Deep Reinforcement Learning with Forward Prediction, Memory, and Hierarchy

Honglak Lee

Google Brain / U. Michigan

Joint work with

Junhyuk Oh, Ruben Villegas, Xiaoxiao Guo, Jimei Yang, Sungryull Sohn, Xunyu Lin, Valliappa Chockalingam, Rick Lewis, Satinder Singh, Pushmeet Kohli

Overview

- Deep RL with Forward Prediction

- Deep RL with Memory

- Deep RL with Hierarchy

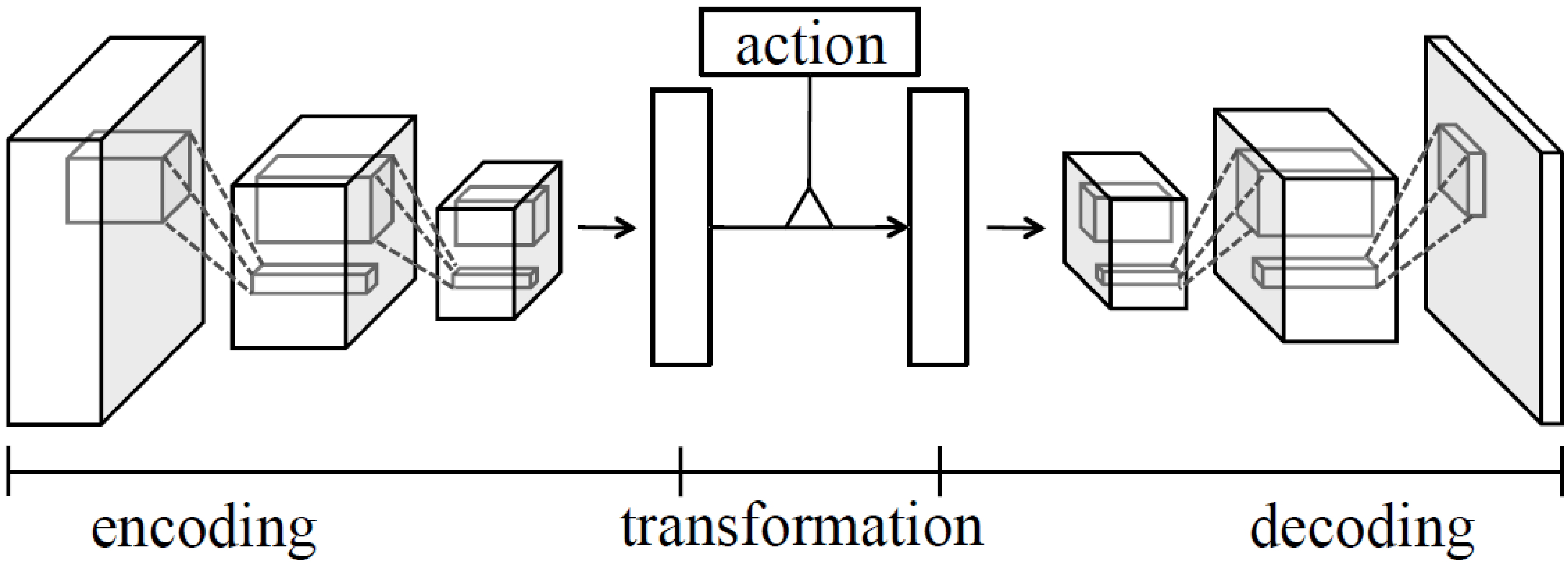
Action-conditional video prediction with Deep Architectures

- Motivation:
  - Deep convolutional encoder-decoder architecture modulated by control actions
  - End-to-end multi-step prediction
  - Application to reinforcement learning (e.g., Atari games)
- Results:
  - long-term video prediction (30-500 steps) for atari games
  - Informed exploration: Faster learning and improved performance

- Related work on video prediction
  - [Ranzato et al., 2014], [Srivastava et al., 2015], [Mathieu et al, 2015], [Finn el al., 2016]

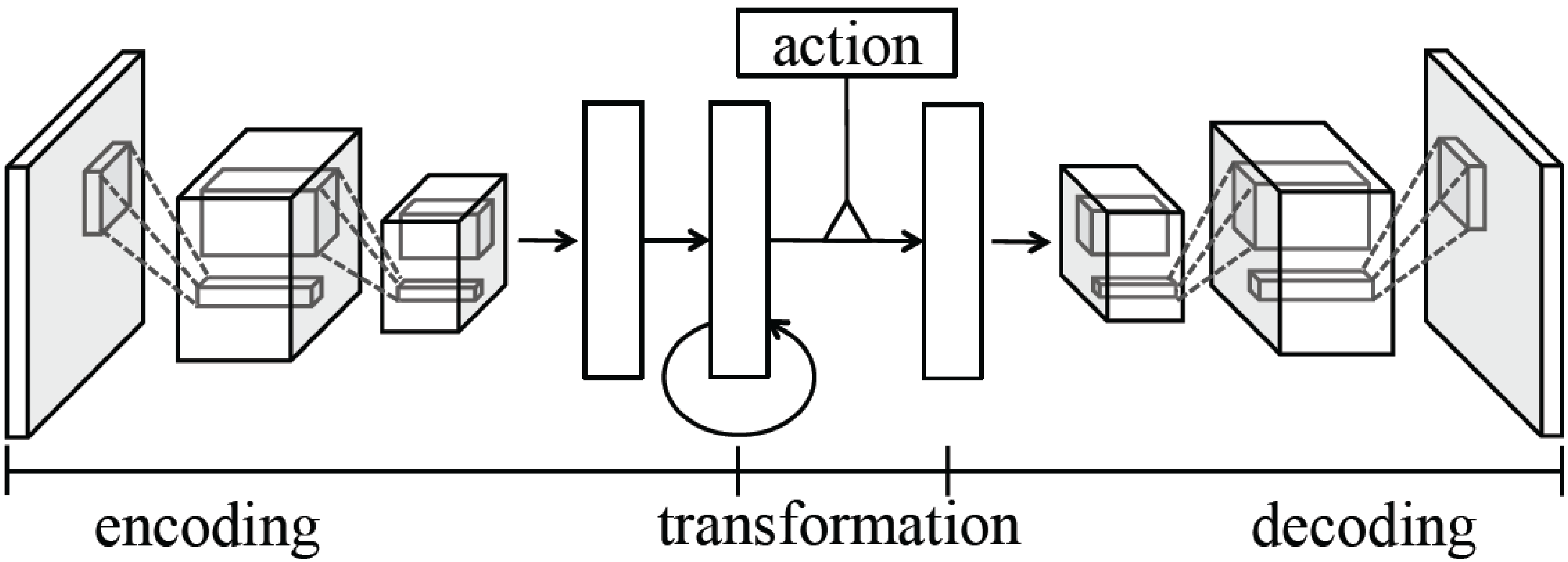
Action-conditional video prediction with Deep Architectures

- Convolutional Neural Networks (CNN)



Action-conditional video prediction with Deep Architectures

- CNN combined with Long short-term memory (LSTM)



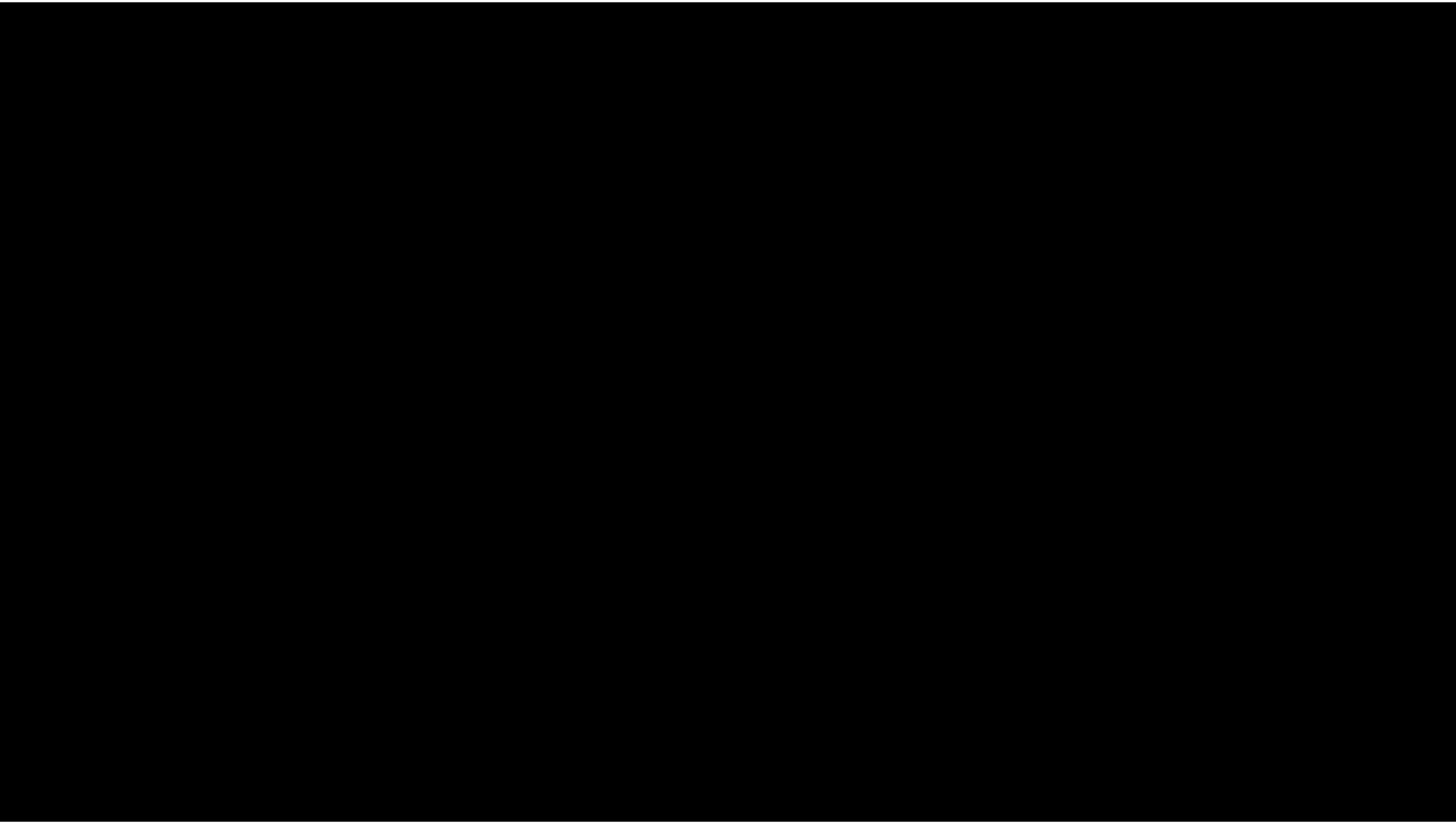
Freeway: 100 steps Predictions (LSTM)

Video: https://www.youtube.com/watch?v=4e-PqfpS8_4
More at: https://sites.google.com/a/umich.edu/junhyuk-oh/action-conditional-video-prediction

Seaquest: Multi-Step Predictions

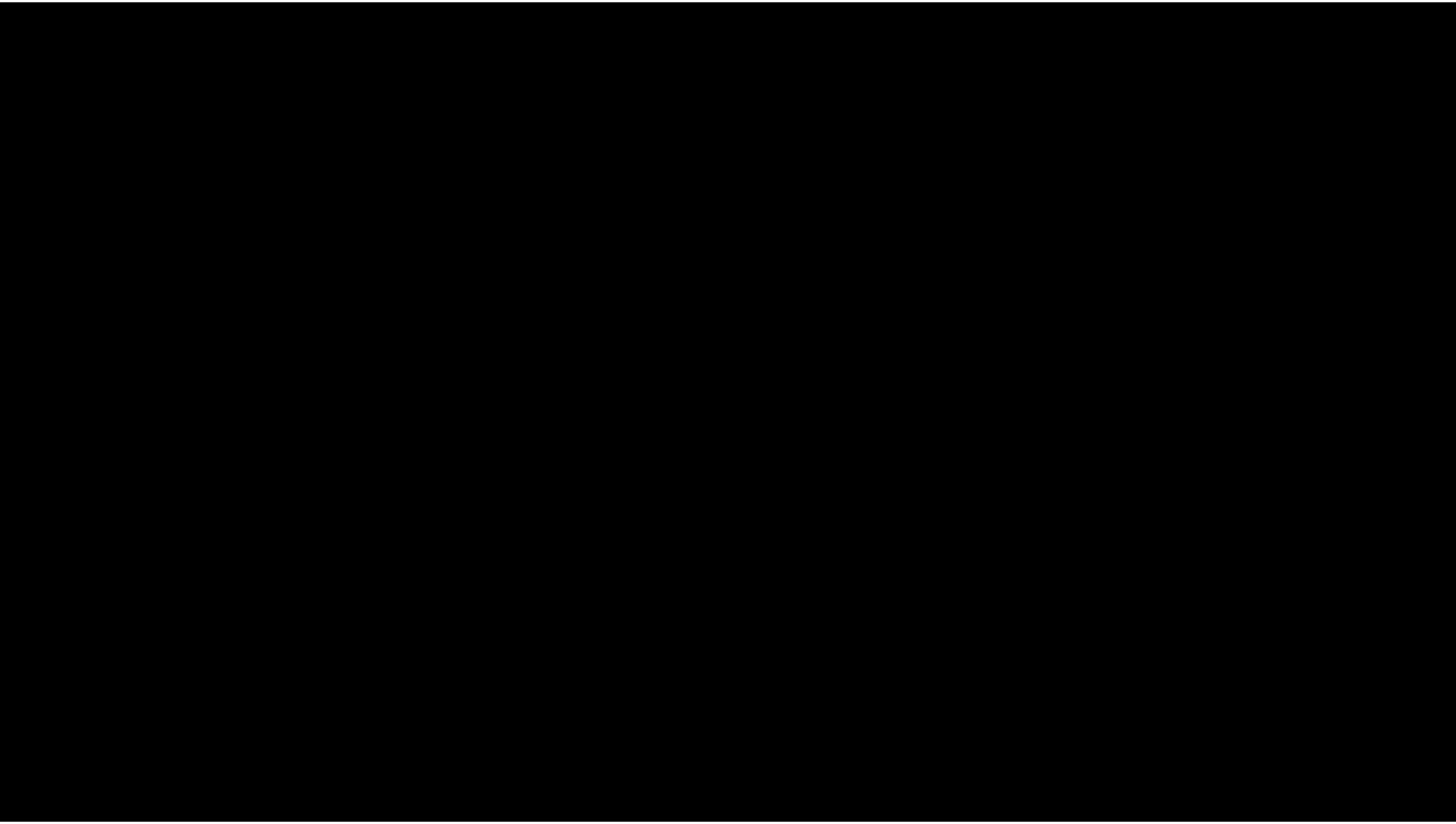
https://sites.google.com/a/umich.edu/junhyuk-oh/action-conditional-video-prediction

Space Invaders: Multi-Step Predictions



https://sites.google.com/a/umich.edu/junhyuk-oh/action-conditional-video-prediction

Ms Packman: Multi-Step Predictions



https://sites.google.com/a/umich.edu/junhyuk-oh/action-conditional-video-prediction

Informed Exploration

- **Idea** : choose an exploratory action that leads to a less-frequently-visited frame
- **Method** : estimate visit-frequency by comparing predictive frames with previous frames
  - Store the most recent *d* frames in a *trajectory memory*.
  - The predictive model is used to get the next frame ($\boldsymbol{x}^{(a)}$) for every action.
  - Estimate visit-frequency using Gaussian kernels.

$$n_D(\mathbf{x}^{(a)}) = \sum_{i=1}^{d} k(\mathbf{x}^{(a)}, \mathbf{x}^{(i)}); \quad k(\mathbf{x}, \mathbf{y}) = \exp(-\sum_{j} \min(\max((x_j - y_j)^2 - \delta, 0), 1)/\sigma)$$

  - Choose an action that leads to the frame with the smallest visit-frequency: $\operatorname{argmin}_a n_D\left(\mathbf{x}^{(a)}\right)$

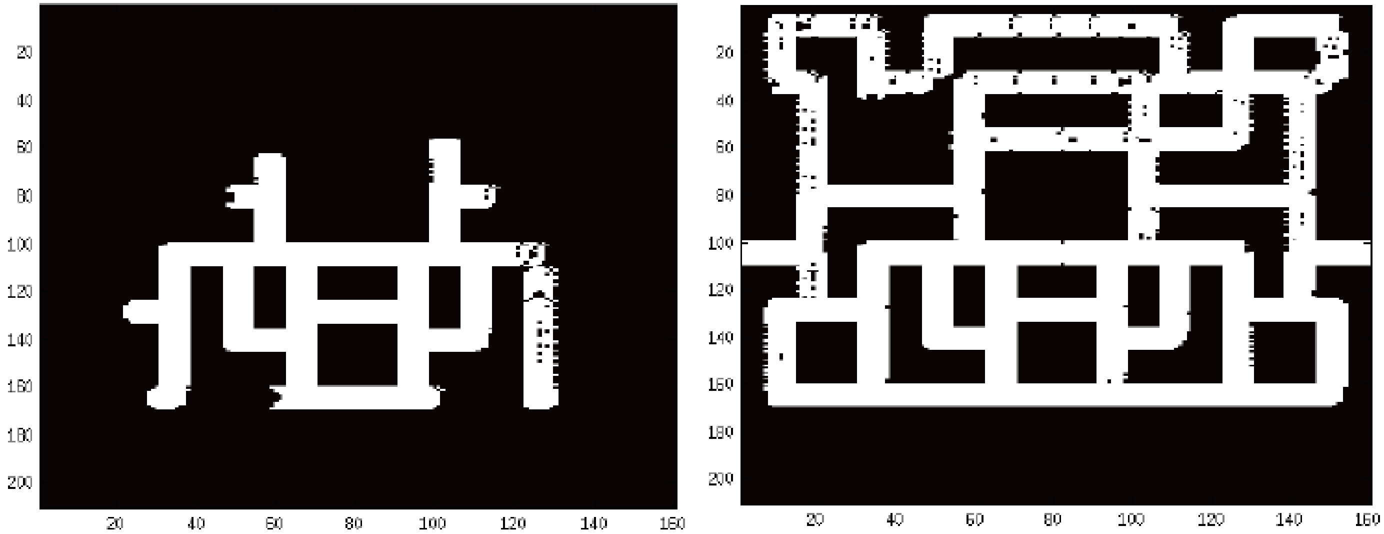
Informed exploration with future predictions

