Learning Discrete World Models for Heuristic Search

Forest Agostinelli
Misagh Soltani

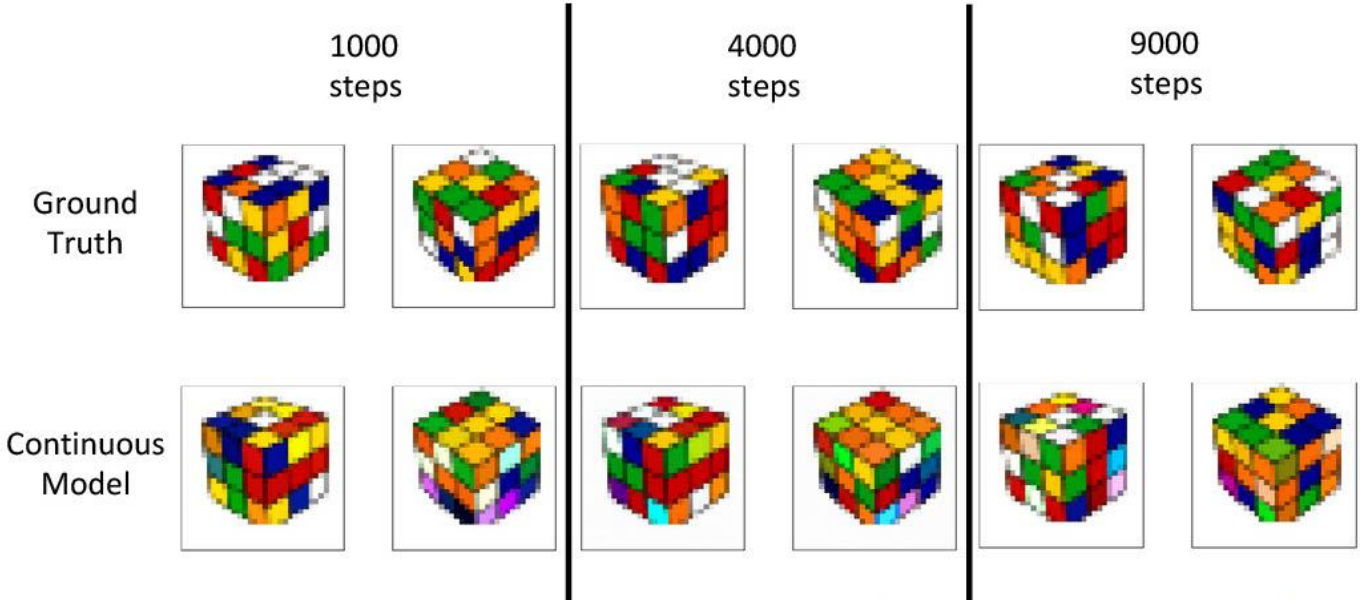
Presenter:
Misagh Soltani
msoltani@email.sc.edu

UNIVERSITY OF South Carolina
Molinaroli College of Engineering and Computing

AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

Motivation

- Planning is crucial for solving sequential decision-making problems, but it requires a state-transition function, also known as a world model.
- In domains where the world model is unknown, model-based reinforcement learning can be used to learn it.
- Continuous world models face two major challenges:
  - Lack of state re-identification
  - Model degradation



Agostinelli, Forest, Misagh, Soltani. "Learning Discrete World Models for Heuristic Search". Reinforcement Learning Journal 1. (2024).

Our Approach - DeepCubeAI

To solve these problems, we introduce DeepCubeAI (DeepCubeA + “Imagination”):

