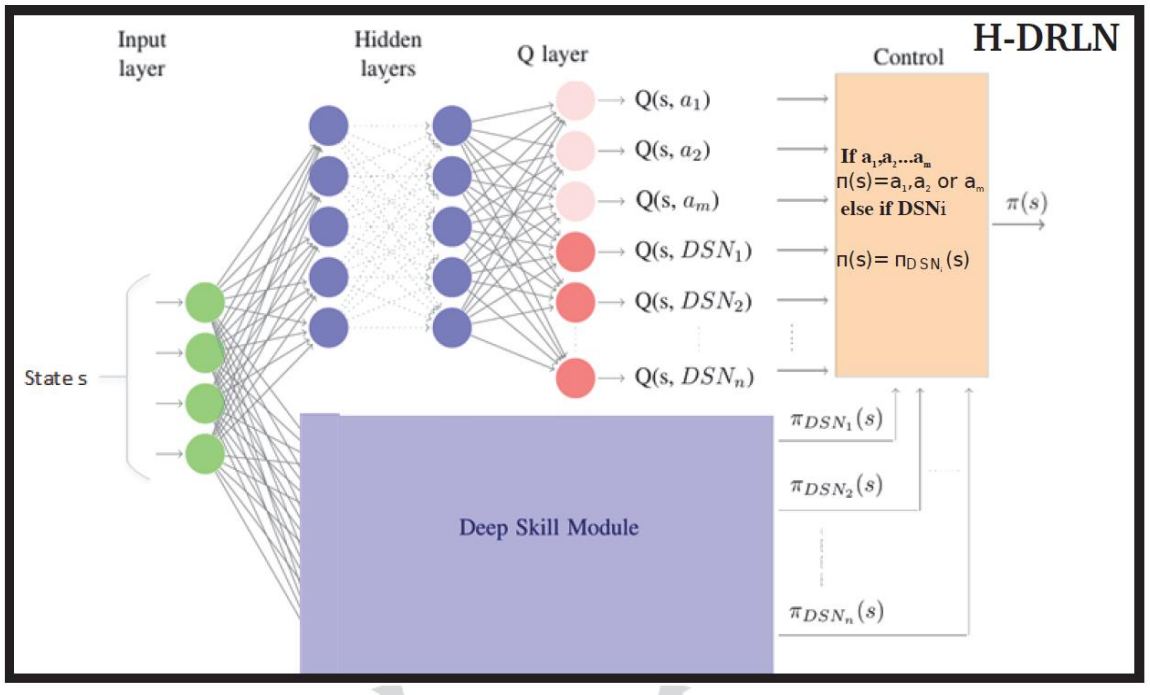
$$Q^*_\Sigma(s, \sigma) = \mathbb{E}[R^\sigma_s + \gamma^k \max_{\sigma' \in \Sigma} Q^*_\Sigma(s', \sigma')] \ .$$

Hierarchical Deep RL Network (H-DRLN)