- **Idea:** choose an exploratory action that leads to a less-frequently-visited frame

(Oh et al., NIPS 2015)

Model	Seaquest	S. Invaders	Freeway	QBert	Ms Pacman
DQN - Random exploration	13119 (538)	698 (20)	30.9 (0.2)	3876 (106)	2281 (53)
DQN - Informed exploration	13265 (577)	681 (23)	32.2 (0.2)	8238 (498)	2522 (57)

Average Game Score over 100 plays with DQN

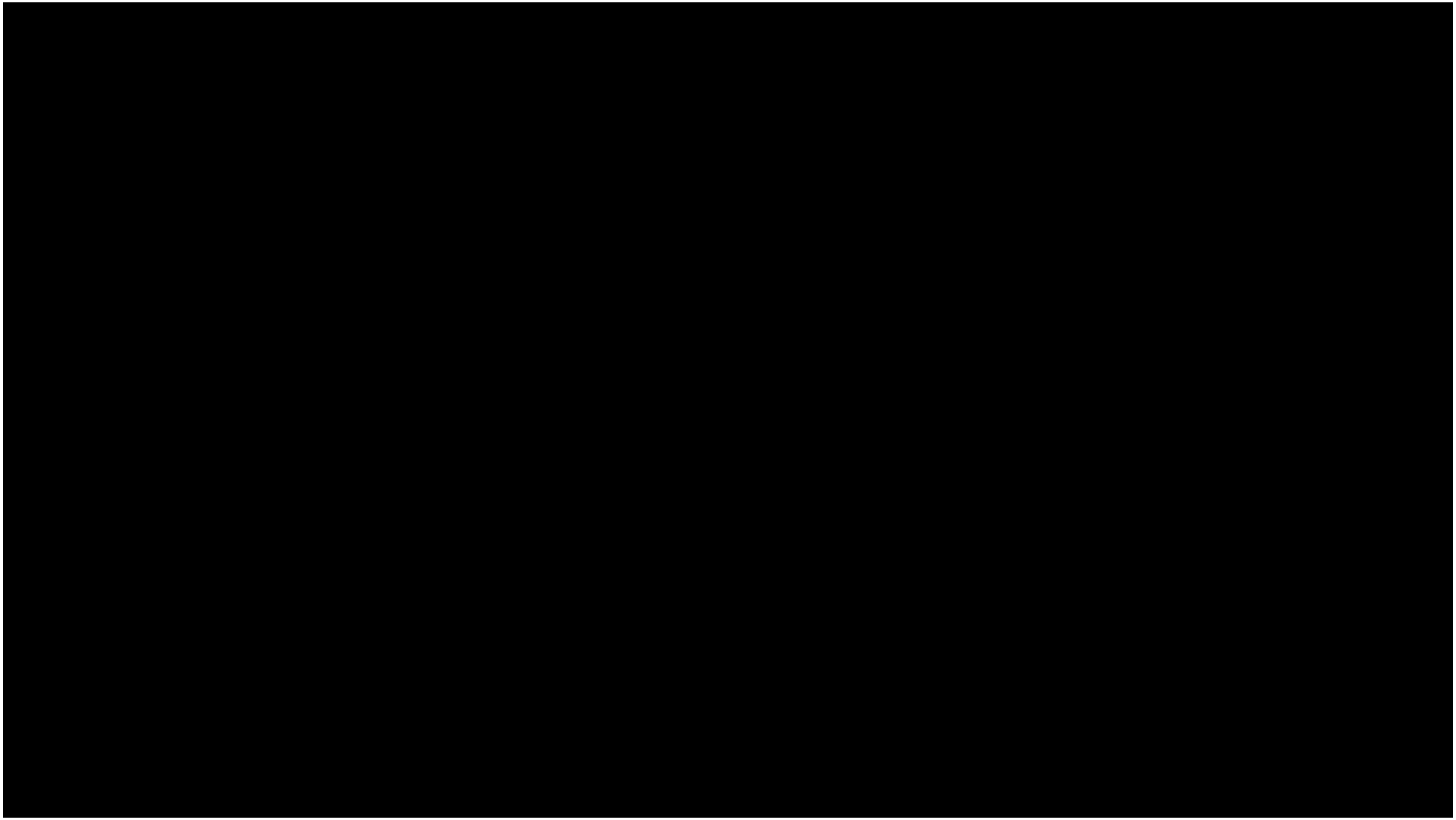


(a) Random (b) Informed

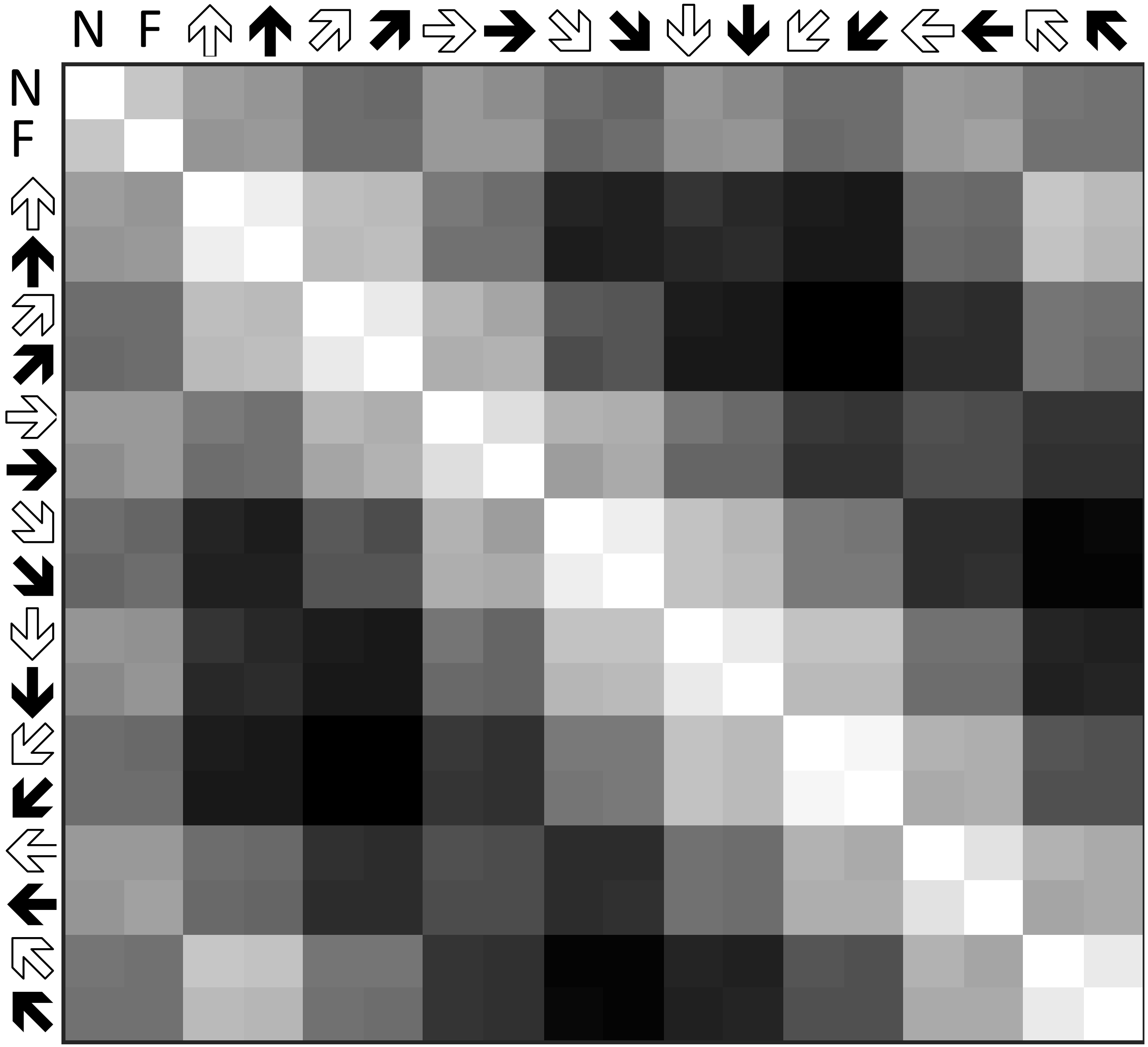
Comparison on exploration methods

Video demo: https://www.youtube.com/watch?v=DLIWo16r5LA

Using Predictions to Improve Exploration in DQN



(18) Action Representations: Correlations for Seaquest

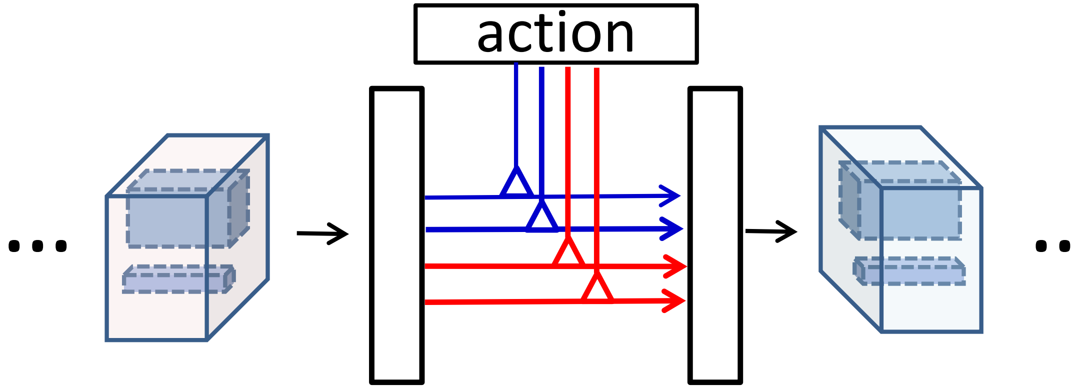


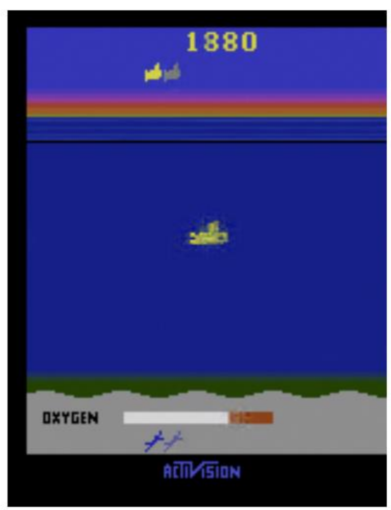
Emergence of disentangling

Prev. frame

Prediction

Next frame

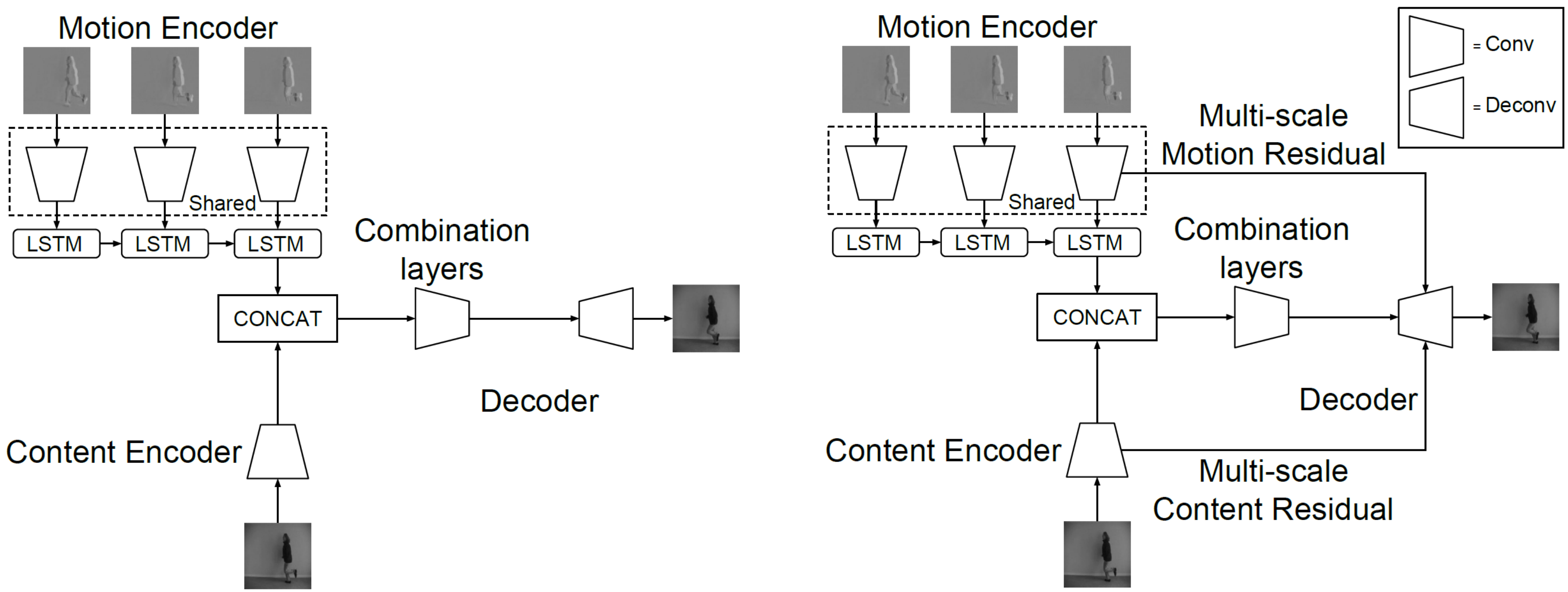




Action factors

Non-action factors

Improving forward prediction with motion/content decomposition



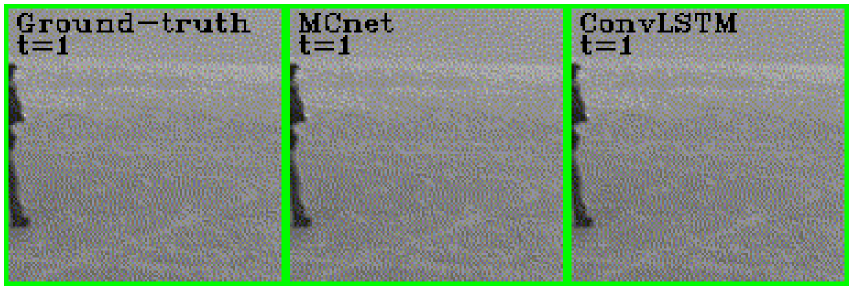
(a) Base MCnet

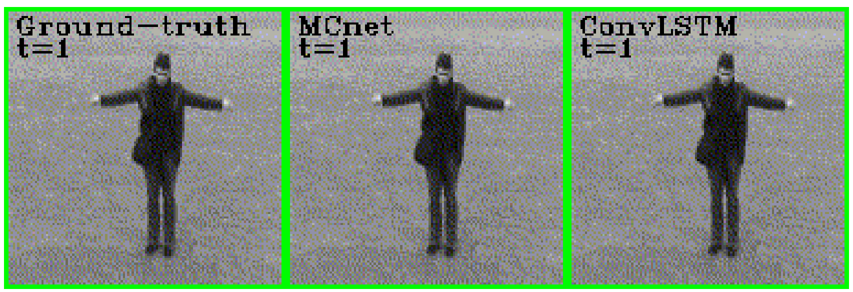
(b) MCnet with Multi-scale Motion-Content Residuals

Decomposing Motion and Content for Natural Video Sequence Prediction.
Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, Honglak Lee. ICLR 2017.

