Latent Action Space for Efficient Planning in Theorem Proving

Minchao Wu*, Yuhuai Wu*

August 24, 2021

Abstract

One of the most critical challenges in applying machine learning techniques to automated reasoning is the need to work with an enormous action space. Not only does it make the exploration difficult, but also it is very time-consuming to generate at inference time. In this work, we introduce latent action space with a world model to speed up the efficiency of action generation, with the potential of alleviating the exploration problem, as well as improving sample efficiency using the world model.

1 Introduction

One of the major challenges in theorem proving [14, 2, 7, 12, 10, 11, 6] is the need to deal with an enormous action space. In the most general setting, the action space for a theorem proving agent consists of a sequence of strings, representing a tactic application along with theorem parameters, or an intermediate proposition, or a new definition, lemma, theorem statements. This approach also has been adopted in recent works using transformer-based models [12, 6], because of its generality (e.g., the capability of generating new terms). However, due to the nature of autoregressive generation for such actions, even doing one action generation requires a quiet amount of time, not to mention if one wants to search multiple steps ahead.

In this work, we propose to learn such a high-level representation, by embedding the raw action space into a latent action space.

2 Method

In order to embed the action into a continuous latent space, we first introduce an action encoder and an action decoder.

- Encoder_{action}: action space $\to$ latent action space : $\alpha \sim q(\alpha|a)$.
- Decoder_{action}: latent action space $\to$ action space : $\hat{a} \sim p(\hat{a}|\alpha)$.

We can train the action encoder and decoder using the reconstruction objective:

$$\mathcal{L}_{rec} = D(p(a)||p(\hat{a}|a)), \tag{1}$$

where $p(\hat{a}) = \sum_\alpha p(\hat{a}|\alpha)q(\alpha|a)$ and D is a metric on the distribution of actions. The most natural choice is the KL divergence (cross entropy loss) between the original action distribution (label) and the decoded action distribution.

*Equal contribution. MW is at the Australian National University and YW is at the University of Toronto. Correspondence to `Minchao.Wu@anu.edu.au` and `ywu@cs.toronto.edu`.

However, we cannot directly work with latent action space, because the environment only accepts the raw action space as input. Therefore, an "environment" in the latent space is necessary for this purpose. This then naturally leads us into the model-based RL techniques [13, 3, 4, 8, 5].

We hence introduce the following components. Firstly, we introduce a state encoder, that encodes the proof state x_t into a latent state space $\mathcal{Z}$ represented by a continuous vector, as well as its counterpart, a state decoder – that decodes the latent state back to the proof state.

- State encoder: $z_t \sim q(z_t|z_{t-1}, \alpha_t, x_t)$
- State decoder: $\hat{x}_t \sim p(\hat{x}_t|z_t)$

Next, given the latent state space, we introduce the latent transition operator. The transition operator will sample the next latent state given the latent state and action at the current time step. Namely, we use a neural network model to learn the internal dynamics of the theorem proving engine, performing the deduction step of theorem proving.

- Latent transition operator: $\hat{z}_t \sim p(\hat{z}_t|z_{t-1}, \alpha_{t-1})$

To train the state encoder and decoder, we also use a reconstruction objective as the case of latent actions. To train the transition operator, we use a forward prediction loss – the cross-entropy loss between the ground truth latent state and the predicted latent state. Furthermore, to add more semantic groundings for the latent action space, we also use the forward prediction loss to train the action and state encoder. To summarize, the total loss objective is written below:

$$\mathcal{L} = \mathcal{L}_{rec}(a) + \mathcal{L}_{rec}(x) + \mathcal{L}_{forward}.$$

Given a latent transition operator, one can perform efficient planning in the latent space – looking ahead by unrolling the state dynamics for a number of steps. Unlike generating a full sequence of tokens at each step, the latent action allows a one-shot generation, immensely shortening the planning time. There are many possibilities in terms of integrating the transition operator with various kinds of search algorithms, such as best-first search, MCTS, etc. – a vast space for exploration.

3 Experiments

We start by learning the state and action encoding, as well as the dynamics of the INequality Theorem proving benchmark (INT) [15]. We generate 40000 proof trajectories using INT using a cardinality of an axiom combination $K = 3$ and a length of a proof $L = 7$. The data set then contains 149009 distinct transitions which are split into training and test data sets with a 80:20 ratio.

We use a character-level transformer to learn the latent representations of both state and action, and use an MLP to learn the internal dynamics (*i.e.* the transition operator) of INT. The transformer uses 256 embedding dimensions,

Table 1: Performance on the test set. $\texttt{BLEU}_{rec-action}$ (resp. $\texttt{BLEU}_{rec-state}$) denotes the BLEU score of the reconstructed actions. $\texttt{BLEU}_{trans}$ denotes the BLEU score of the predicted states by applying the transition operator once. QED accuracy is the percentage of correctly predicted QEDs by applying the transition operator once.

Methods	$\texttt{BLEU}_{rec-action}$	$\texttt{BLEU}_{rec-state}$	$\texttt{BLUE}_{trans}$	QED accuracy (%)
$\mathcal{L}_{CE}$	96.98	94.12	88.23	94.20
$\mathcal{L}_{MSE}$	73.87	69.38	60.18	0

8 attention heads and 1024 hidden dimensions for position-wise feed-forward layers. We also use dropout with rate 0.1, label smoothing with coefficient 0.1, and a maximum 128 tokens for both training and evaluation examples. The MLP is a residual block with two hidden layers of dimensions 1024 and 512. We use the Adam optimizer [9] for training.

