Cracking PHYRE with a World Model

Sheng Liu[1]

University at Buffalo, SUNY

Abstract. The ability to reason about physics is crucial for intelligent agents to interact with the environment. However, not much progress has been made in endowing machines with such an ability [1]. We presume that the lack of progress is due to the lack of an explicit world model. Inspired by the superior performance world model achieved on various benchmarks in OpenAI Gym [2], we propose to explicitly model the physical world with a novel world model, which is composed of a Perceptor (P) and an Imaginator (I). The world model is capable of perceiving environmental changes and predicting plausible evolution of the environment based on its perceived information. To validate the effectiveness of the proposed world model, we conduct experiments on PHYRE, *i.e.*, a benchmark for physical reasoning, the results show that the world model is able to help an agent reason about physics.

Keywords: World model, Physical reasoning, Actor critic

1 Introduction

Humans, even children who have not taken a single physics course, are able to reason about physics. However, intelligent agents struggle with physics, including those equipped with state-of-the-art reinforcement learning (RL) algorithms, *e.g.*, REINFORCE [5], DQN [8], A3C [7], PPO [9]. The goal of this project is to design a novel world model which is able to endow an agent with the ability to master laws of physics and leverage the mastered laws to solve challenging tasks in *unseen* environments.

The world model we propose is composed of two components, *i.e.*, a Perceptor and an Imaginator. The Perceptor is responsible for processing the visual information the agent observes and the Imaginator is responsible for hallucinating future observations of the environments based on the information that the Perceptor provides. In order to generate high quality predictions, the Imaginator has to understand Newton's law of motion as objects in environments defined in PHYRE move following Newton's law of motion. With the help of world model, we will train a rather simple agent using actor critic algorithm.

2 Related Work

Bakhtin *et al.* [1] proposed the task of PHYRE. As PHYRE is a rather new topic, [1] is the only related work and the only baseline with which we could compare. Specifically, Bakhtin *et al.* designed a novel benchmark composed of 50 classical physical puzzles. An agent has to master laws of physics in order to solve the puzzles.

Ha *et al.* [3, 2] proposed the concept of world model and tested it in environments defined in OpenAI Gym. It improves sample efficiency of state-of-the-art RL algorithms [10].

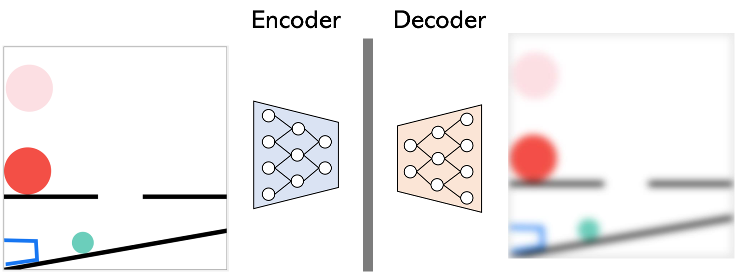

Fig. 1. An illustration of the autoencoder which is used to train the Perceptor. The autoencoder is composed of an encoder *i.e.*, the Perceptor and a decoder, both of which are convolutional neural networks (CNNs). The encoder encodes its input, *i.e.*, an image, into a feature vector, and the decoder decodes the feature vector into an image. The objective of the autoencoder is to reconstruct its input from the feature vector, thus allowing us to train it in an unsupervised manner.

3 Methodology

Our goal is to design a novel world model which is able to reason about physics (at least Newtonian mechanics as the motion of objects in environments defined in PHYRE follows Newtonian mechanics). Ideally, an agent equipped with the world model should be able to solve the 2D physical puzzles from PHYRE in a sample efficient way.

3.1 Rules of PHYRE

PHYRE contains a set of physical puzzles. The agent is allowed to add a ball (a red ball) to the environment. The goal is to control the position and size of the red ball the agent adds to the environment to ensure that the blue object and the green object in the environment contact each other when a trial ends, *i.e.*, all the objects stop moving. The objects moves following Newton's law of motion. The objects in black is not movable.

3.2 Notations and Symbols

- observation o_t: observation at timestep t (o_t) is composed of $n+1$ images whose resolution is 256×256, *i.e.*, $o_t = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, \mathbf{x} \in \mathbb{R}^{256\times256}$,

$\forall \mathbf{x} \in {o_t}^1$. $\mathbf{x}_0$ is the initial scene that specifies the layout of a physical puzzle. Hence, it is the same for all trials.

- state s_t: state of the environment at timestep t is composed of observations of all trials.
- action a_t: action at timestep t ($a_t \in \mathbb{R}^3$) specifies size and initial position *i.e.*, x, y coordinates, of the red ball which the agent adds.
- reward r_t: reward at timestep t is 1 if the trial at timestep t solves the puzzle otherwise it is $-\alpha$. α is a small positive number which encourages the agent to solve the puzzle with less number of trials.