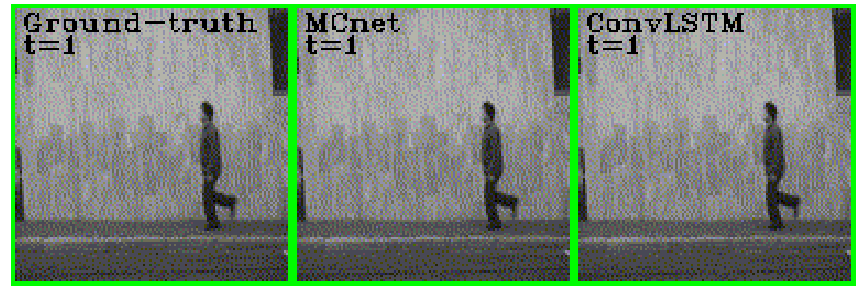
Experimental results

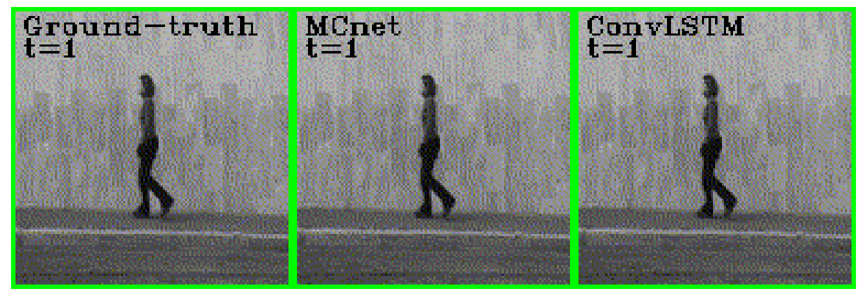
KTH dataset

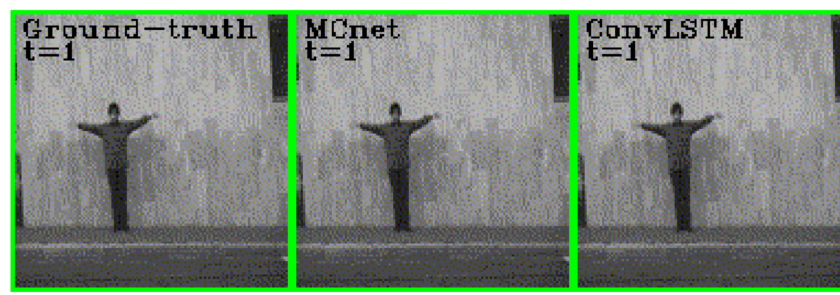




Weizmann dataset

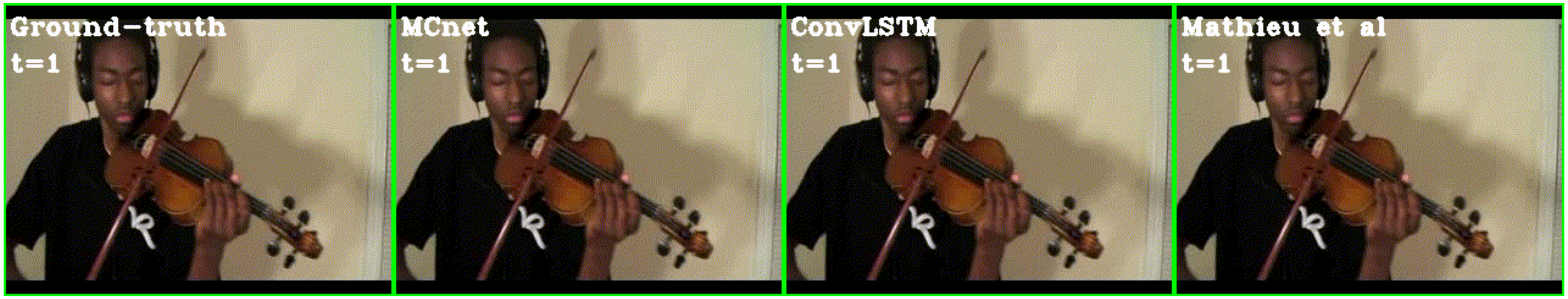


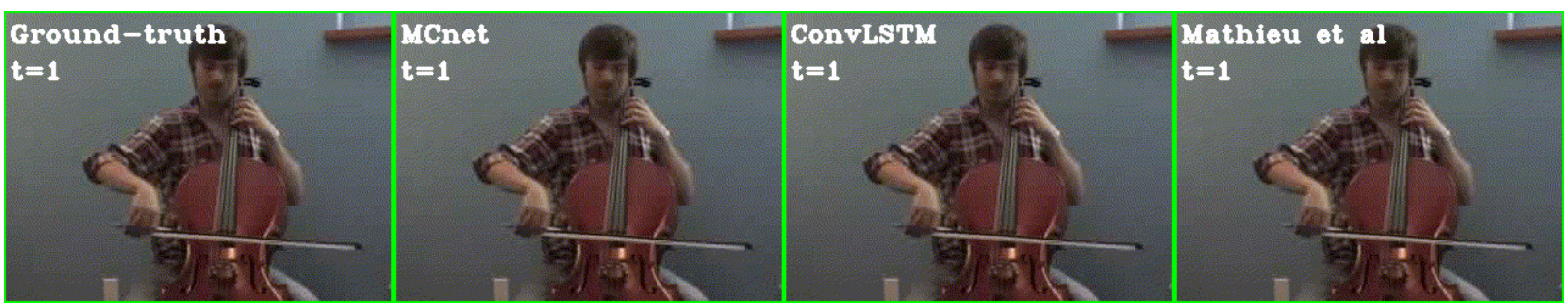




Experimental results

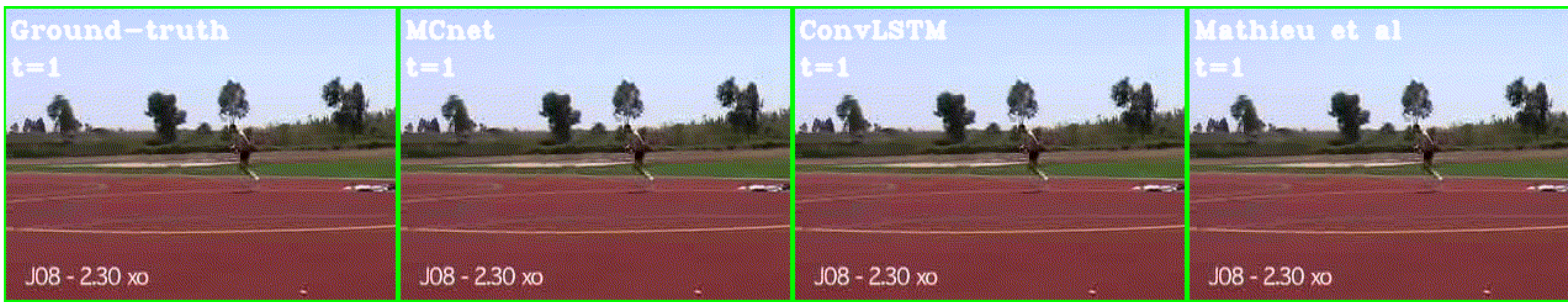
UCF-101 dataset

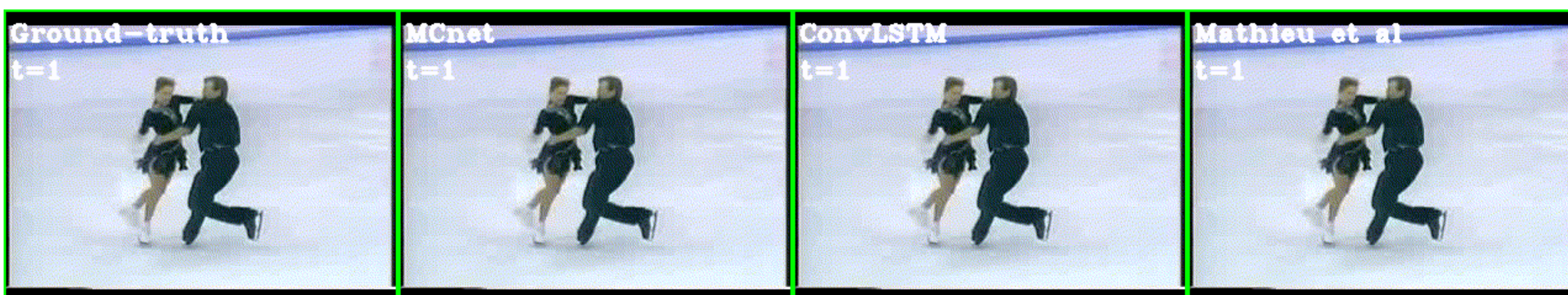




Experimental results

UCF-101 dataset





Experimental results

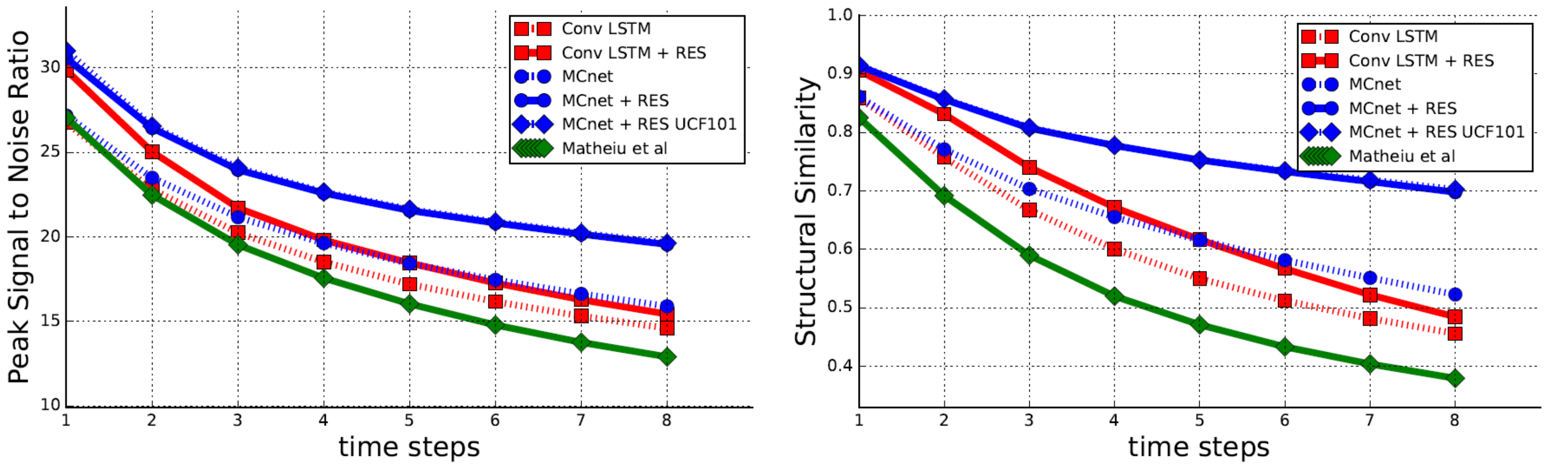
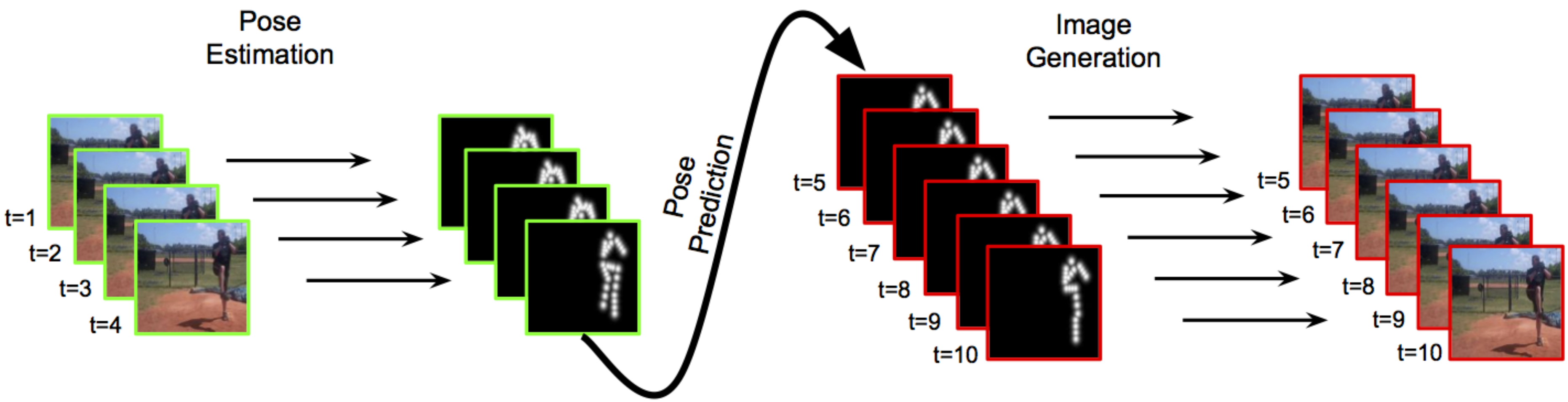


Figure 4: Quantitative comparison between our model, convolutional LSTM Shi et al. (2015), and Mathieu et al. (2015). Given 4 input frames, the models predict 8 frames recursively, one by one.

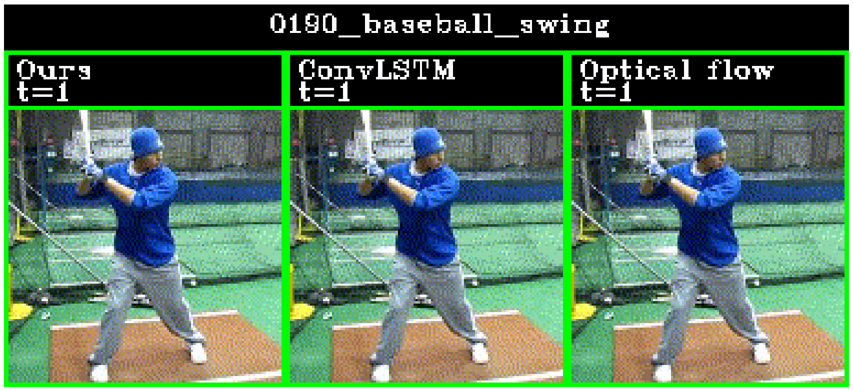
Long-term future prediction with structures

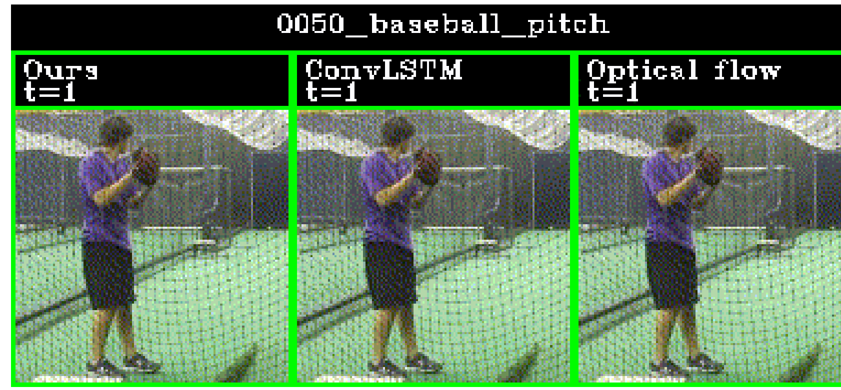


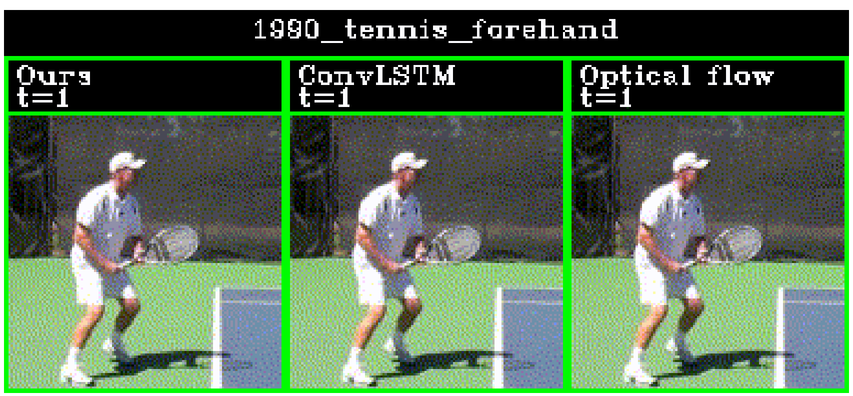
Learning to Generate Long-term Future via Hierarchical Prediction.
Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, Honglak Lee. Arxiv (coming soon)

Experimental results

- End to end prediction (Penn Action dataset)

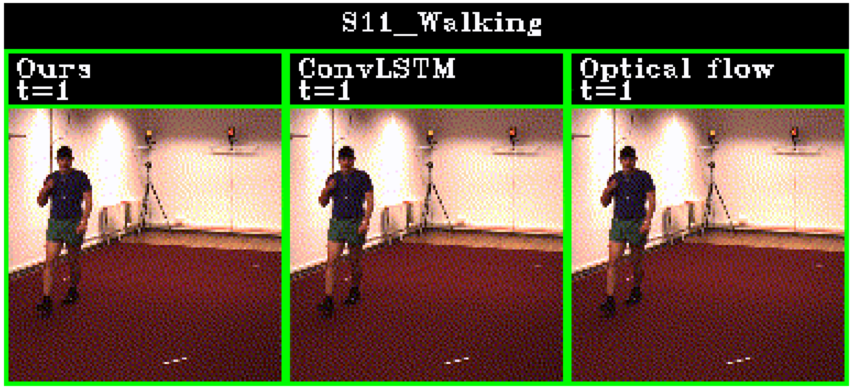


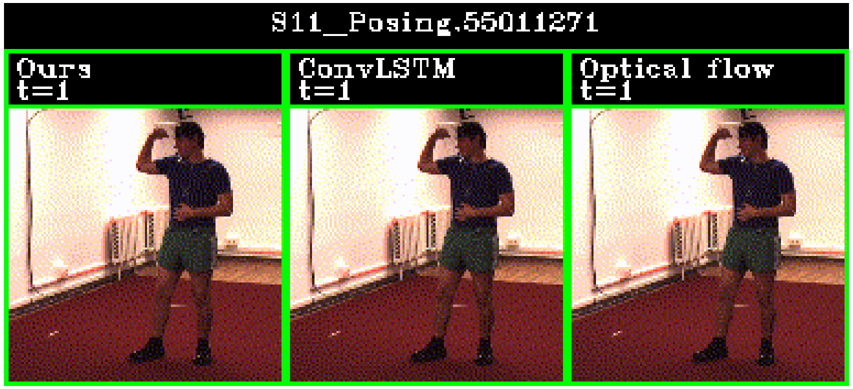


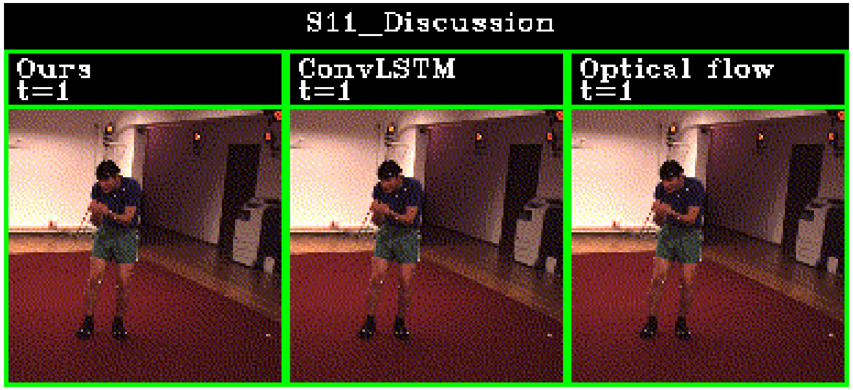


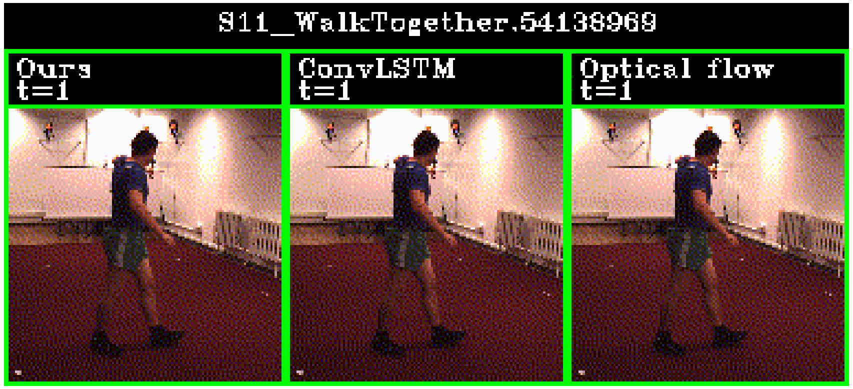
Experimental results

- End to end prediction (Human 3.6M dataset)









Experimental results

- Prediction when provided with GT landmarks

Combining Active Perception (partial observation) and Memory

- **Active Perception**: Can the agent learn to use its perceptual actions to collect useful information in partially observable environments?