H-DRLN, DSN, Deep Skill Module, Experiment, Result

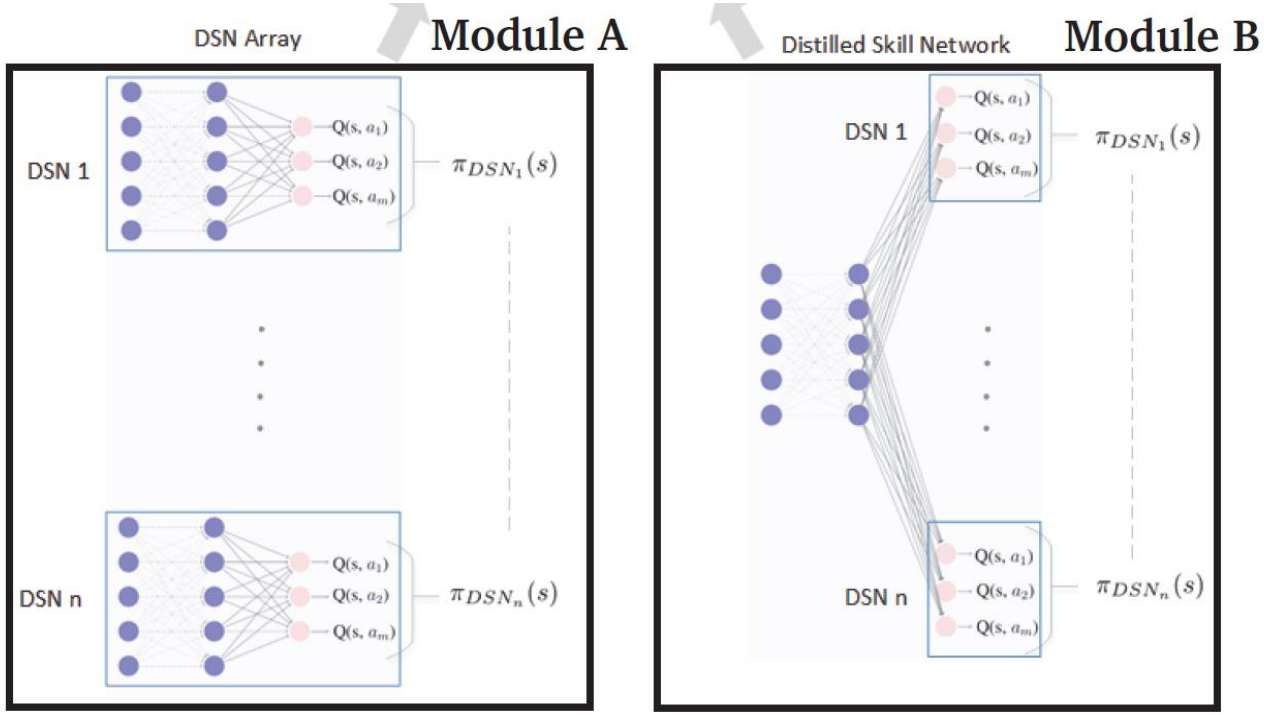
Hierarchical Deep RL Network (H-DRLN)