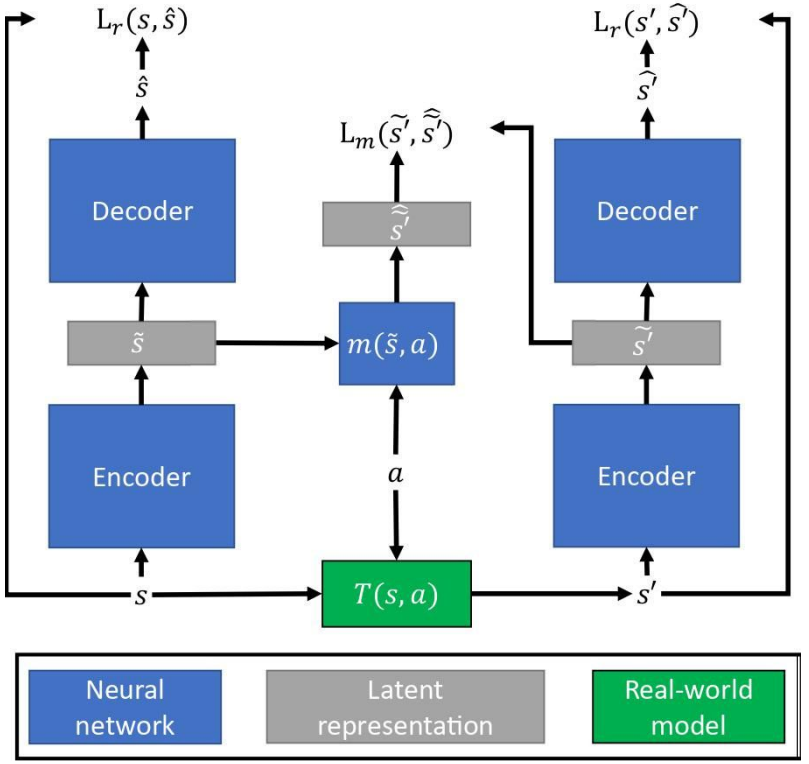
- A domain-independent method for training domain-specific heuristic functions that generalize across problem instances.

- DeepCubeAI consists of three key components:
  a. Discrete world model
     - Learns a world model that represents states in a discrete latent space.
       - Small errors in prediction can be corrected by simply rounding
       - Can re-identify states by comparing two vectors
  b. Heuristic function
     - Uses RL to learn a heuristic function that generalizes over start and goal states
  c. Search
     - Combines learned model and learned heuristic function with heuristic search to solve problems.

Agostinelli, Forest, Misagh, Soltani. "Learning Discrete World Models for Heuristic Search". Reinforcement Learning Journal 1. (2024).

Learning Discrete World Models

- Encoder
  - Maps the state to a discrete representation
  - To allow training with gradient descent, use a straight through estimator
- Decoder
  - Maps the discrete representation to the state
  - Ensures the discrete representation is meaningful
- Environment model
  - Maps discrete states and actions to next discrete state



Agostinelli, Forest, Misagh, Soltani. "Learning Discrete World Models for Heuristic Search". Reinforcement Learning Journal 1. (2024).

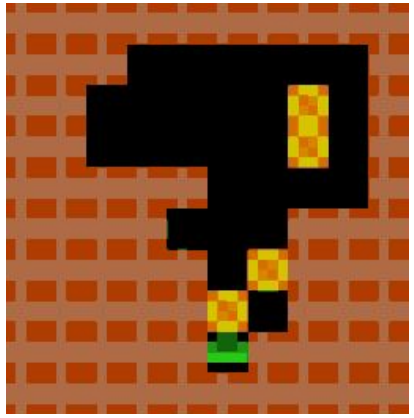
Learning Discrete World Models

- Another benefit of the discrete world model
  - We can see the percentage of the bits that match

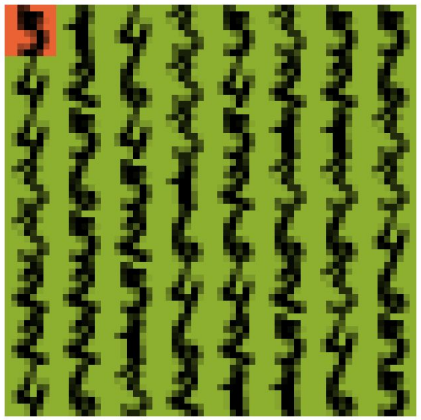
```
-------
Train
loss: 8.30E-07, l_recon: 8.16E-07, l_env: 1.43E-04, %on: 33.33, eq_bit: 99.98, eq_bit_min: 98.96, eq: 98.00
Validation
loss: 1.45E-06, l_recon: 1.45E-06, l_env: 1.49E-05, %on: 33.33, eq_bit: 100.00, eq_bit_min: 100.00, eq: 100.00
Itr: 11300, lr: 9.92E-04, env_coeff: 0.000100, times - all: 1.59

-------
Train
loss: 1.18E-05, l_recon: 1.18E-05, l_env: 1.35E-05, %on: 33.33, eq_bit: 100.00, eq_bit_min: 100.00, eq: 100.00
Validation
loss: 1.89E-06, l_recon: 1.89E-06, l_env: 1.23E-05, %on: 33.33, eq_bit: 100.00, eq_bit_min: 100.00, eq: 100.00
Itr: 11400, lr: 9.92E-04, env_coeff: 0.000100, times - all: 1.60
```