- **Memory**: Can the agent remember useful information in partially observable environments?

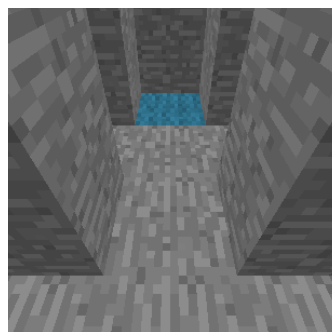
- **Generalization**: Can the agent generalize to unseen and/or larger environments given the same task?

→ These are hard to examine in the existing benchmarks.

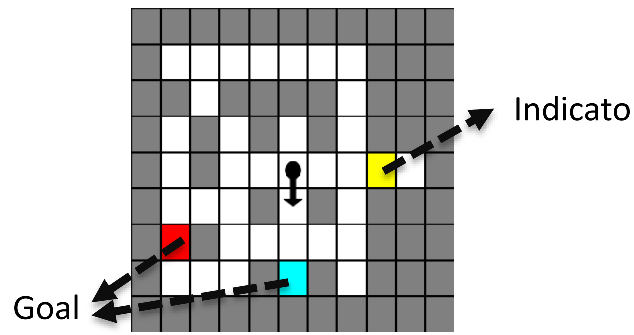
Control of Memory, Active Perception, and Action in Minecraft.
J. Oh, V. Chockalingam, S. Singh, and H. Lee. ICML 2016.

Minecraft domain

- Minecraft provides a rich environment for RL [ICML 2016]
  - Flexible 3D environment (e.g., moving, collecting, building)
  - We can define many tasks and control the level of tasks
  - Deep partial observability due to the first-person-view observations



First-person-view observation

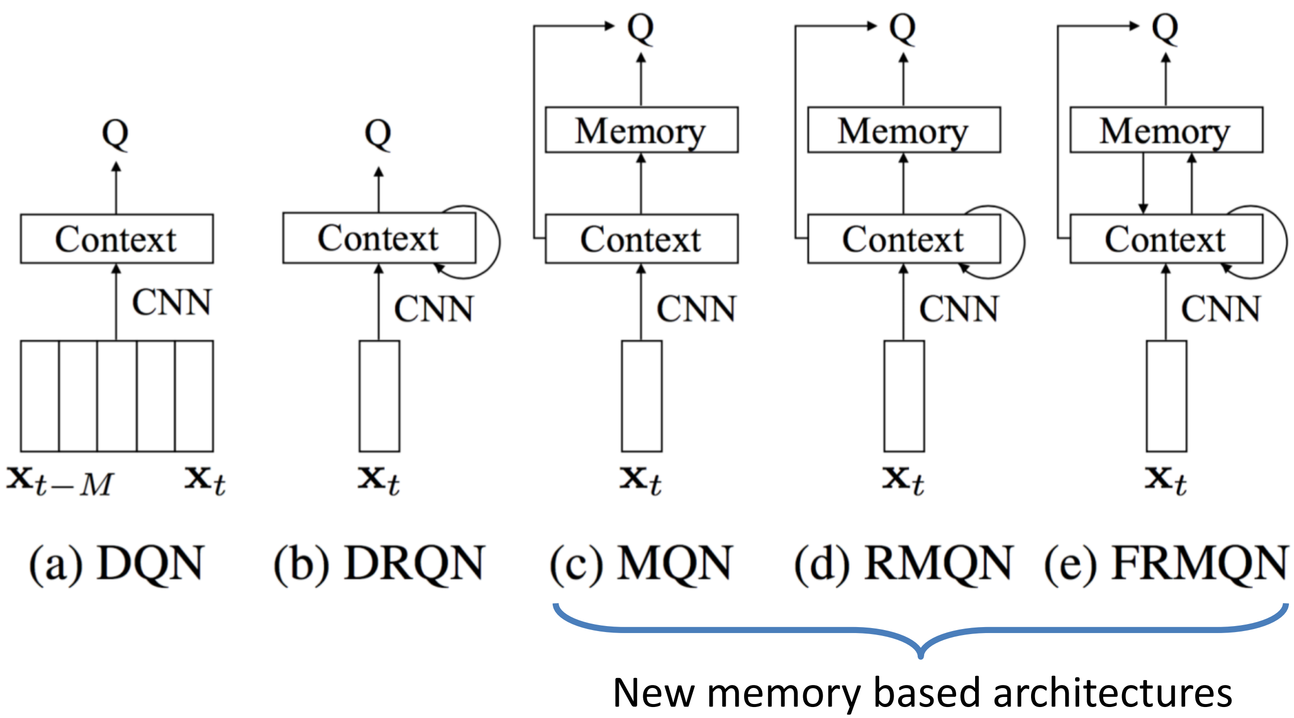


Top-down-view (not available to agent)

Control of Memory, Active Perception, and Action in Minecraft.
J. Oh, V. Chockalingam, S. Singh, and H. Lee. ICML 2016.

Related work: Cognitive Mapping and Planning for Visual Navigation. Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, Jitendra Malik, CVPR 2017

Overview of architectures

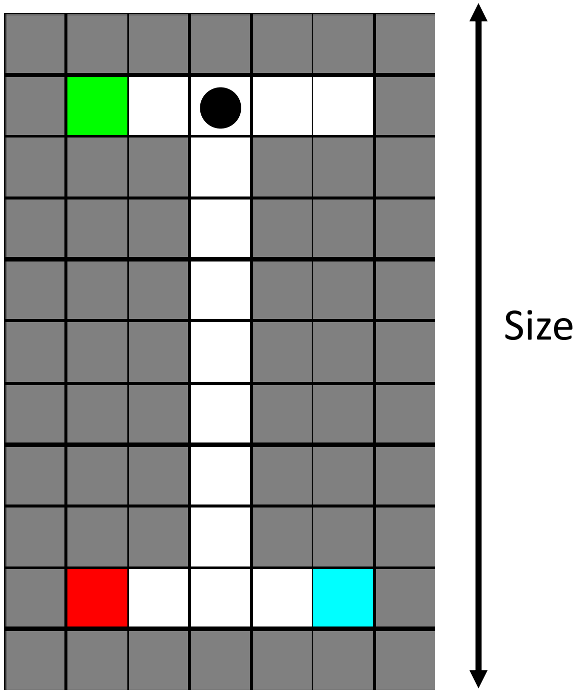


Related work:

Weston, J., Chopra, S., & Bordes, A. Memory networks. *ICLR 2015.*

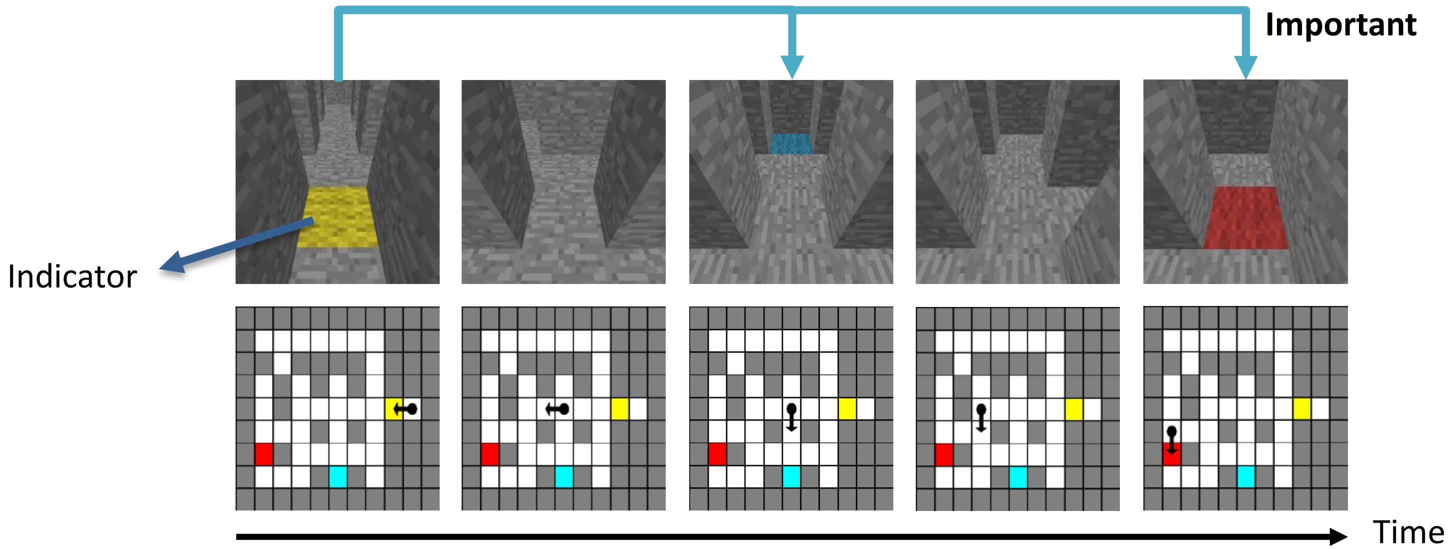
I-Maze: Task Description

- The indicator has an equal chance to be green or yellow.
  - Green indicator: Blue gives +1
  - Yellow indicator: Red gives +1
- Fixed sizes of maps ({5,7,9}) are given during training.
- Q) Can the agent generalize to unseen sizes of maps?



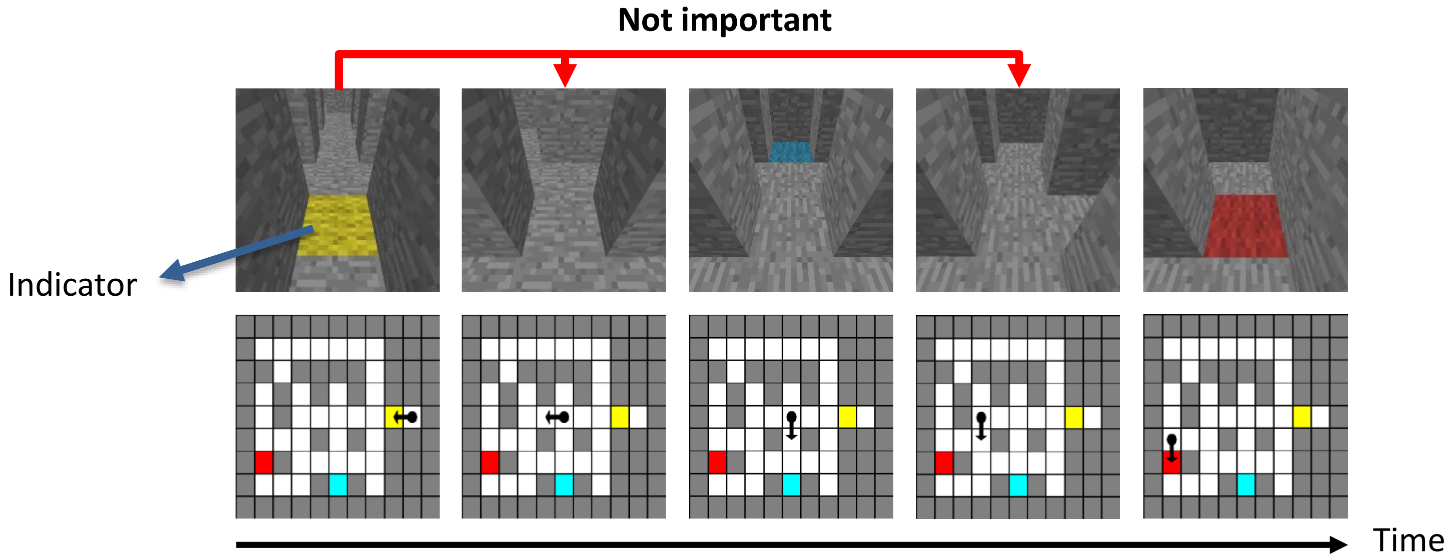
Why context-dependent memory retrieval?

- The importance of a past event depends on the current context.
- ex) the color of the indicator (yellow) is important only when the agent finds a goal block (blue/red) and decides whether to visit it or not.



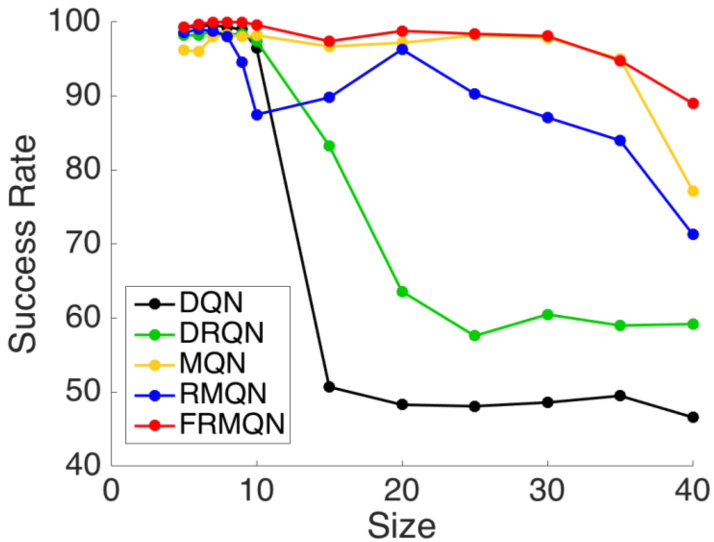
Why context-dependent memory retrieval?

- The importance of a past event depends on the current context.
- ex) the color of the indicator (yellow) is important only when the agent finds a goal block (blue/red) and decides whether to visit it or not.



I-Maze: Result

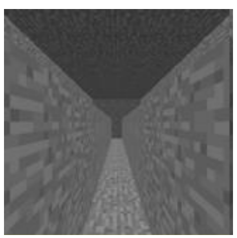
SIZE	TRAIN	DQN	DRQN	MQN	RMQN	FRMQN
4		92.1(1.5)	94.8(1.5)	87.2(2.3)	89.2(2.4)	**96.9**(1.0)
5	✓	**99.3**(0.5)	98.2(1.1)	96.2(1.0)	98.6(0.5)	**99.3**(0.7)
6		**99.4**(0.4)	98.2(1.0)	96.0(1.0)	99.0(0.4)	**99.7**(0.3)
7	✓	99.6(0.3)	98.8(0.8)	98.0(0.6)	98.8(0.5)	**100.0**(0.0)
8		99.3(0.4)	98.3(0.8)	98.3(0.5)	98.0(0.8)	**100.0**(0.0)
9	✓	99.0(0.5)	98.4(0.6)	98.0(0.7)	94.6(1.8)	**100.0**(0.0)
10		96.5(0.7)	97.4(1.1)	98.2(0.7)	87.5(2.6)	**99.6**(0.3)
15		50.7(0.9)	83.3(3.2)	**96.7**(1.3)	89.8(2.4)	**97.4**(1.1)
20		48.3(1.0)	63.6(3.7)	97.2(0.9)	96.3(1.2)	**98.8**(0.5)
25		48.1(1.0)	57.6(3.7)	**98.2**(0.7)	90.3(2.5)	**98.4**(0.6)
30		48.6(1.0)	60.5(3.6)	**97.9**(0.9)	87.1(2.4)	**98.1**(0.6)
35		49.5(1.2)	59.0(3.4)	**95.0**(1.1)	84.0(3.2)	**94.8**(1.2)
40		46.6(1.2)	59.2(3.6)	77.2(4.2)	71.3(5.0)	**89.0**(2.6)



- All architectures perform well on the training set of maps.
- Our architectures generalize better to larger I-mazes than DQN and DRQN architectures.