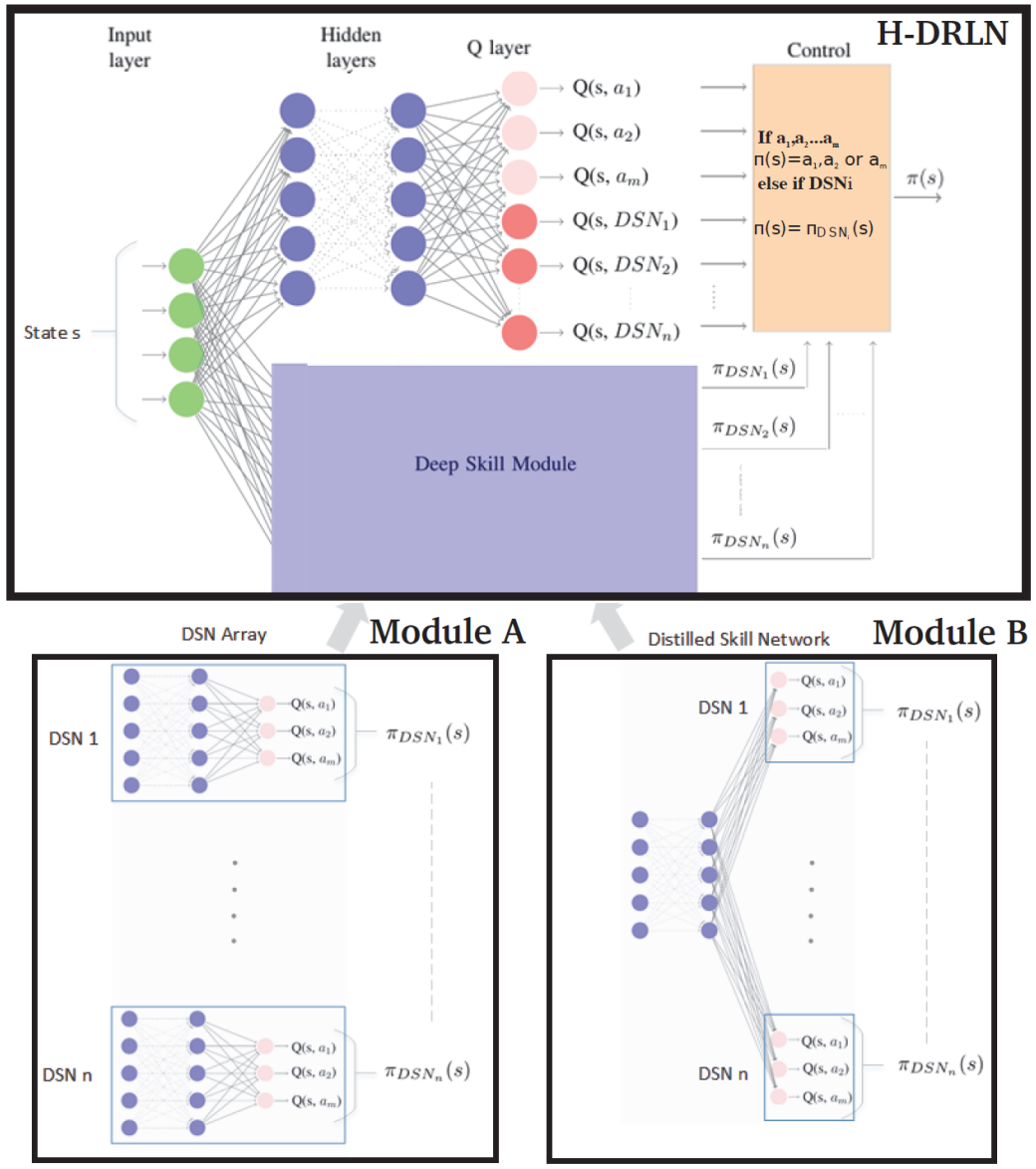
- Extends DQN
- Outputs either
  - Primitive action
    - Move forward, rotate, pick up, place break a block
    - Executes action for t
  - Learned skills
    - Navigation, pick up, placement, break
    - Executes policy π_{DSNi} until it terminates
- Using Deep Skill Module



Deep Skill Module

- Deep Skill Network (DSN)
  - Previously learned skills
  - DSN executes its policy π_{DSNi} if a skill is executed
- Deep Skill Module
  - A set of N DSNs
  - Input: s, skill index i, policy π_{DSNi}
  - Output: a
- DSN Array
  - Separate DQN for each DSN
- The Distilled Multi-Skill Network
  - Single network for multiple DSNs
  - Hidden layers are shared
  - Output layer trained separately for each DSN
    - Trained with policy distillation
  - # of skills -> scalable to lifelong learning





$$(s_t, a_t, s' = s_{t+1}, r) \rightarrow (s_t, \sigma_t, s' = s_{t+k}, \tilde{r}_t)$$

Experiment

Sub-Domain, Two-Room Domain, Complex Domain, Results

Experiment

- States: raw image pixels from picture frames
- Primitive Actions
  - Move forward
  - rotate left/right by 30 degrees
  - break a block
  - pick up an object
  - place an object
- Rewards
  - Small negative reward after each timestep
  - Non-negative reward after reaching the final goal

Experiment

- Domain
  - Sub-domain (DSNs)
  - Two-room domain
  - Complex domain with three different tasks

- Training
  - Episodes with 30, 60, 100 steps for single DSN, two-room and Complex domain
  - Initialization
    - Random in each DSN, 1st room in other domains