We experiment with two different forward losses for training the transition operator. $\mathcal{L}_{CE}$ denotes the cross-entropy loss between the ground truth target state and the decoded predicted latent state. $\mathcal{L}_{MSE}$ denotes the mean squared error between the encoded ground truth of target state and the predicted latent state. We implement our algorithms in JAX [1] and run both experiments for 100k training steps using a single NVIDIA Tesla V100 GPU and 8 cores of an Intel(R) Xeon(R) CPU @ 2.20GHz.

Table 1 shows the quality of the transition prediction and the reconstruction of states and actions when evaluated on the test set. When calculating the BLEU scores for transition predictions, we separate out those whose references are a single "QED" token (which indicates the end of the proof) to make sure that BLEU scores reflect the quality of prediction properly[1]. We add an additional metric called QED accuracy which is the percentage of exact matches of the QED token. Figure 1 shows the quality of transition prediction when the state dynamics is unrolled for a number of steps using the learned transition operator. It can be seen that the transition operator trained using $\mathcal{L}_{CE}$ outperforms the one trained using $\mathcal{L}_{MSE}$ by a large margin, and that the latter lacks the ability to correctly predict QEDs.

4 Discussion

There has been an early investigation on latent space for mathematical reasoning [10], which shows promising results of neural networks for predicting the latent state several steps ahead, in the HOList system with an ad-hoc action

[1]For example, if we have references: ["QED","QED","QED","QED"] with predictions: ["*to* $((((b*a)+(a*b))*(a*(a+b)))*(c*1)) = ((((a+b)*a)*(a+b))*(c*1))$","QED","QED","QED"]), the BLEU score of this corpus is only 0.79, which does not reflect the quality of prediction properly.

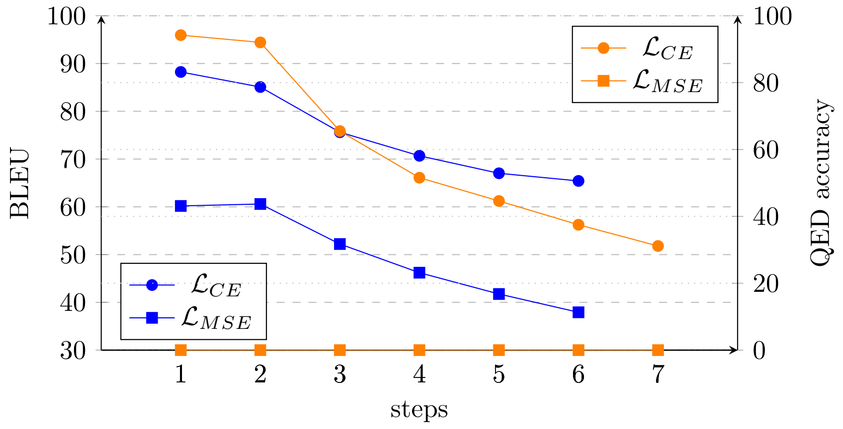

Figure 1: Quality of transition operator with respect to the number of steps unrolled. Given a state s, we look ahead n steps by recursively applying the transition operator to s and the subsequent ground truth actions corresponding to s. The further we unroll, the more difficult it becomes for the transition operator to correctly predict the target states. Note the different scale on right for QED accuracy. Step 7 has a QED accuracy instead of a BLEU score because all target states at step 7 are QEDs.

space. Greatly inspired by it, we propose to build a full-fledged latent space system for the most general action space, to improve mathematical reasoning in planning efficiency. In the meantime, the world model potentially can speed up the interaction time with the environment, and also improve the sample efficiency. Furthermore, we believe if the latent space is semantically grounded, exploration in the latent action space can also provide big gains over exploring with a sequence of long tokens. We hope our work provides a meaningful direction to future machine learning models for theorem proving.

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[2] James P. Bridge, Sean B. Holden, and Lawrence C. Paulson. Machine learning for first-order theorem proving - learning to select a good heuristic. *J. Autom. Reason.*, 53(2):141–172, 2014.

[3] David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

[4] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[5] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *CoRR*, abs/2010.02193, 2020.

[6] Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. Proof artifact co-training for theorem proving with language models. *CoRR*, abs/2102.06203, 2021.

[7] Geoffrey Irving, Christian Szegedy, Alexander A. Alemi, Niklas Eén, François Chollet, and Josef Urban. Deepmath - deep sequence models for premise selection. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2235–2243, 2016.

[8] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[10] Dennis Lee, Christian Szegedy, Markus N. Rabe, Sarah M. Loos, and Kshitij Bansal. Mathematical reasoning in latent space. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[11] Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence Paulson. Isarstep: a benchmark for high-level mathematical reasoning. 2021.

[12] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *CoRR*, abs/2009.03393, 2020.

[13] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Bruce W. Porter and Raymond J. Mooney, editors, *Machine Learning, Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, USA, June 21-23, 1990*, pages 216–224. Morgan Kaufmann, 1990.

[14] Josef Urban, Jirí Vyskocil, and Petr Stepánek. Malecop machine learning connection prover. In Kai Brünnler and George Metcalfe, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings*, volume 6793 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2011.

[15] Yuhuai Wu, Albert Jiang, Jimmy Ba, and Roger B. Grosse. INT: an inequality benchmark for evaluating generalization in theorem proving. *CoRR*, abs/2007.02924, 2020.