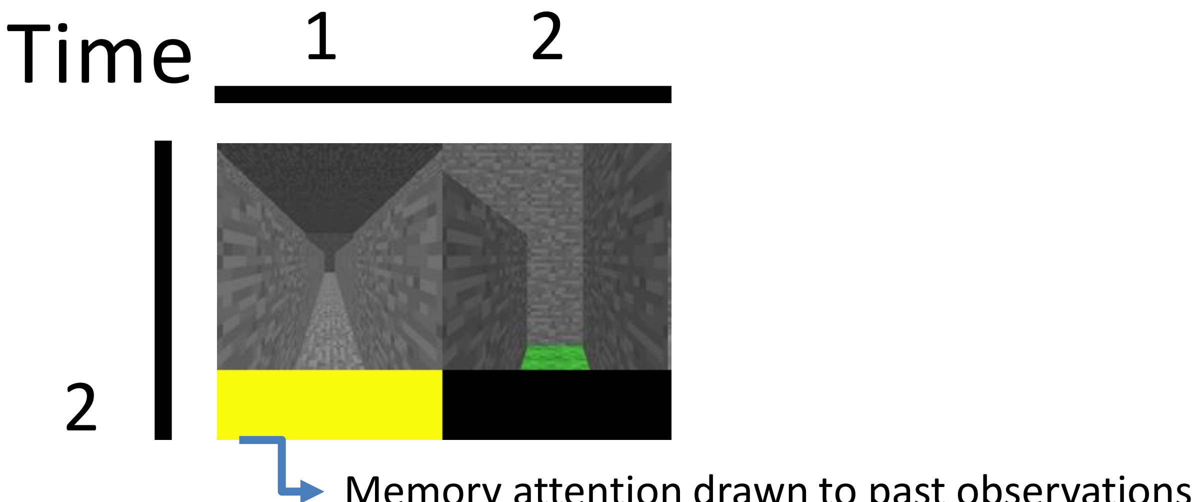
I-Maze: Memory Retrieval Visualization

Time 1



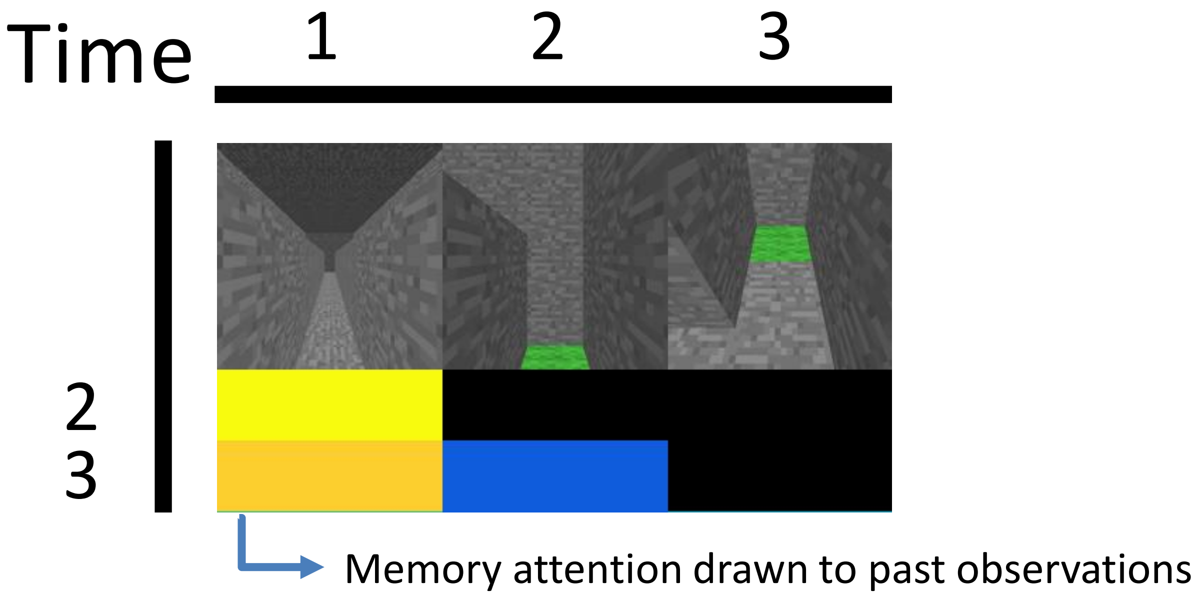
I-Maze: Memory Retrieval Visualization

- Our agent (FRMQN) looks at the indicator.



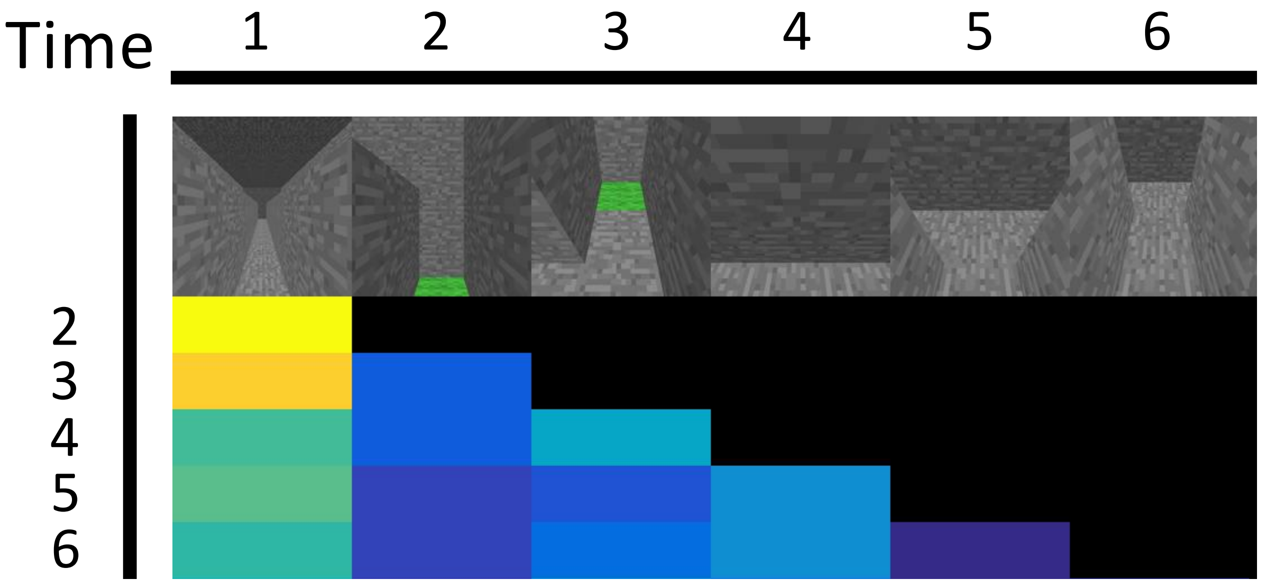
I-Maze: Memory Retrieval Visualization

- Our agent (FRMQN) looks at the indicator.



I-Maze: Memory Retrieval Visualization

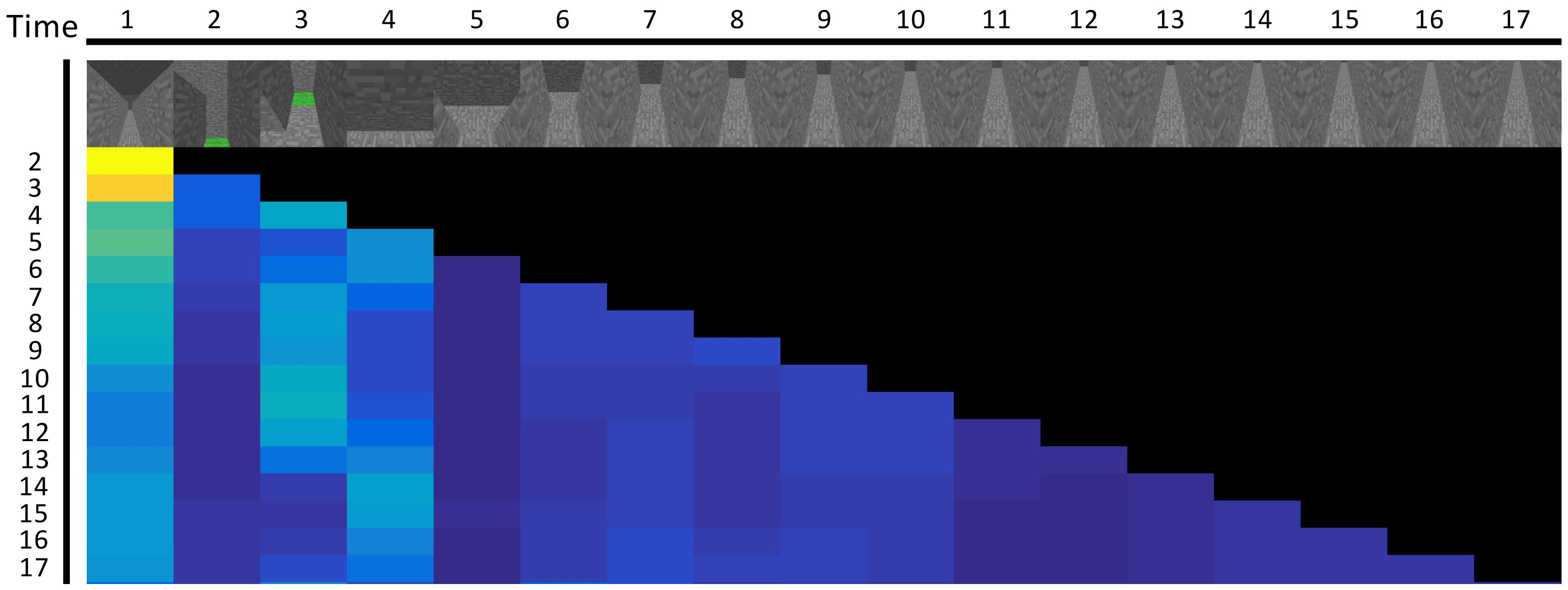
- Our agent goes to the end of the corridor.



I-Maze: Memory Retrieval Visualization

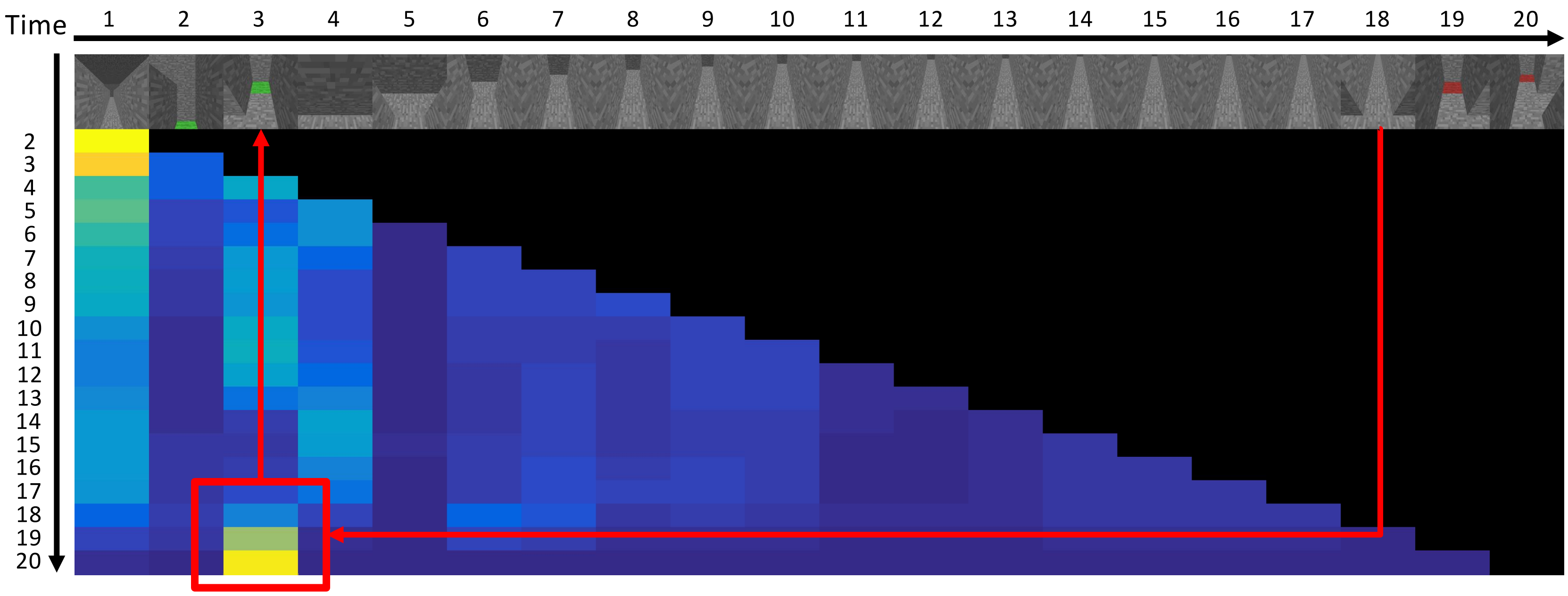
- Our agent goes to the end of the corridor.

→ It does not sharply retrieve to the indicator along the way.



I-Maze: Memory Retrieval Visualization

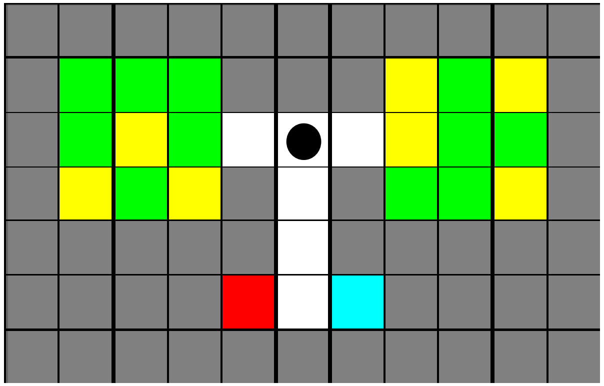
- Our agent sharply retrieves the indicator information only when it has to decide which way to go at the end of the corridor.



I-Maze: Demo

Pattern Matching: Task Description

- There are two 3x3 rooms with color patterns that have an equal chance to be identical or different.
  - If two patterns are identical: Blue gives +1
  - Otherwise: Red gives +1



- A subset of visual patterns is given during training.

- Q) Can the agent generalize to unseen visual patterns?

Pattern Matching: Result

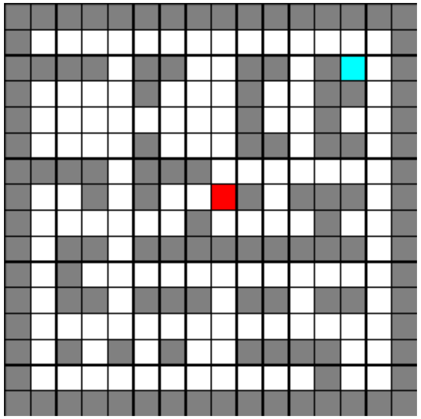
	TRAIN	UNSEEN
DQN	62.9% (±3.4%)	60.1% (±2.8%)
DRQN	49.7% (±0.2%)	49.2% (±0.2%)
MQN	99.0% (±0.2%)	69.3% (±1.5%)
RMQN	82.5% (±2.5%)	62.3% (±1.5%)
FRMQN	**100.0%** (±0.0%)	**91.8%** (±1.0%)

- DQN/DRQN/RMQN tend to learn a sub-optimal policy that goes to any goal blocks regardless of visual patterns.
- Although MQN performs well on the training maps, it fails to generalize to unseen visual patterns.
- FRMQN generalizes better to unseen maps across different runs.

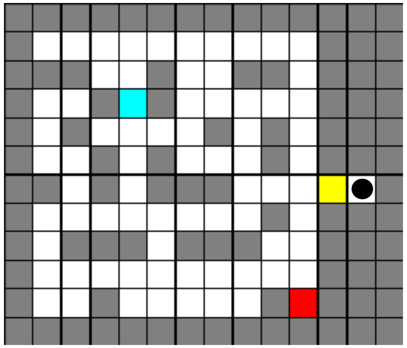
Pattern Matching: Demo

Random Maze: Task Description

- **Single Goal**
  - Blue gives +1, Red gives -1
- **Sequential Goals**
  - Red → Blue
- **Single Goal with Indicator**
  - Green indicator: Blue gives +1
  - Yellow indicator: Red gives +1
- **Sequential Goals with Indicator**
  - Green indicator: Red → Blue
  - Yellow indicator: Blue → Red



Random maze
without indicator



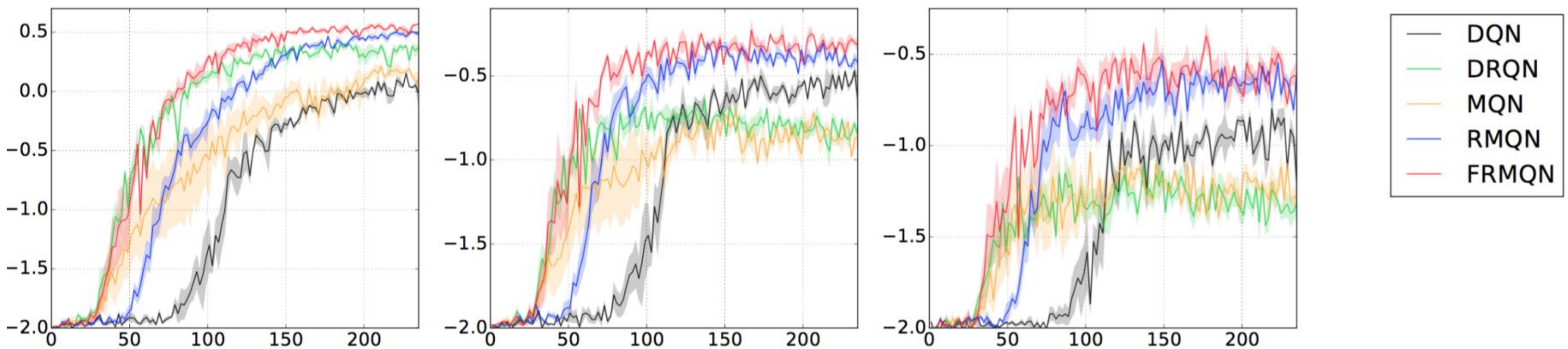
Random maze
with indicator

Random Maze: Result

Task	Type	Size	DQN	DRQN	MQN	RMQN	FRMQN
Single	train	4-8	0.31	**0.45**	0.01	**0.49**	**0.46**
	unseen	4-8	0.22	0.23	0.02	**0.30**	**0.26**
	unseen-L	9-14	**-0.28**	−0.40	−0.63	**-0.28**	**-0.28**
Seq	train	5-7	−0.60	−0.08	−0.48	**0.21**	**0.22**
	unseen	5-7	−0.66	−0.54	−0.59	**-0.13**	**-0.18**
	unseen-L	8-10	−0.82	−0.89	−0.77	**-0.43**	**-0.42**
Single+I	train	5-7	−0.04	0.23	0.11	**0.34**	0.24
	unseen	5-7	−0.41	−0.46	−0.46	**-0.27**	**-0.23**
	unseen-L	8-10	−0.74	−0.98	−0.66	**-0.39**	**-0.43**
Seq+I	train	4-6	−0.13	0.25	−0.07	0.37	**0.48**
	unseen	4-6	−0.58	−0.65	−0.71	**-0.32**	**-0.28**
	unseen-L	7-9	−0.95	−1.14	−1.04	**-0.60**	**-0.54**

- RMQN and FRMQN perform better than the other architectures on most of the tasks and maps.