Sub-Domains in Minecraft

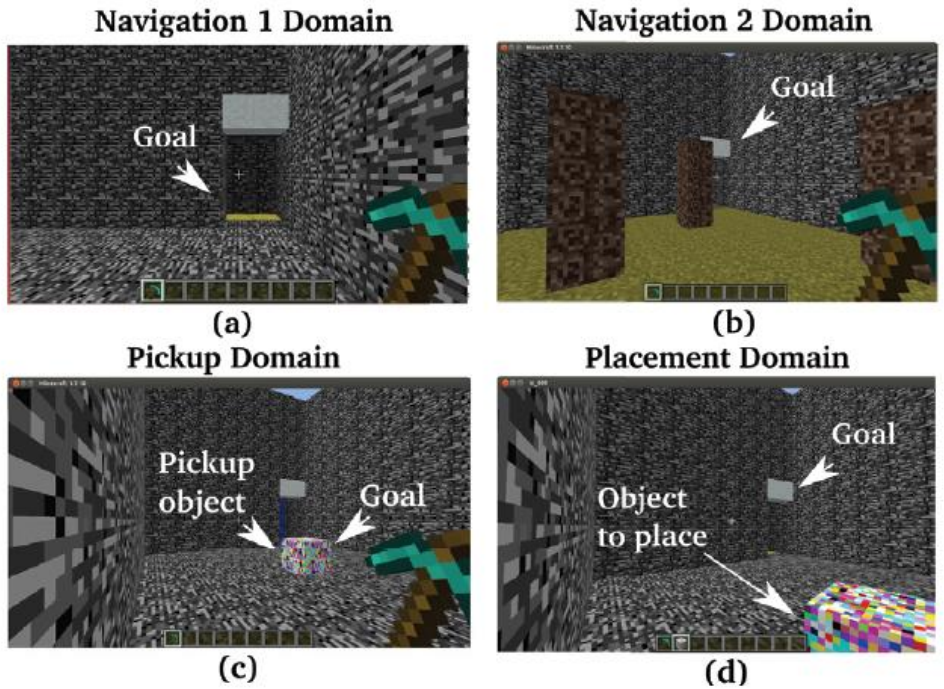


Figure 4: *The domains:* (a)-(d) are screenshots for each of the domains we used to train the DSNs.

Training a DSN (sub-domain)

- Challenge
  - Identical walls – visual ambiguity
  - Obstacles
  - navigating to a specific location and ending with the execution of a primitive action (Pickup, Break or Place respectively).
- Optimal Hyper-parameters for DQN on Minecraft emulator
  - Higher learning ratio, learning rate
  - Less exploration
  - Smaller ER
  - Rest unchanged
- Almost 100% success rate on task completion

