Research Poster:

Learning Discrete World Models for Heuristic Search

Forest Agostinelli and Misagh Soltani

Computer Science and Engineering, University of South Carolina

AI Institute of University of South Carolina

UNIVERSITY OF South Carolina

AI INSTITUTE #AIISC UNIVERSITY OF SOUTH CAROLINA

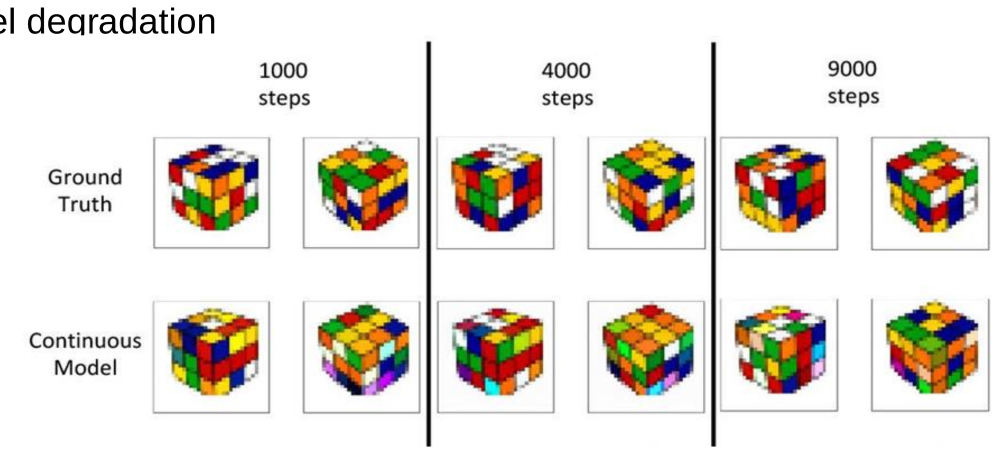
Scan to access the **paper**

Scan for the **GitHub repository**

Scan to connect on **LinkedIn**

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, such as robotics, model-based reinforcement learning can be used to learn it.
- Continuous world models face two major challenges:
  - Lack of state re-identification
  - Model degradation



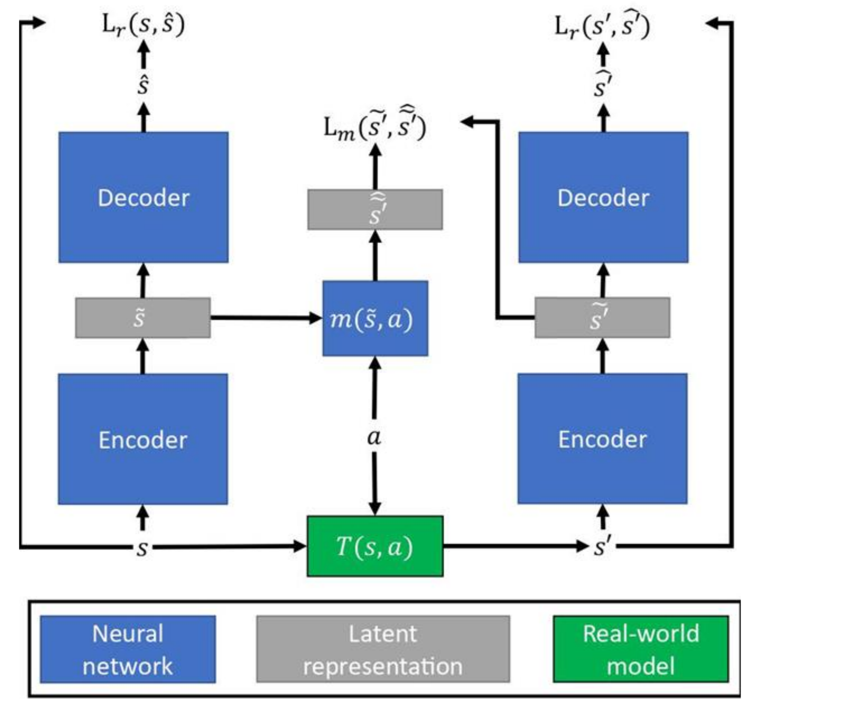
Our Approach - DeepCubeAI

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:
  - **Discrete world model**
    - Learns a world model that represents states in a discrete latent space.
      - ❑ Errors less than 0.5 in prediction can be corrected by simply rounding
      - ❑ Can re-identify states by comparing two binary vectors
  - **Heuristic function**
    - Uses RL to learn a heuristic function that generalizes over start and goal states
  - **Search**
    - Combines learned model and learned heuristic function with heuristic search to solve problems.