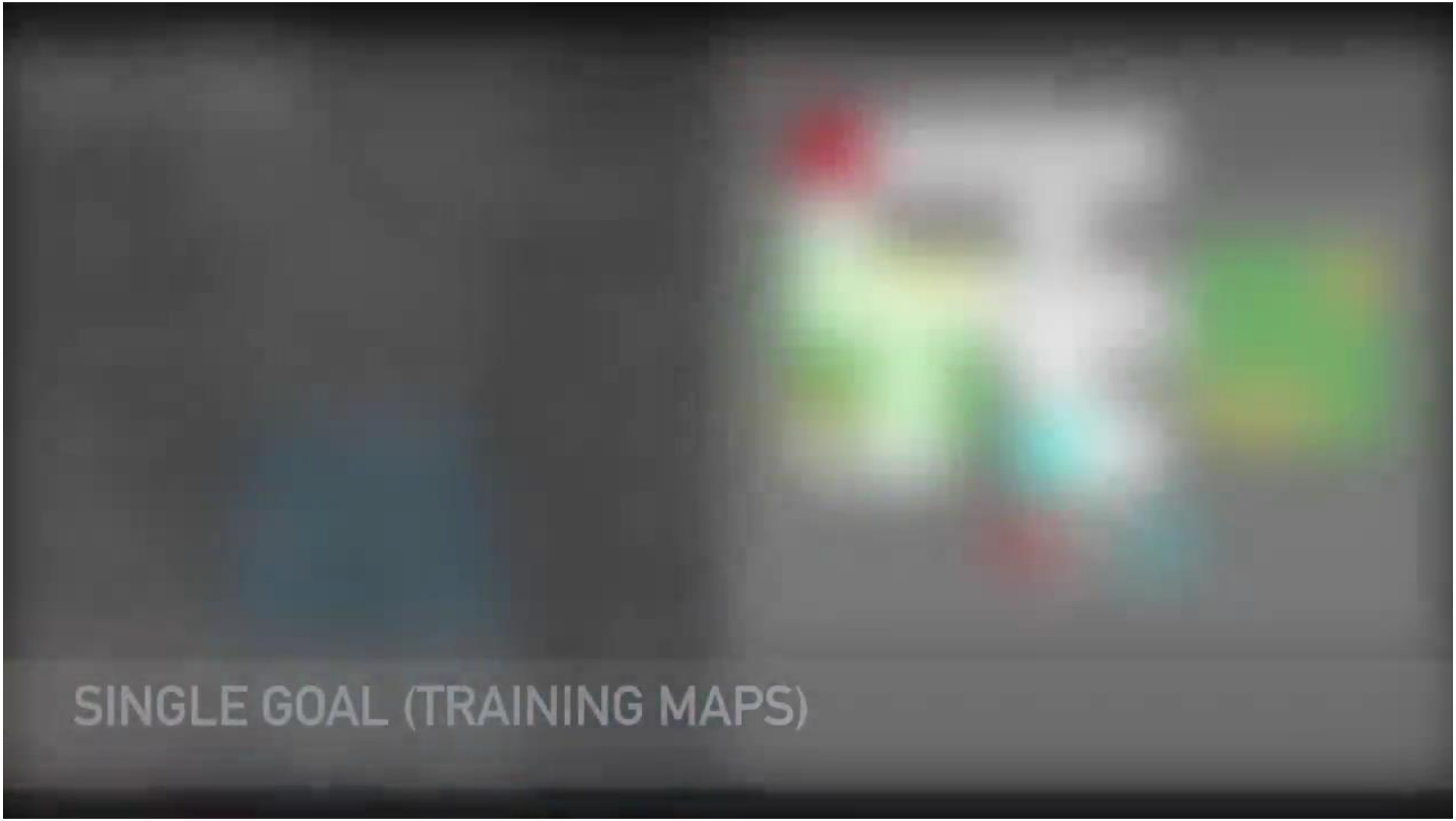
Random Maze: Result



Training maps Unseen maps Unseen/Larger maps

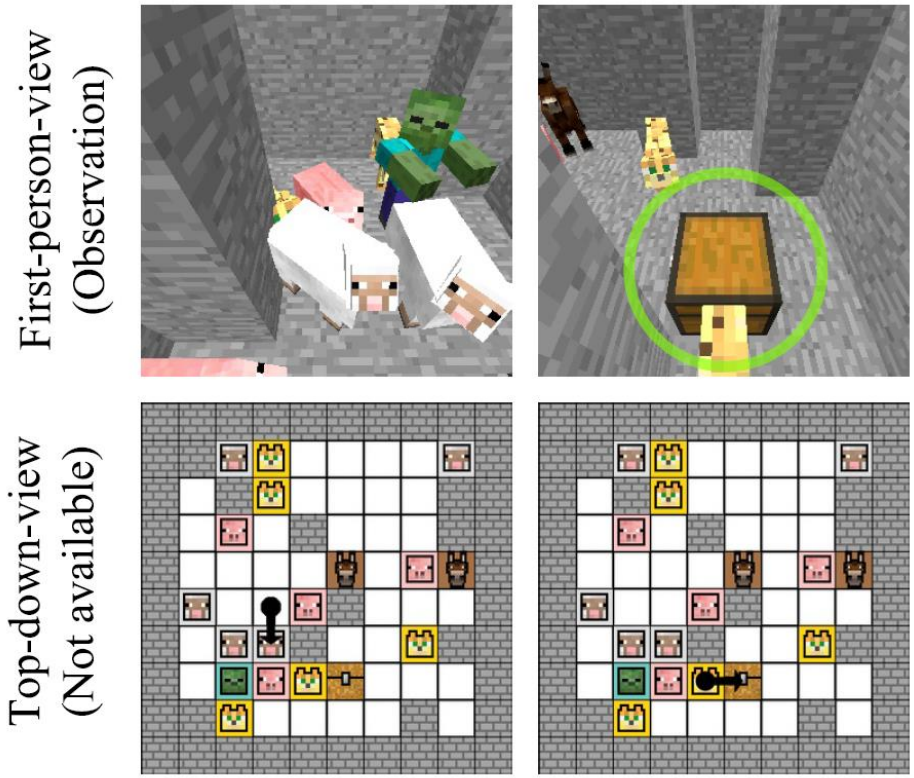
- RMQN and FRMQN perform better than the other architectures on most of the tasks and maps.
- The performance gap is larger on unseen sets of maps.

Random Maze: Demo



Hierarchical Deep RL for task generalization

- Following unseen sequence composition of instructions
- Executing unseen tasks (action/object combination)
- Needs to deal with interruptions (random events), long delayed reward



Training

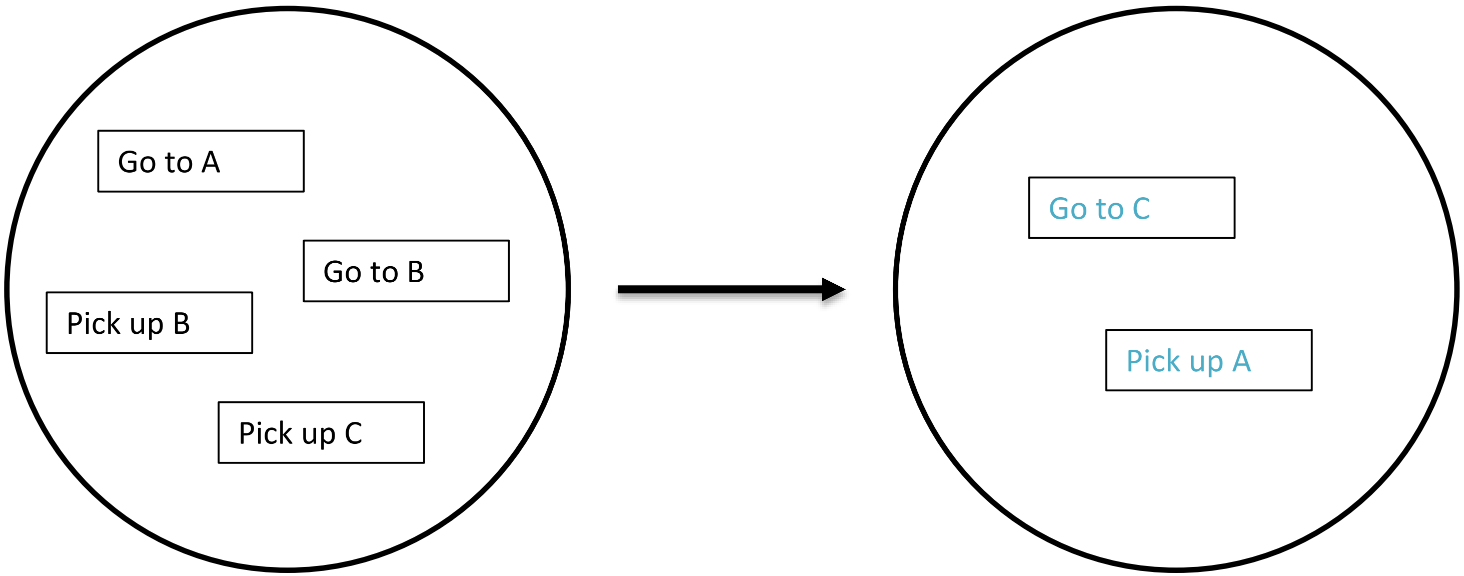
1. Visit pig
2. Pick up 3 sheep
3. Transform greenbot
4. Pick up horse

Testing

1. Pick up pig
2. Visit sheep
3. Transform cat
4. Transform 3 sheep
5. Pick up greenbot
6. Pick up 2 pig
7. Transform 2 sheep
8. Transform 2 cat

...

Types of generalizations:
1) Unseen instructions

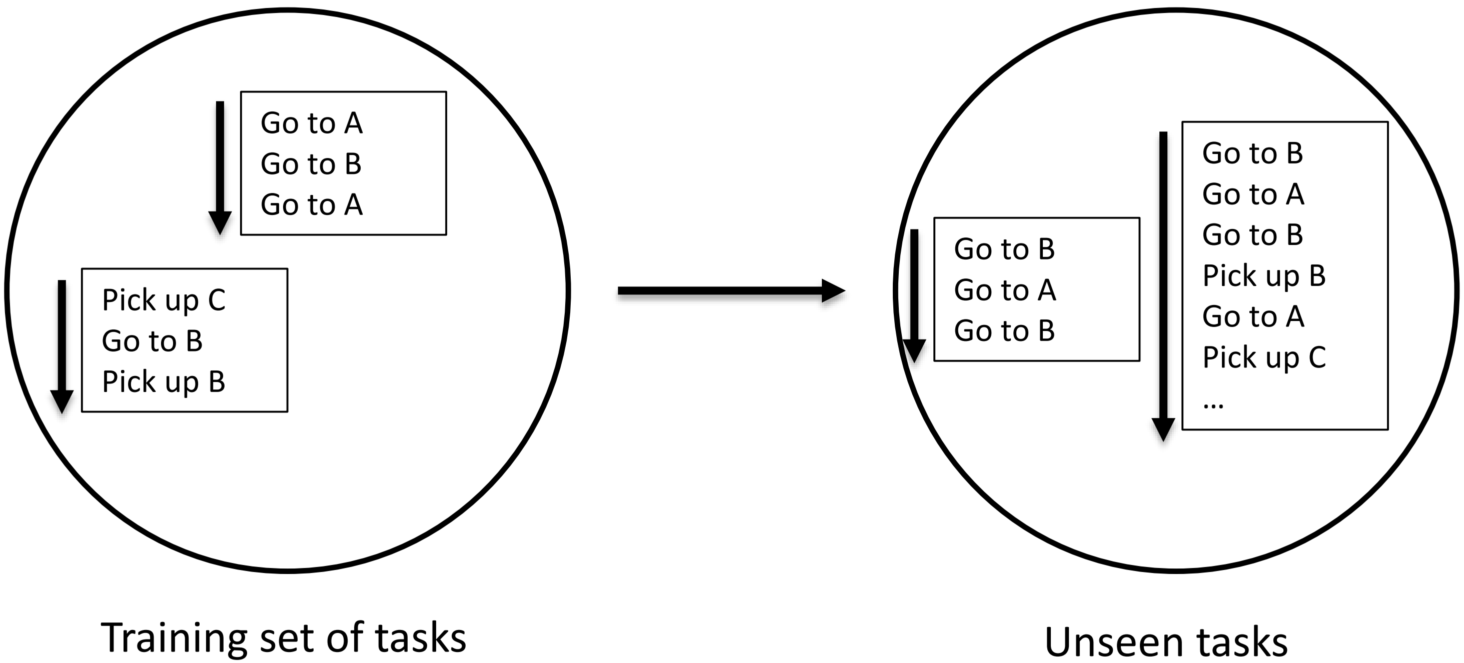


Training set of tasks

Unseen tasks

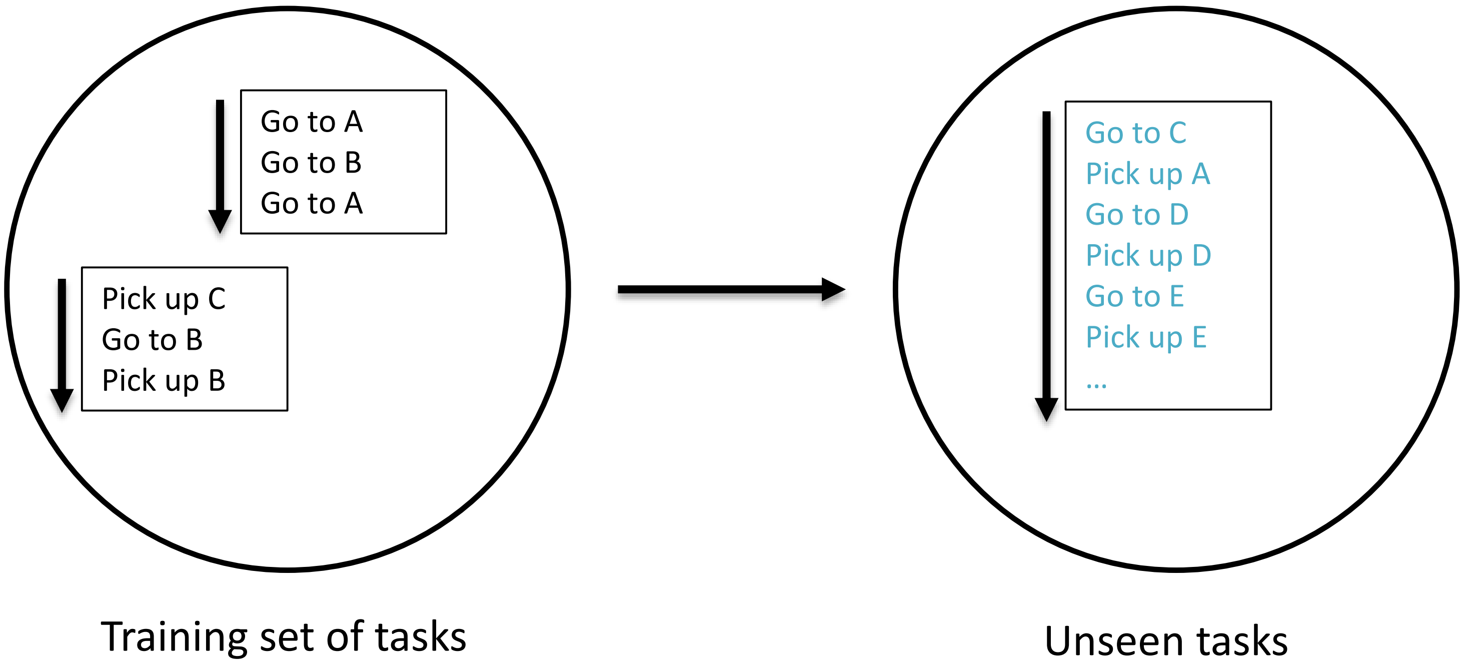
Junhyuk Oh, Satinder Singh, Honglak Lee, Pushmeet Kohli. Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. Arxiv (coming soon).

Types of generalizations:
2) Unseen/Longer sequences of instructions



Junhyuk Oh, Satinder Singh, Honglak Lee, Pushmeet Kohli. Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. Arxiv (coming soon).

Longer sequences of unseen instructions



Junhyuk Oh, Satinder Singh, Honglak Lee, Pushmeet Kohli. Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. Arxiv (coming soon).

Challenges

- Solving unseen instruction itself is a hard problem.
- Deciding when to move on to the next instruction.
  - The agent is not given which instruction to execute.
  - Should keep track of which instruction to solve.
  - Should detect when the current instruction is finished.
- Dealing with random events.
- Dealing with unbounded number of instructions.
- Delayed reward

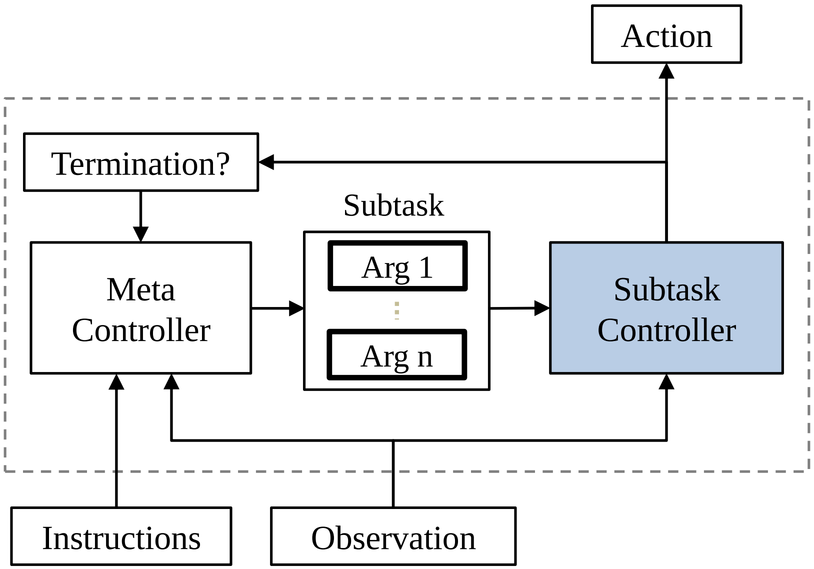
Junhyuk Oh, Satinder Singh, Honglak Lee, Pushmeet Kohli. Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. Arxiv (coming soon).