Composite Domains

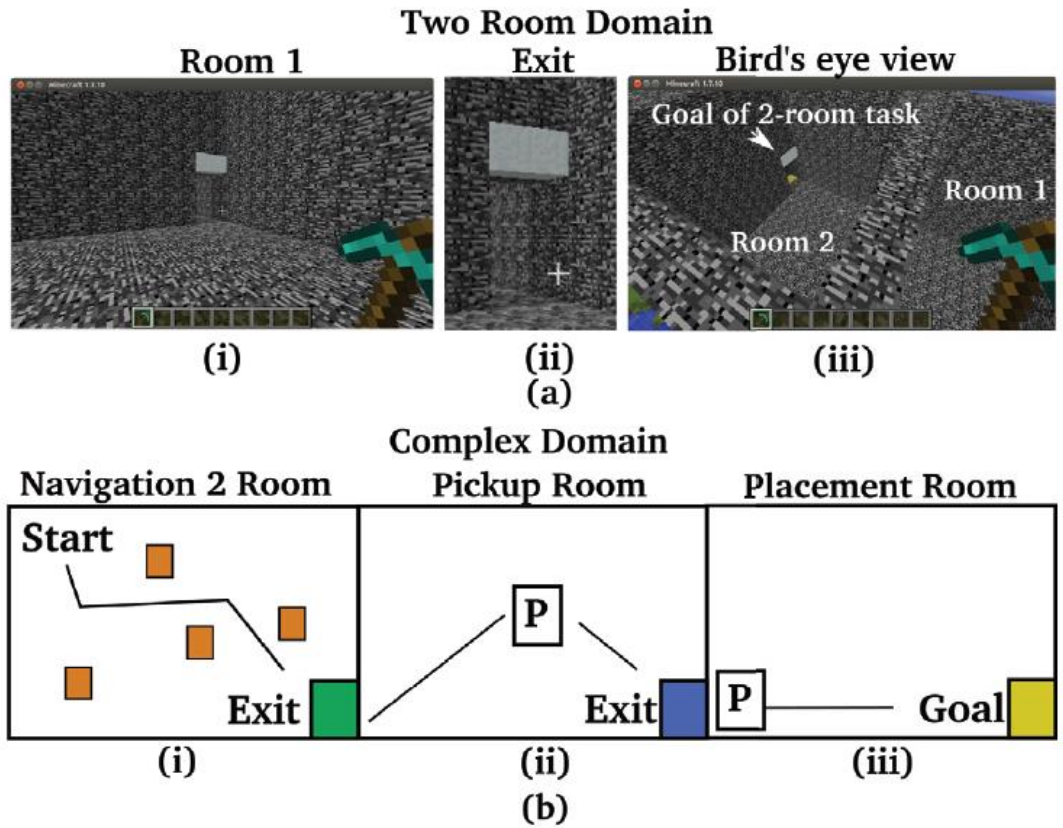
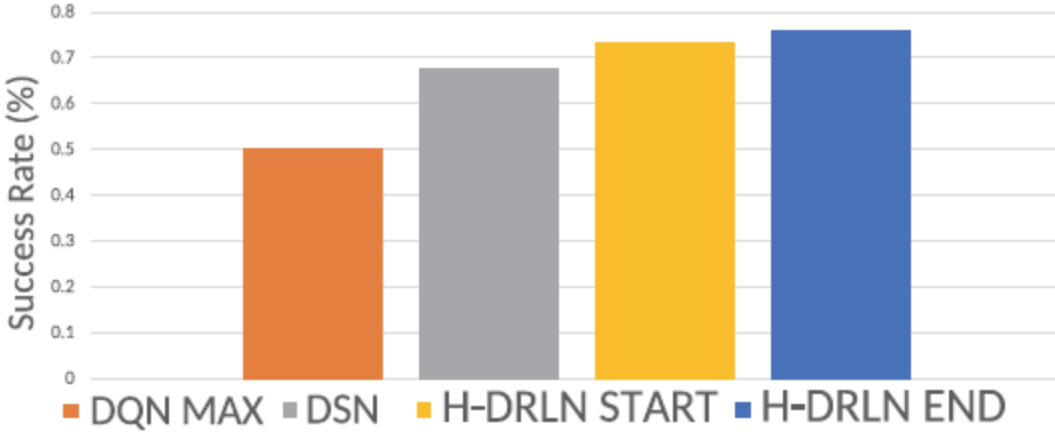


Figure 5: *Composite domains:* (a) The two-room domain and (b) the complex domain with three different tasks, (i) navigation, (ii) pickup and (iii) placement