Agostinelli, Forest, Misagh, Soltani. "Learning Discrete World Models for Heuristic Search". Reinforcement Learning Journal 1. (2024).

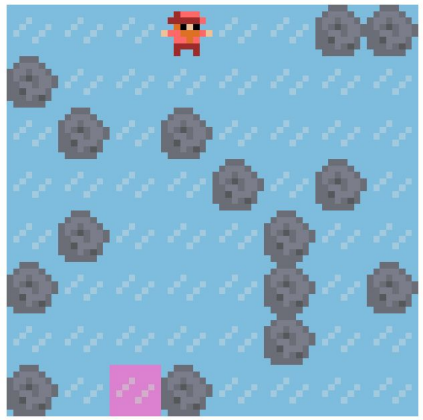
Test Environments



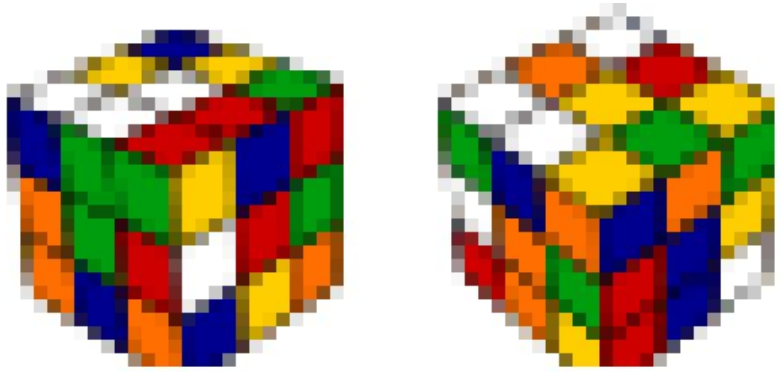
Sokoban



DigitJump



IceSlider

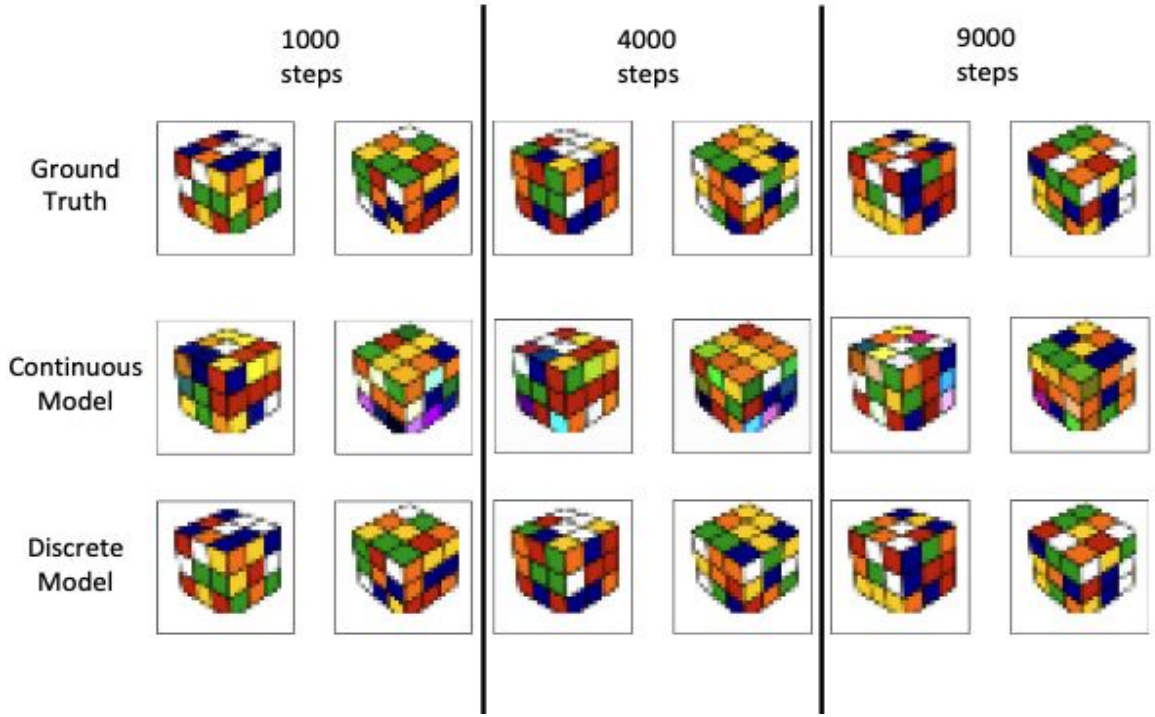


Rubik's Cube

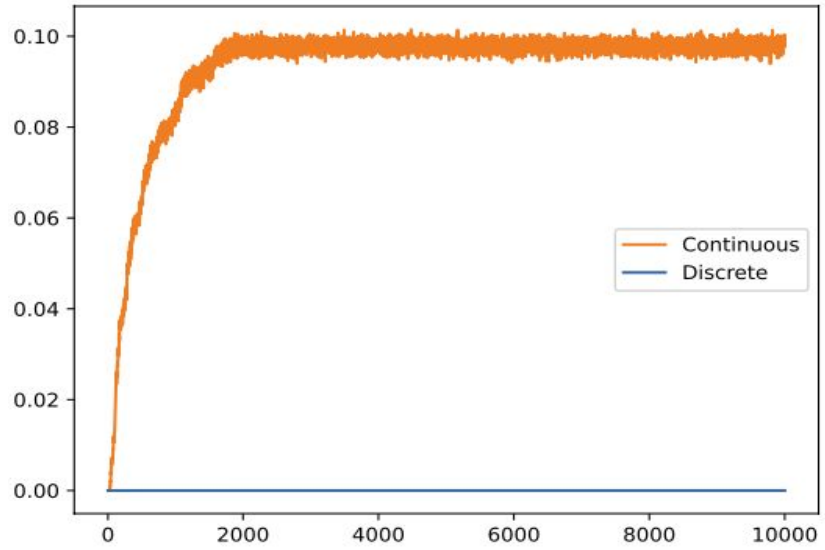
Agostinelli, Forest, Misagh, Soltani. "Learning Discrete World Models for Heuristic Search". Reinforcement Learning Journal 1. (2024).

Discrete vs Continuous Model Performance

a. A visualization of the reconstructions for models with continuous and discrete latent states at different timesteps.
b. Mean squared reconstruction error as a function of timestep.



(a)



(b)

Rubik's Cube

Agostinelli, Forest, Misagh, Soltani. "Learning Discrete World Models for Heuristic Search". Reinforcement Learning Journal 1. (2024).

Heuristic Learning and Search with Discrete Model

- Use offline data and the learned world model to generate training data
- Heuristic learning: Q-learning with hindsight experience replay
  - Results in a domain-independent algorithm for training domain-specific heuristic functions that generalize across problem instances
- Heuristic search: Q* search [1]
  - A variant of A* search for DQNs
  - Q* search can compute the heuristic values for all next states with a single pass through a DQN.
  - In practice, Q* search has been shown to perform similar to A* search while being orders of magnitude faster and more memory efficient.