Related works

- Hierarchical RL
  - Sutton, Precup, and Singh (1999); Dietterich (2000); Parr and Russell (1997); Bacon and Precup (2015); Kulkarni et al. (2016); etc.
- Task generalization
  - Schaul et al. (2015)
- Instruction execution
  - Tellex et al. (2011; 2014); MacMahon et al. (2006); Chen and Mooney (2011); Mei et al. (2015)

Architecture Overview

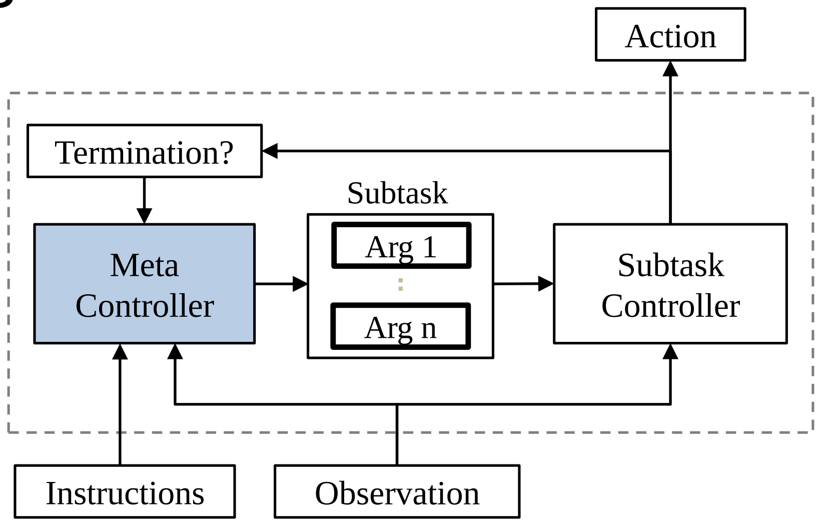
- **Subtask controller**: 1) execute primitive actions given a subtask and 2) predict whether the current subtask is finished or not.



Junhyuk Oh, Satinder Singh, Honglak Lee, Pushmeet Kohli. Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. Arxiv (coming soon).

Architecture Overview

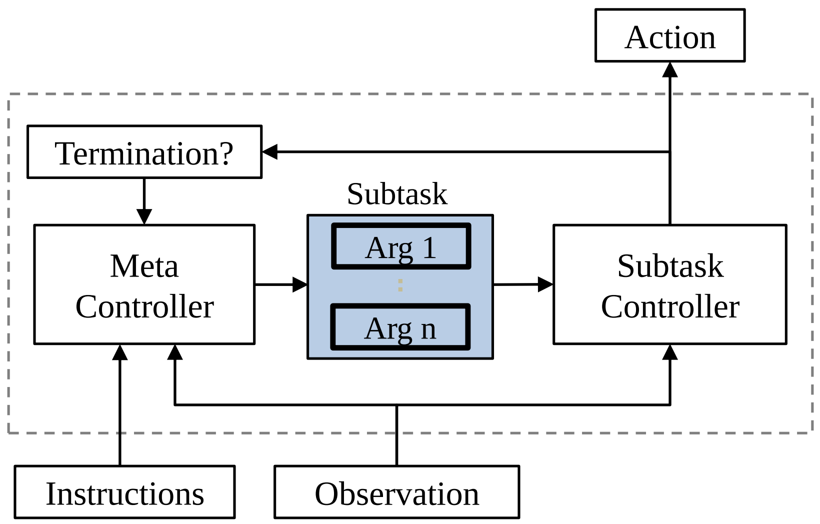
- **Subtask controller**: 1) execute primitive actions given a subtask and 2) predict whether the current subtask is finished or not.
- **Meta controller**: set subtasks given a list of instructions



Junhyuk Oh, Satinder Singh, Honglak Lee, Pushmeet Kohli. Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. Arxiv (coming soon).

Subtask Space

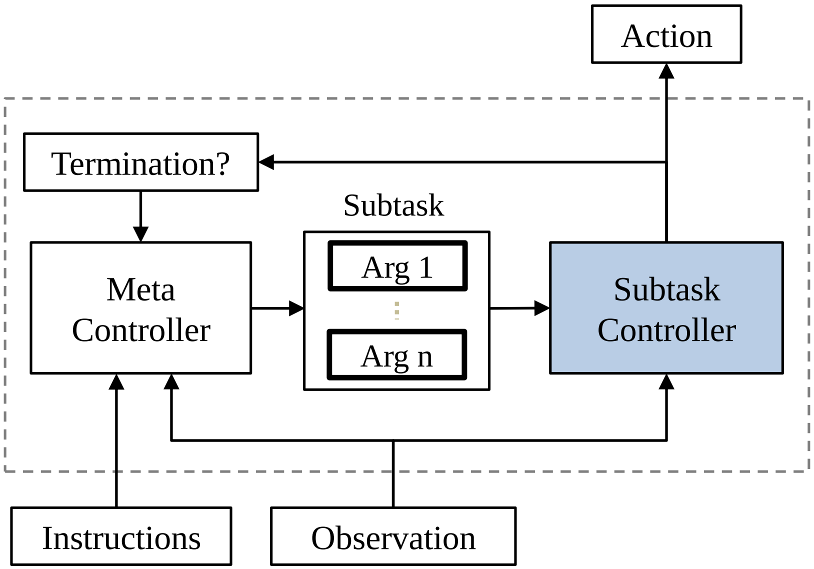
- A subtask is decomposed into several **arguments**.
- This serves as a communication protocol between two controllers.



Junhyuk Oh, Satinder Singh, Honglak Lee, Pushmeet Kohli. Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. Arxiv (coming soon).

Architecture Overview

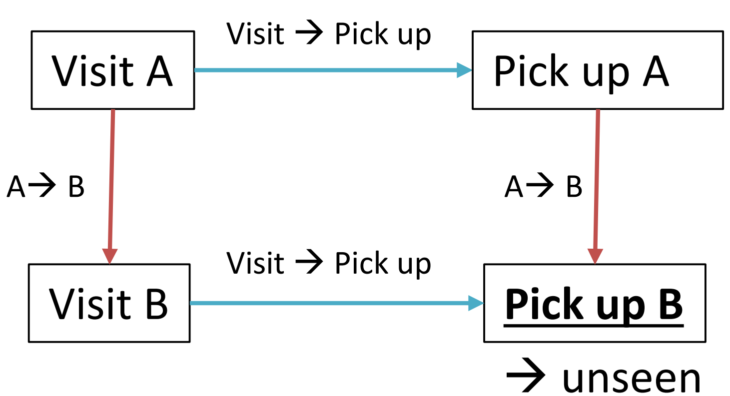
- **Subtask controller**: 1) execute primitive actions given a subtask and 2) predict whether the current subtask is finished or not.

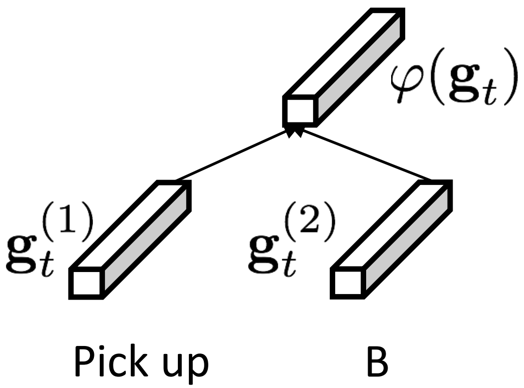


Junhyuk Oh, Satinder Singh, Honglak Lee, Pushmeet Kohli. Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning. Arxiv (coming soon).

Analogy Making Regularization

- Idea: learn a low-dimensional manifold that captures the correspondences between similar subtasks.
  - Visit A : Visit B :: Pick up A : Pick up B
  - Visit A : Visit C != Pick up A : Pick up B





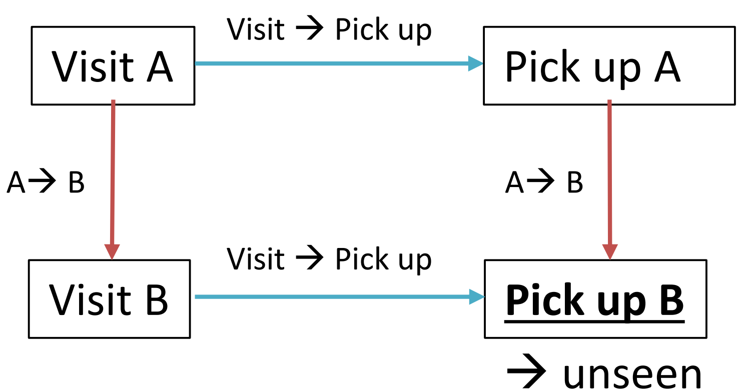
Analogy Making Regularization

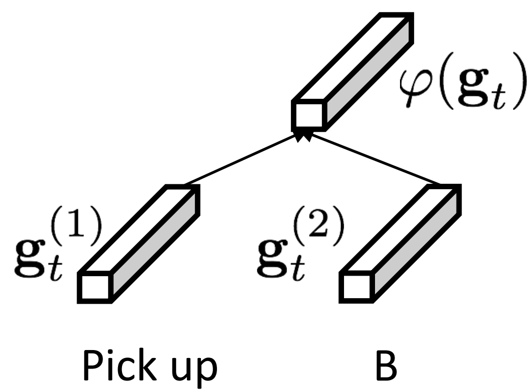
- Constraints

$$\left\|\varphi\left(\mathbf{g}_A\right)-\varphi\left(\mathbf{g}_B\right)-\varphi\left(\mathbf{g}_C\right)+\varphi\left(\mathbf{g}_D\right)\right\| \approx 0 \qquad \text{if } \mathbf{g}_A : \mathbf{g}_B :: \mathbf{g}_C : \mathbf{g}_D$$

$$\left\|\varphi\left(\mathbf{g}_A\right)-\varphi\left(\mathbf{g}_B\right)-\left(\mathbf{g}_C\right)+\varphi\left(\mathbf{g}_D\right)\right\| \geq \tau_{dis} \qquad \text{if } \mathbf{g}_A : \mathbf{g}_B \neq \mathbf{g}_C : \mathbf{g}_D$$

$$\left\|\varphi\left(\mathbf{g}_A\right)-\varphi\left(\mathbf{g}_B\right)\right\| \geq \tau_{diff} \qquad \text{if } \mathbf{g}_A \neq \mathbf{g}_B$$





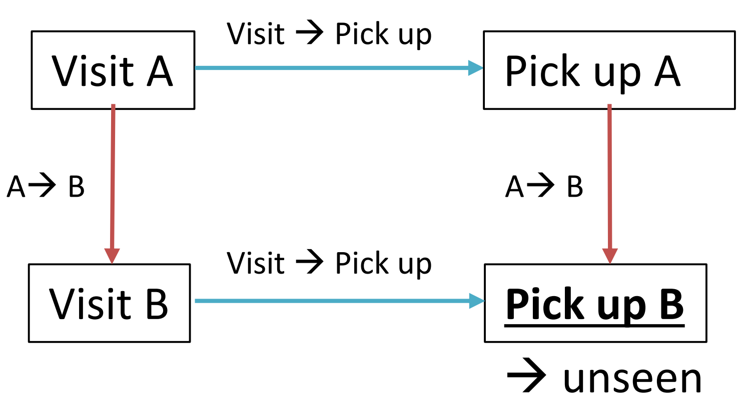
Analogy Making Regularization

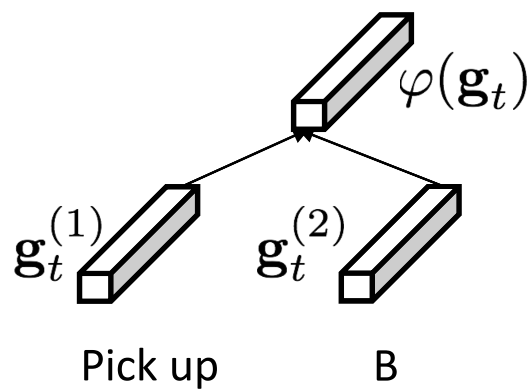
- Objective functions (Contrastive loss)

$$\mathcal{L}_{sim} = \mathbb{E}_{(\mathbf{g}_A,\mathbf{g}_B,\mathbf{g}_C,\mathbf{g}_D)\sim\mathcal{G}_{sim}}\left[\left\|\varphi\left(\mathbf{g}_A\right) - \varphi\left(\mathbf{g}_B\right) - \left(\mathbf{g}_C\right) + \varphi\left(\mathbf{g}_D\right)\right\|^2\right]$$

$$\mathcal{L}_{dis} = \mathbb{E}_{(\mathbf{g}_A,\mathbf{g}_B,\mathbf{g}_C,\mathbf{g}_D)\sim\mathcal{G}_{dis}}\left[\max\left(0, \tau_{dis} - \left\|\varphi\left(\mathbf{g}_A\right) - \varphi\left(\mathbf{g}_B\right) - \left(\mathbf{g}_C\right) + \varphi\left(\mathbf{g}_D\right)\right\|\right)^2\right]$$

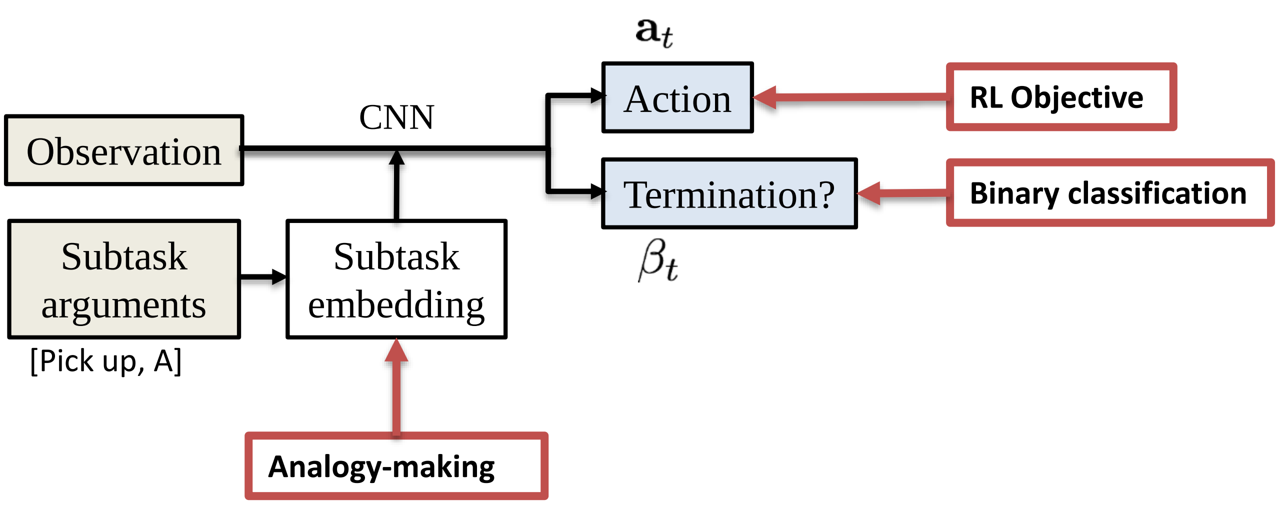
$$\mathcal{L}_{diff} = \mathbb{E}_{(\mathbf{g}_A,\mathbf{g}_B)\sim\mathcal{G}_{diff}}\left[\max\left(0, \tau_{diff} - \left\|\varphi\left(\mathbf{g}_A\right) - \varphi\left(\mathbf{g}_B\right)\right\|\right)^2\right]$$





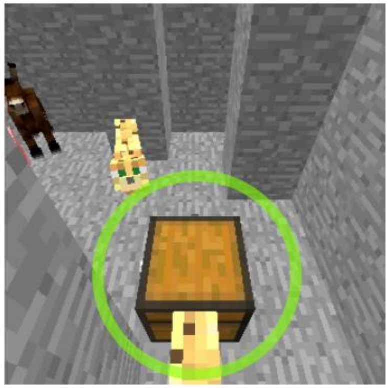
Subtask Controller: Objective function

- Objective function
  - RL objective + Analogy making + Termination prediction objective



Experimental Setting

- **Observation**: 3d first person environment with randomly generated objects



- **Actions**: primitive actions
  - Move NSWE
  - Pick up NSWE
  - Transform NSWE
  - No operation
- **Subtasks**: "action" + "target object type"
  - **<u>Visit X</u>**: should be on top of object type X
  - **<u>Pick up X</u>**: perform "pick up" to object type X
  - **<u>Transform X</u>**: perform "transform" to object type X
- **Variations:**
  - **Interact** with X: interaction with objection can vary depending on target X
  - **Repeated actions**: e.g., Pick up 3 X's

Only a subset of pairs of "action" + "object" is presented during training.