Learning Discrete World Models

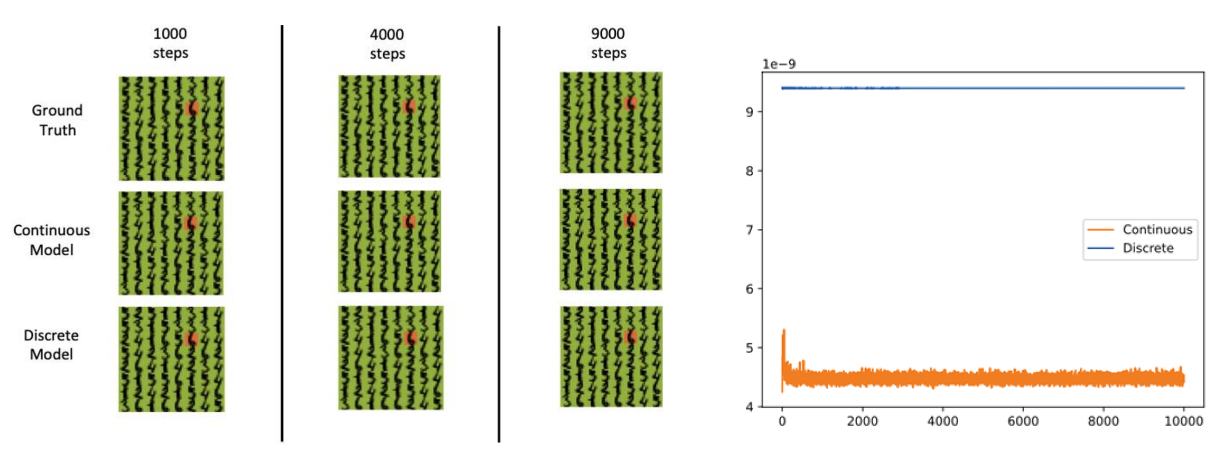
- **Encoder**
  - Maps the state to a discrete representation by rounding the output of the encoder.
  - Uses a straight-through estimator to allow training with gradient descent.
- **Decoder**
  - Maps the discrete representation to the state.
  - Ensures the discrete representation is meaningful.
- **Environment model**
  - Maps discrete states and actions to next discrete state.
  - We train the autoencoder and model together to ensure that the parameters of the autoencoder are encouraged to learn a representation that the model can also learn.
    - 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: $L(\theta) = (1-\omega)L_r(\theta) + \omega L_m(\theta)$
- Another benefit of the discrete world model
  - We can see the percentage of the bits that match.
    - If the latent representation is meaningful and we can predict it accurately, then if the Markov assumption holds, we can roll it out for as many steps as we want.



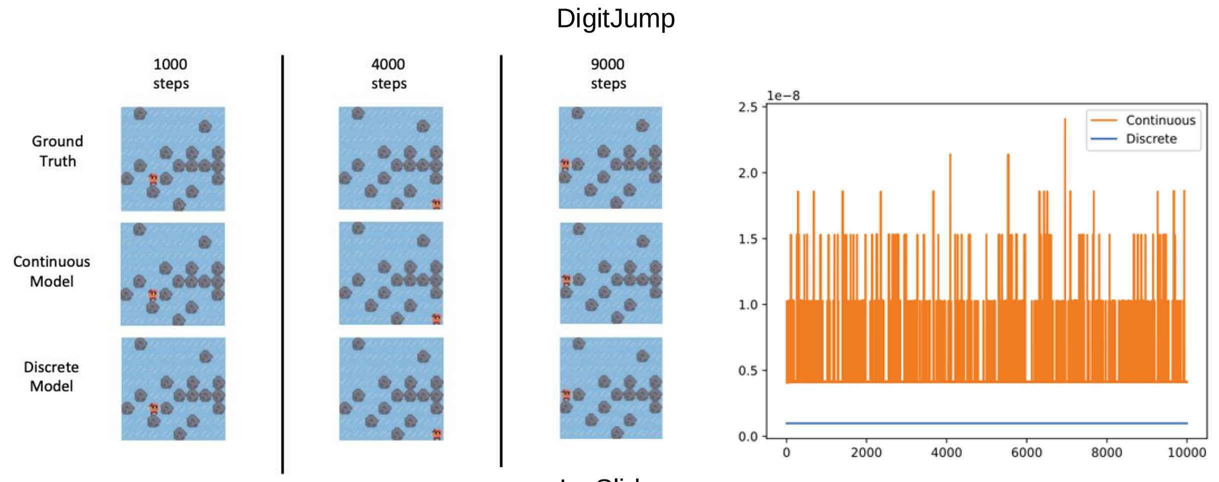
$$L_m(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}||r(\tilde{s}_i') - r(\hat{\tilde{s}}_i').detach()||_2^2 + \frac{1}{2}||r(\tilde{s}_i').detach() - \hat{\tilde{s}}_i'||_2^2$$