1. Agostinelli, Forest, et al. 2024. "Q Search: Heuristic Search with Deep Q-Networks."*

Problem Solving Performance

Domain	Solver	Len	Opt	Nodes	Secs	Nodes/Sec	Solved
RC	PDBs$^+$	20.67	**100.0%**	2.05E+06	2.20	1.79E+06	100%
	DeepCubeA	21.50	60.3%	6.62E+06	24.22	2.90E+05	100%
	Greedy	-	0%	-	-	-	0%
	DeepCubeAI	22.85	19.5%	2.00E+05	6.21	3.22E+04	**100%**
RC_{rev}	Greedy	-	0%	-	-	-	0%
	DeepCubeAI	22.81	**21.92%**	2.00E+05	6.30	3.18+04	**99.9%**
Sokoban	LevinTS	39.80	-	6.60E+03	-	-	100%
	LevinTS (*)	39.50	-	5.03E+03	-	-	100%
	LAMA	51.60	-	3.15E+03	-	-	100%
	DeepCubeA	32.88	-	1.05E+03	2.35	5.60E+01	100%
	Greedy	29.55	-	-	1.68	-	41.9%
	DeepCubeAI	33.12	-	3.30E+03	2.62	1.38E+03	**100%**
IceSlider	PPGS	-	-	-	-	-	97.0%
	Greedy	9.83	84.78%	-	0.03	-	46.0%
	DeepCubeAI	9.85	**100%**	31.84	0.09	3.50E+02	**100%**
DigitJump	PPGS	-	-	-	-	-	99.0%
	Greedy	5.72	88.89%	-	0.04	-	90.0%
	DeepCubeAI	5.83	**96.0%**	8.97	0.06	1.40E+02	**100%**

Forest Agostinelli, Misagh Soltani. "Learning Discrete World Models for Heuristic Search". Reinforcement Learning Journal 1. (2024).

Future Work

- Address rare mistakes in identifying latent goal states by training a DNN to correct slightly corrupted latent states or using *Hallucinated Replay* for self correction.[1]

- Improve goal specification in environments where goal images are difficult to generate, potentially using formal logic to specify goals without generating goal images.[2]

- Extend benefits of discrete models to stochastic and partially observable robotic tasks, enhancing exploration for training and obtaining more lookahead during search.[3]

1. *Talvitie, Erik. "Self-correcting models for model-based reinforcement learning. " Proceedings of the AAAI conference on AI. Vol. 31. No. 1. 2017.*
2. *Agostinelli, Forest, et al. "Specifying goals to deep neural networks with answer set programming." Proceedings of the ICAPS Vol. 34. 2024.*
3. *Kaiser, Lukasz, et al. "Model-based reinforcement learning for atari." arXiv preprint arXiv:1903.00374 (2019).*

Thank you for your attention!

Scan to access the paper

Scan for the GitHub repository

Scan to connect on LinkedIn

Misagh Soltani
Research Assistant, AI Institute of University of South Carolina
Ph.D. Student, University of South Carolina
msoltani@email.sc.edu

Molinaroli College of Engineering and Computing
UNIVERSITY OF SOUTH CAROLINA

- The reconstruction error is shown below where N is the batch size, sˆ is the output of the decoder, and θ are the parameters of the autoencoder and model.

$$L_r(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}||s_i - \hat{s}_i||_2^2 + \frac{1}{2}||s'_i - \hat{s'_i}||_2^2$$

- We only round the output of the model when encouraging the output of the encoder to match the output of the model and the output of the encoder is always rounded. This is shown below, where r() is the rounding function that uses a straight-through estimator during gradient descent and .detach() removes the tensor from the computation graph.

$$L_m(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{2}\|r(\tilde{s'_i}) - r(\hat{\tilde{s}}'_i).detach()\|_2^2 + \frac{1}{2}\|r(\tilde{s'_i}).detach() - \hat{\tilde{s}}'_i\|_2^2)$$

- We use a weight ω to first weight the L_r loss higher than L_m and gradually adjust ω to be 0.5 to weight them equally.

$$L(\theta) = (1-\omega)L_r(\theta) + \omega L_m(\theta)$$

- A target network with parameters φ− is maintained and periodically updated to be φ during training. The loss function used is:

$$L(\phi) = (G(s,a,s') + \min_{a'} q_{\phi^-}(s',a',s_g) - q_\phi(s,a,s_g))^2$$