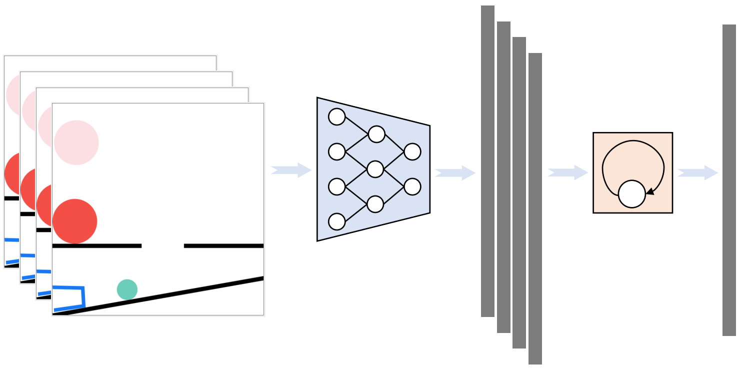

Fig. 2. An illustration of how the Perceptor, *i.e.*, the CNN shown in blue, and the Imaginator, *i.e.*, the LSTM shown in orange, work. Given observation of a trial (a set of images), the Perceptor first encodes each image into a feature vector, which is then fed into the Imaginator. The task of the Imaginator is to predict the *next* image. In order to make high quality prediction, the Imaginator has to reason about the position of the movable objects, *e.g,* the red ball, the green ball, the blue cup, whose motion follows Newton's law. Hence, the Imaginator has to be able to master Newton's law.

3.3 Perceptor

The first component of the world model is the Perceptor. As shown in Figure 1, the Perceptor is implemented as a CNN. It takes an image as input and encodes it into a feature vector. It is trained in an unsupervised manner with the help of an autoencoder.

3.4 Imaginator

The other component of the world model is the Imaginator. As shown in Figure 2, the Imaginator is implemented as a LSTM [4]. At each time step, it takes a

[1] subscripts are abbreviated for simplicity

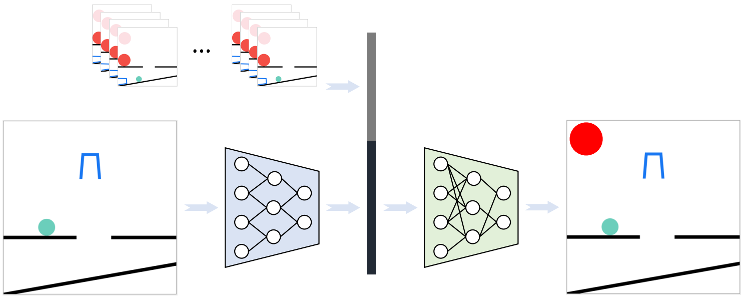

Fig. 3. An illustration of the agent. Given an initial scene of a puzzle, together with observations obtained from past trials, the Perceptor encodes the initial scene into a feature vector (the dark gray bar), and an LSTM encodes past observations into another vector (the gray bar). The two vectors are concatenated and fed into a MLP for action prediction.

feature vector as input; it updates its hidden state and predicts the *next* image.. It is trained in an unsupervised manner with the help of an autoencoder. In order to make high quality prediction, the Imaginator has to reason about the position of the movable objects, *e.g,* the red ball, the green ball, the blue cup, whose motion follows Newton's law. Hence, the Imaginator has to be able to master Newton's law. Note that as the *next* image is also part of the observation, the Imaginator is also trained in an unsupervised manner.

3.5 Agent

With the help of the world model, the agent is rather simple. It is implemented as a two-layer perceptron (shown in Figure 3).

4 Experiments

4.1 Implementation Details

Perceptor:
The Perceptor is implemented as a six-layer CNN. The numbers of output channels of the six convolutional layers are $32, 64, 64, 64, 64, 64$. The kernel size of convolutional layer is 3. Three pooling layers are applied after the second, the forth and the sixth convolutional layers. The decoder of the autoencoder used to train the Perceptor is implemented as a six-layer CNN as well. All of its six layers are deconvolutional layers with 64 output channels. Their kernel size is 3.

Imaginator:
The Imaginator is implemented as a one-layer LSTM. It's hidden state dimension is set to 128.

Agent:
The agent is implemented as a two-layer perceptron. The number of output channels of the first layer is set to 128. That of the second layer is set to 3 (as action $a \in \mathbb{R}^3$).

Training:
We train our agent using actor critic algorithm (A2C) [7]. The decay factor γ is set to 0.95. We use ADAM optimizer [6] to optimize parameters of all the components of our model. The learning rate is set to $1e-4$, weight decay is not used.

4.2 Comparison with Baselines

We compare our method with two simple baselines on PHYRE in Table 1. AUCESS and SP@10 are two evaluation metrics defined in [1]. The larger the two metrics are, the better the algorithm is. Please refer to [1] for details regarding the two baselines and the two evaluation metrics. As can be seen, our method outperforms the two baselines, especially MEM. The reason that our method is not able to significantly outperforms RAND is that the two LSTMs used in the Imaginator and the agent (for encoding past observations) is hard to train in an RL setting.

Method	AUCCESS	SP@10
RAND [1]	13.7	7.7
MEM [1]	2.4	2.7
Ours	16.8	9.4

Table 1. Comparison with baselines on PHYRE benchmark. RAND and MEM are two baseline methods proposed in [1]. AUCESS and SP@10 are two evaluation metrics defined in [1]. The larger the two metrics are, the better the algorithm is. Please refer to [1] for details regarding the two baselines and the two evaluation metrics.

References

1. Bakhtin, A., van der Maaten, L., Johnson, J., Gustafson, L., Girshick, R.: Phyre: A new benchmark for physical reasoning. In: Advances in Neural Information Processing Systems. pp. 5083–5094 (2019)
2. Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution. In: Advances in Neural Information Processing Systems. pp. 2450–2462 (2018)
3. Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
5. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of artificial intelligence research **4**, 237–285 (1996)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. pp. 1928–1937 (2016)
8. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
9. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
10. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)