Discrete vs Continuous Model Performance

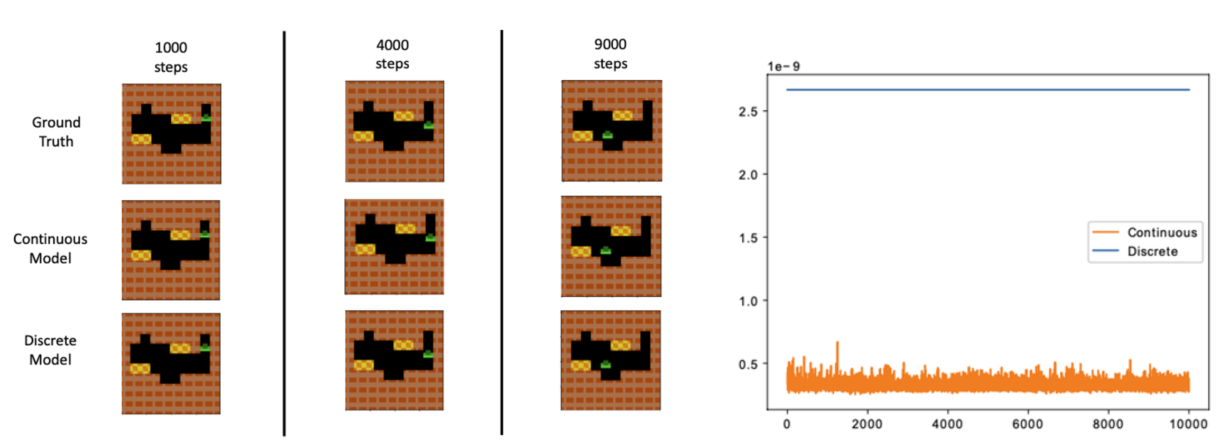
- **Left:** A visualization of the reconstructions for models with continuous and discrete latent states at different timesteps.
- **Right:** Mean squared reconstruction error as a function of timestep.



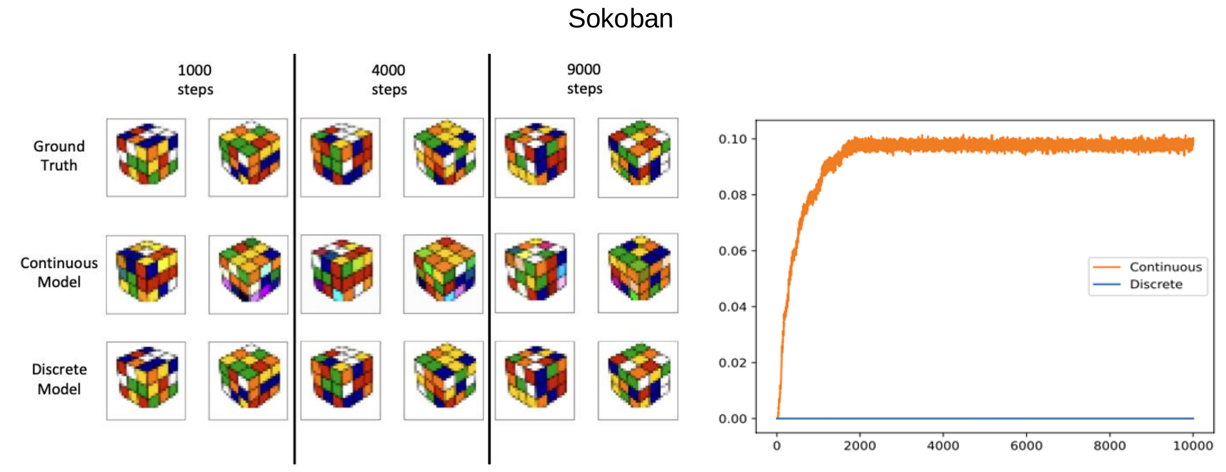
DigitJump



IceSlider



Sokoban



Rubik's Cube

Heuristic Learning and Search with Discrete Model

- We use offline data and the learned world model to generate training data.
- **Heuristic learning:** we use 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 Deep Q-Networks.
  - 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.

Problem Solving Performance

- Pattern Databases (PDBs) use human knowledge from group theory. DeepCubeA uses a predefined goal during training and requires retraining for each Rubik's Cube Reverse problem instance.
- Poor performance when following heuristic values greedily highlights the necessity of planning.

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%**

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.[2]
- Improve goal specification in environments where goal images are difficult to generate, potentially using formal logic to specify goals without generating goal images.[3]
- Extend benefits of discrete models to stochastic and partially observable robotic tasks, enhancing exploration for training and obtaining more lookahead during search.[4]

Conclusion

- We introduce **DeepCubeAI**, a domain-independent method for learning a model that operates on discrete latent states.
- We address the challenges of **model degradation** and **lack of state re-identification**.
- The learned model is used to learn a **heuristic function that generalizes across problem instances**.
- We combine the learned model and the heuristic function with **search** to solve problems.
- For the Rubik's cube, using a discrete model **prevents error accumulation**.
- DeepCubeAI solves **100% of test cases** for Rubik's cube, Sokoban, IceSlider, and DigitJump, and **99.9% of test** cases for Rubik's cube reverse, demonstrating effective generalization across goal states.

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