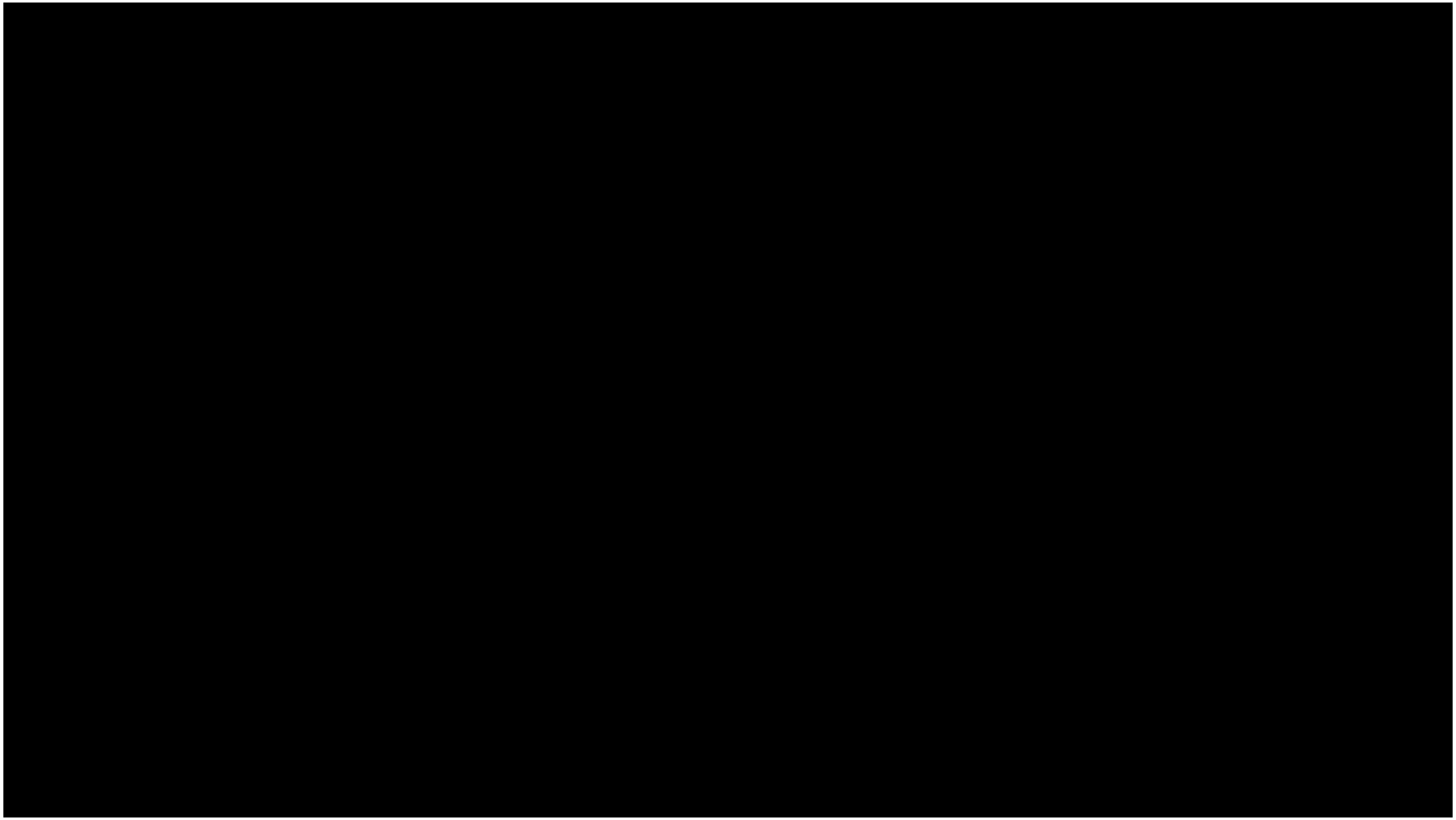
Subtask Controller: Result

- Analogy making is crucial for generalization

Scenario	Analogy	Train	Unseen
Independent	×	**0.3** (99.8%)	-3.7 (34.8%)
	✓	**0.3** (99.8%)	**0.3** (99.5%)
Object-dependent	×	**0.3** (99.7%)	-5.0 (2.2%)
	✓	**0.3** (99.8%)	**0.3** (99.7%)
Inter/Extrapolation	×	**-0.7** (97.5%)	-2.2 (24.9%)
	✓	**-0.7** (97.5%)	**-1.7** (94.5%)

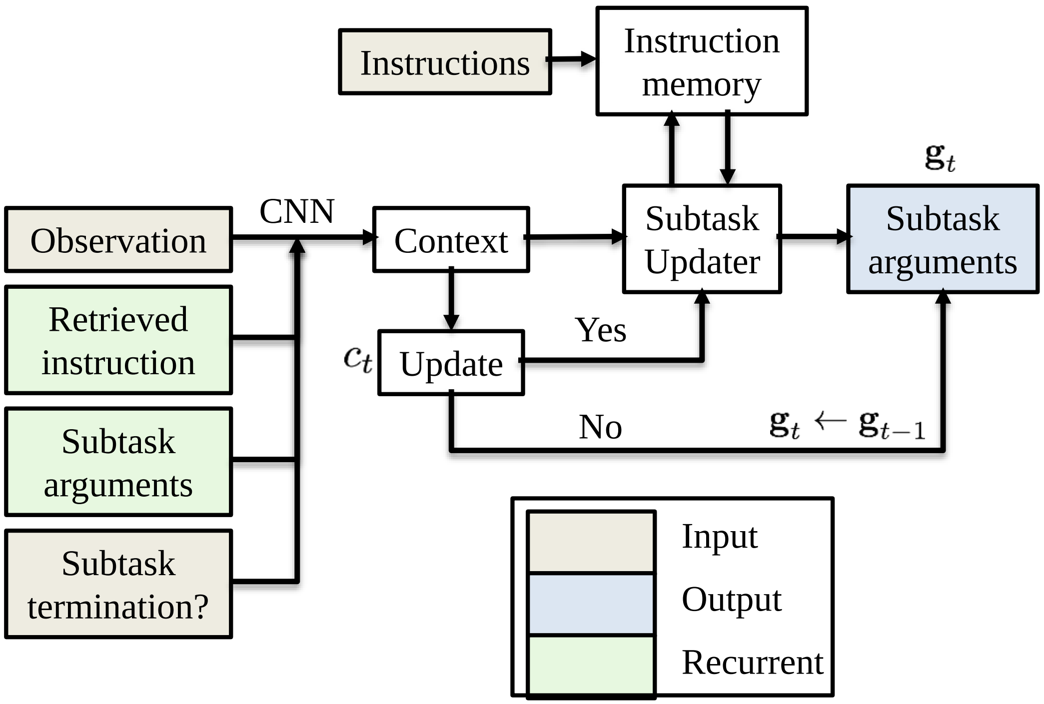
Table 1: Performance on parameterized tasks. Each entry shows 'Average reward (Success rate)'. We assume an episode is successful only if the agent successfully finishes the task and its termination predictions are correct throughout the whole episode.

Demo Video on Parameterized Tasks



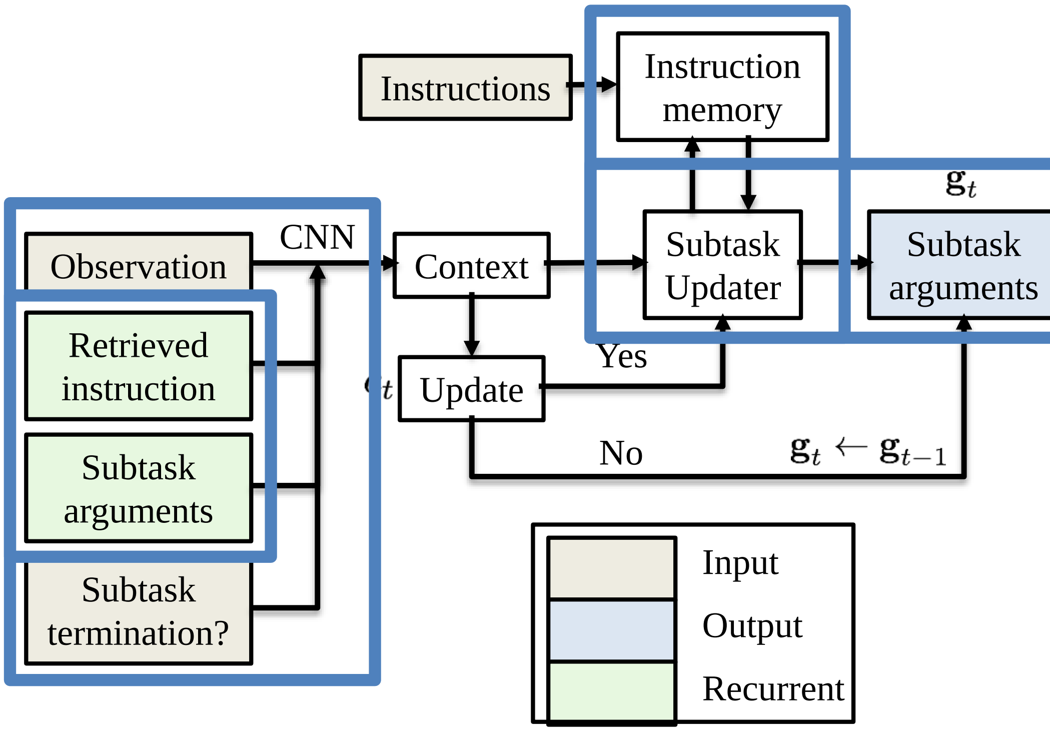
Meta Controller Architecture

- Given
  - Observation
  - Instructions
  - Subtask termination
- Do
  - Set subtask arguments



Meta Controller Architecture

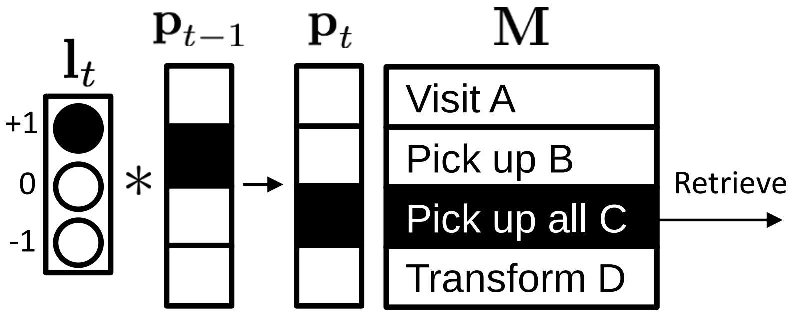
- Retrieve one instruction from the list of instructions
- Choose subtask arguments
- The retrieved instruction and selected arguments are used as input for the next time-step (recurrence)
- Parameter prediction and analogy-making are applied.



Meta Controller: Instruction Memory

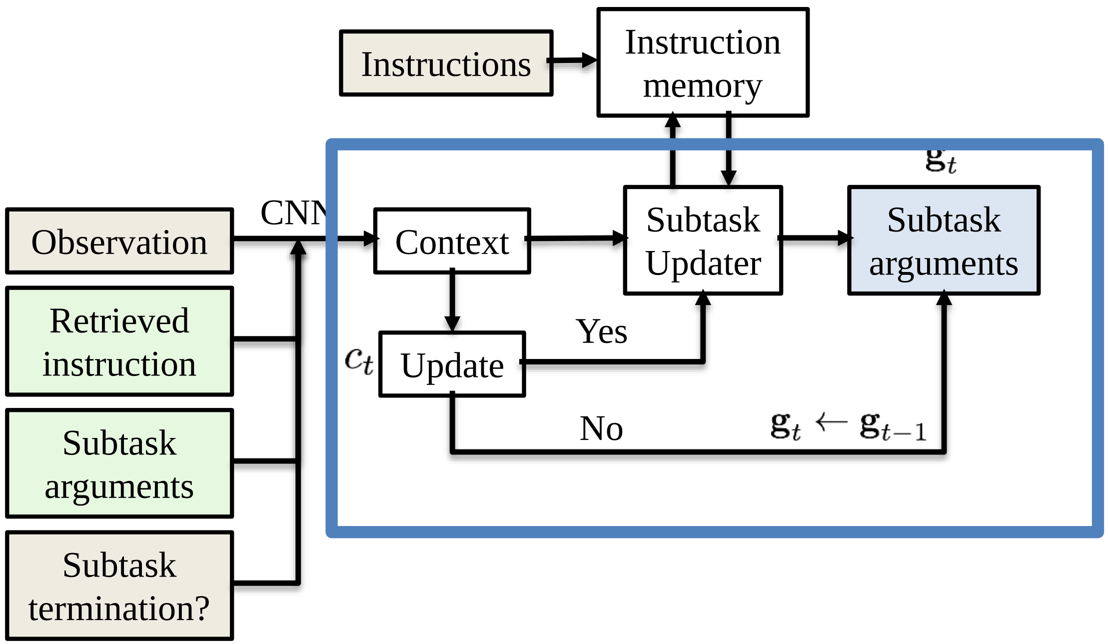
- Store all sentence embeddings into the **instruction memory**
- Maintain a pointer to a memory location
- Change the memory pointer through internal action: -1, 0, +1

- Independent of
  - The number of instructions
  - Compositions of instructions



Meta Controller: Differentiable Temporal Abstraction

- Temporal abstraction allows for infrequent updates for subtask controller



Updating the sub-tasks dynamically

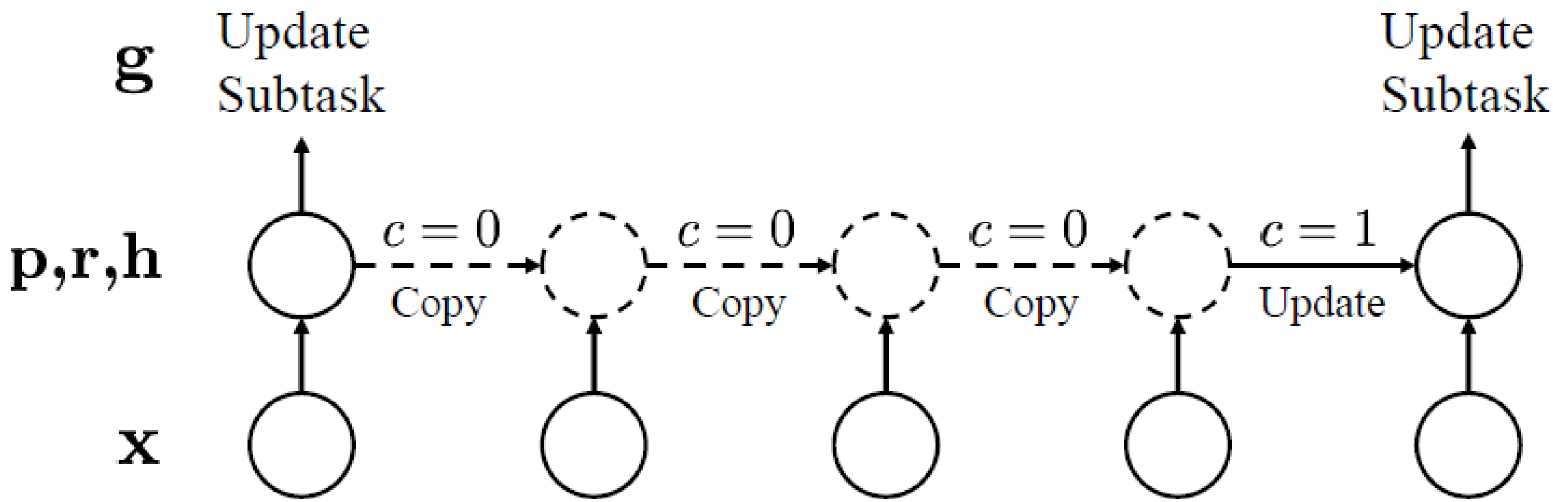
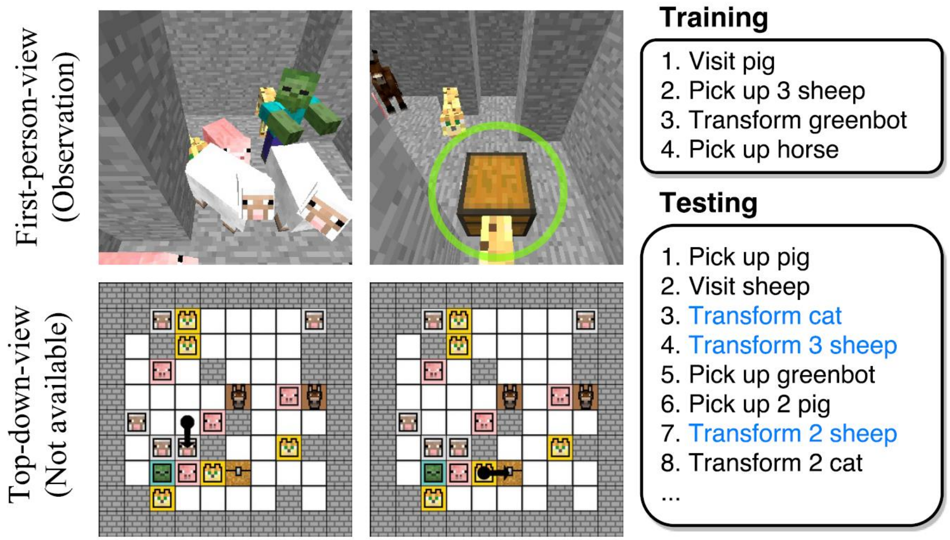


Figure 5: Unrolled illustration of the meta controller with a learned time-scale. The internal states (**p**, **r**, **h**) and the subtask (**g**) are updated only when $c = 1$. If $c = 0$, the meta controller continues the previous subtask without updating its internal states.

Experimental Setting

- **Instructions**: “action” + “target object type”
  - **<u>Visit X</u>**: should be on top of object type X
  - **<u>Pick up X</u>**: perform “pick up” to object type X
  - **<u>Transform X</u>**: perform “transform” to object type X
  - **<u>Pick up N X’s</u>**: pick up N objects with type X
  - **<u>Transform N X’s</u>**: transform N objects with type X
- Only a subset of pairs of “action” + “object” is presented during training.



Experimental Setting

- **Reward**
  - Default reward: -0.1 (time penalty)
  - Visiting water: -0.3
  - Transforming an enemy: +0.9
  - Finishing all instructions: +1.0
    - No intermediate reward

- **Random event**:
  - a box randomly appears with probability of 0:03
  - transforming a box gives +0:9 reward

Evaluation of Meta Controller

- Training set: 4 seen instructions
- Test set: 20 seen or unseen instructions

	Train	Test (Seen)	Test (Unseen)
Length of instructions	4	20	20
Flat	-7.1 (1%)	-63.6 (0%)	-62.0 (0%)
Hierarchical-Long	-5.8 (31%)	-59.2 (0%)	-59.2 (0%)
Hierarchical-Short	-3.3 (83%)	-53.4 (23%)	-53.6 (18%)
Hierarchical-Dynamic	**-3.1** (95%)	**-30.3** (75%)	**-38.0** (56%)

Baselines:

- **Flat**: directly chooses primitive actions without using the parameterized skill.
- **Hierarchical-Long**: meta controller can update the subtask only when the current subtask is finished. Similar to (Kulkarni et al., 2016; Tessler et al., 2016).
- **Hierarchical-Short**: meta controller updates the subtask at every time-step.

Summary

- Deep Reinforcement Learning can benefit from building models, memory and hierarchy
- Forward prediction: can be useful for better exploration and possibly planning
- Memory: handle partial observability. Relevant to robotic agents.
- Hierarchical RL: Temporal abstraction and analogy making is beneficial for multi-task generalization and instruction execution