World Models

January 29, 2023

Giorgio Strano

Abstract

In Reinforcement Learning, we can model the entire environment with algorithms that compress the observations and predict the future. Can this technique be used to make the procedure of learning a policy much simpler and faster compared to traditional methods? In this project, we implement the ideas presented in (Ha & Schmidhuber, 2018) and replicate the original results. Then, we extend the algorithm to work in different games and analyze the interesting challenges, difficulties and potential of model-based RL.

1. Introduction

At a high level, we can describe a *model* of any environment as a combination of a representation of the current state and a transition function that describes how the environment evolves over time. We can use a convolutional VAE to learn a representation in *latent space* of the observation and a MDN-RNN model (called *memory*) to learn how the environment transitions from one latent state to the next. This model consists of an LSTM in which the hidden state is fed into a MDN layer. Lastly, the latent encoding of the current observation and the hidden state of the memory are passed to a controller, which is a simple fully-connected layer in charge of finding the best action at any point in the game. The goal is that this model of the environment could help the agent to learn better policies, faster. We will study this method, test this claim, and investigate on the extensibility of this approach to very different environments.

2. Training method

The three models are all trained separately on different types of input data. The procedure starts by generating a dataset of around 10,000 replays of games played with random moves, and training the VAE on the images of the frames. Then, the memory is trained using as inputs the concatenation of `<latent state + action taken>` and has the task to generate a probability distribution for the latent representation of the next frame. The distribution is represented by the MDN layer as a mixture of 5 gaussians. Differently from the previous models, which are trained through traditional backpropagation, the controller, given its very low number of parameters, is trained through evolution with a genetic algorithm. While the original paper tries out several genetic algorithms, the huge amount of time needed for training on our hardware forced us to stick with one algorithm, namely the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen, 2016).

3. CarRacing environment

CarRacing is an environment included in the OpenAI Gym (Brockman et al., 2016) toolkit, in which the agent drives a car along a racetrack with a 2D top-down view. The objective is to drive as fast as possible while keeping the car from going off track. In this environment the frames are very simple, so the VAE has no trouble reconstructing a very accurate image, as shown in figure 1. The main bottleneck in the learning process turned out to be the controller: with our single-GPU setup, the CMA-ES algorithm could run only 3 processes in parallel, causing the training procedures to last up to two full days or more, which made hyperparameter tuning very impractical. Furthermore, adding a hidden layer to the classifier, as the authors of the paper did, increases the parameter count from 864 to 18,691 (with 64 hidden nodes), making the optimization process almost uncomputable on a personal machine.

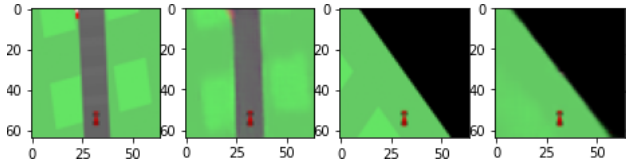

Figure 1. Two pairs of frames from the game and their VAE reconstruction.

Email: Giorgio Strano <strano.1809528@studenti.uniroma1.it>.

Deep Learning and Applied AI 2022, Sapienza University of Rome, 2nd semester a.y. 2021/2022.

4. Different Atari games

Many interesting challenges came up when we tried this algorithm on different Atari games from the Procgen (Cobbe et al., 2019) suite. The first game we tried was **Chaser** (a redesign of Pacman), but training the VAE resulted in a very strong case of **posterior collapse** (Dai et al., 2020), (Lucas et al., 2019). Since the images in the dataset were all extremely similar, the VAE was able to learn a very low-effort policy by always reconstructing the same image, a sort of "average" of the dataset. After trying out many different strategies that did not improve the VAE (cfr. this project for reference), we solved the problem in a pretty original (as far as we know) way: we preprocessed the training dataset replacing the areas in common to every image with random noise, which forced the VAE to also learn how to reconstruct the details (figure 2).