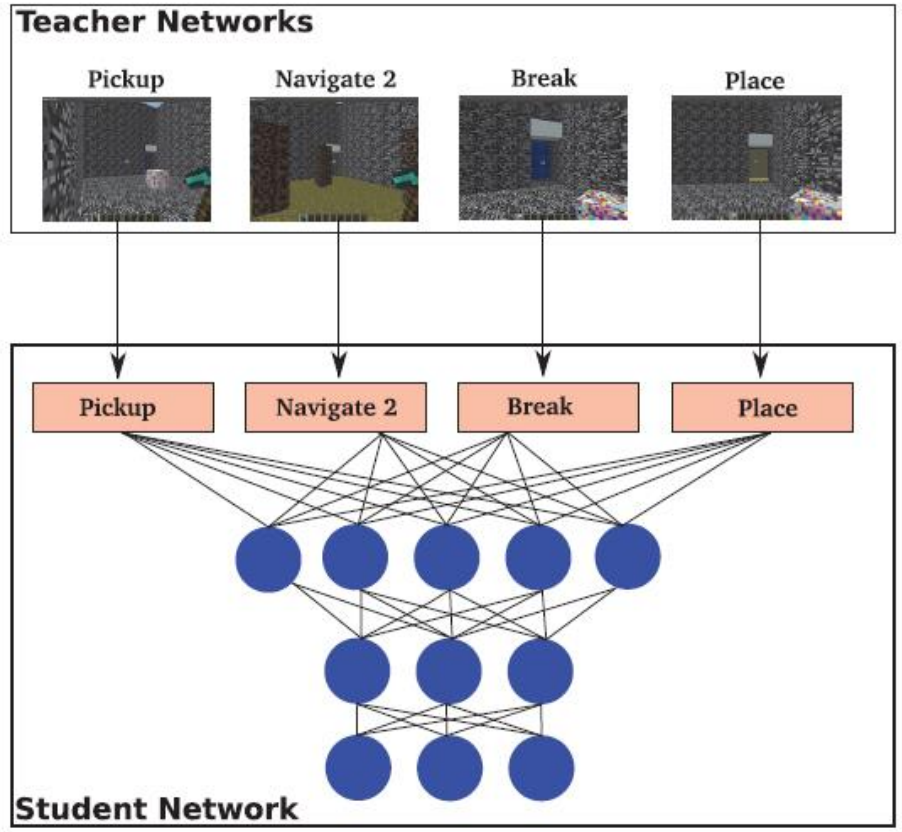
Training an H-DRLN with DSN

- Two Room Domain
- Reuse DSN pretrained in sub-domain
  - Identify the exit in first room
    - different from sub-domain
  - Navigate to the exit in next room
    - same as navigation 1
- H-DRLN solves the task after a single epoch
  - Higher reward than DQN
    - 50% vs 76% after 39 epochs
  - Wall ambiguity
- Knowledge Transfer without Learning
  - Evaluate DSN without any training on the Two-Room domain
  - Still better than DQN – specifically trained on the domain

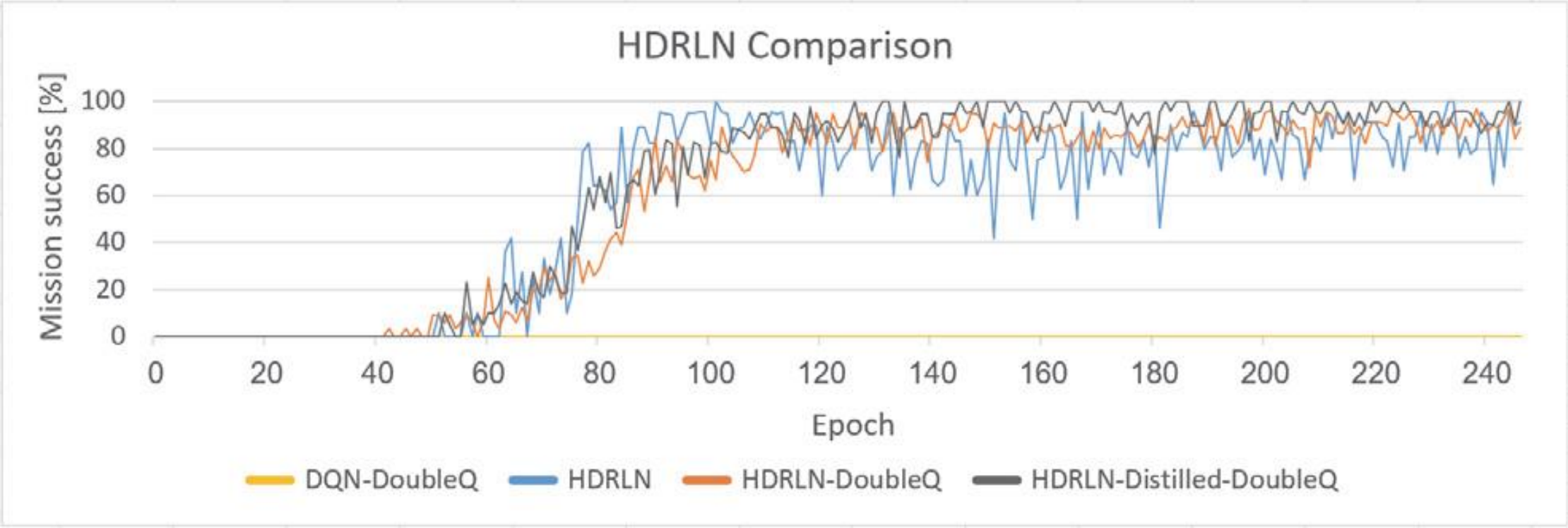


Training an H-DRLN with Deep Skill Module

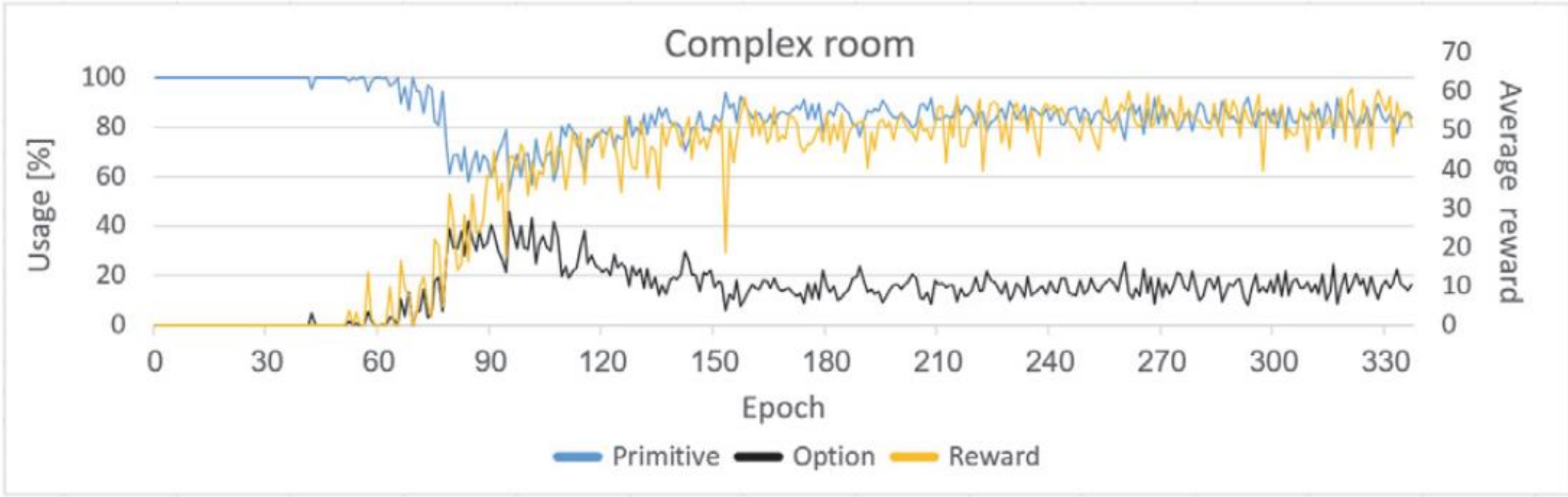
- DDQN was utilized to train the H-DRLN
- Complex Minecraft Domain
  - Room 1: navigate around obstacles
  - Room 2: pick up a block and break the door
  - Room 3: place the block at goal
  - Reward
    - Non-negative reward when all tasks are complete
    - Small negative reward at each timestep
- DSN Array
  - Formed by 4 previously trained DSNs
- Multi-Skill Distillation
  - DSNs are teachers
  - Distil skills into a single network
  - Also learns a control rule that switch between skills



Result – Success Rate



Result - Skill Usage



Conclusion

Conclusion, Contribution, Future Work

Conclusion

- Extension of DQN in Minecraft domain to train DSNs
- Reuse learned skills by H-DRLN
- Multiple skills incorporated using DSN array or distilled multi-skill network
- Better performance than DDQN

Contribution and Future Work

- Contribution
  - Building blocks for lifelong learning
    - Efficient knowledge retention
    - Selective transfer of knowledge and skills
    - Interaction between the last two
  - Potential knowledge transfer without learning
- Future work
  - Capture implicit hierarchal structure when learning DSNs
  - Learn skills online
  - Online refinement of previously learned skills
  - Train agent in real world Minecraft scenarios

Questions?

Thank you