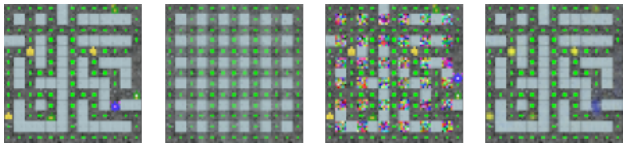

Figure 2. From left to right: original image, reconstruction affected by posterior collapse, example of randomized image, reconstruction by the VAE trained on randomized images.

After solving this problem, however, another issue came up: in this game, most of the crucial details (i.e: the food particles) are drawn as squares as little as 2 pixels wide, which the VAE was not able to reliably represent and reconstruct across the whole dataset. Even after training to convergence a huge variety of different architectures (see this repository), all of our experiments resulted in very slightly blurry reconstructions, which cannot be accepted in a context where pixel-level details matter fundamentally.

The same problem persisted when we tried a different game from Procgen, **Dodgeball**. We come to the conclusion that, without some much more extended reasearch and architecture tuning, VAEs are generally not well suited to encode and reconstruct images with pixel-level precision.

Lastly, we experimented with **Enduro**, an Atari driving game. Here, the VAE had no trouble reconstructing the images properly, though once again the controller seemed to be a bottleneck. After a thorough manual examination of the agent's playstyle, we confirmed that it had definitely learned to drive as fast as possible and how to stay on track, and it had learned the *idea* of moving to dodge incoming cars. However, most of the times, the movements were not performed fast enough and it ended up bumping other cars and slowing down significantly.

5. Results

The results obtained were very interesting: in the **CarRacing** setting the agent learned to drive close to perfection as long as it stayed on track, but when it happened to drift off until the track was no longer on the screen, it often got stuck driving in circles, bringing the average score down to ~ 500. In the better runs, the agent repeatedly beat the game, with a score of 901, which is almost on par with the state of the art results from the paper (906±21). For comparison, A3C (Mnih et al., 2016) does not even reach a score of 700 (Ha & Schmidhuber, 2018).

In **Enduro** the results were not strong enough to consider the game beaten, with an average score of 44.2 and a maximum of 91.0, also due to the very sparse reward policy of the game: the agent only gets a point when it overtakes a car, which ignores very important "human" evaluation strategies, such as the speed of the car. For reference, we implemented the widely used REINFORCE algorithm, which scored an average very close to 0.

Given that in both cases the agent easily picked up the mechanics of the game, we believe that the world model is working. Much better results could be achieved in Enduro, and more consistent ones in CarRacing with a bigger controller, more iterations of training and proper hyperparameter tuning (including a selection of the best optimization algorithm), which would take too much time on any consumer-grade GPU.

6. Discussion and conclusions

We come to the conclusion that the idea of World Models has great potential: we were able to achieve close to state of the art performance on CarRacing even with low spec hardware, and challenge other popular algorithms in Enduro. The downsides, however, are multiple as well. As we showed with Chaser and Dodgeball, being reliant on a single VAE for the world representation means to give up pixel-level information, which may be crucial to the game. The procedure of gathering samples to train the VAE and memory through random play has an intrinsic flaw: it makes it impossible to properly model a world that evolves when the player acts in the correct way. We noticed this while playing Dodgeball, as every level contains a teleport to the next level, which is only activated after defeating all the enemies. The memory has no chance to learn how to model this behavior using random examples. Lastly, the controller offers very little scalability: the parameter count of DL models increases very fast with the complexity of the problems tackled, and the evolutionary algorithms do not keep up. It would be a very interesting research to apply more modern algorithms, such as A3C (Mnih et al., 2016) or DDQN (van Hasselt et al., 2015) to train the controller.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning, 2019. URL https://arxiv.org/abs/1912.01588.

Dai, B., Wang, Z., and Wipf, D. The usual suspects? Reassessing blame for VAE posterior collapse. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2313–2322. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/dai20c.html.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pp. 2451–2463. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution. https://worldmodels.github.io.

Hansen, N. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016. URL http://arxiv.org/abs/1604.00772.

Lucas, J., Tucker, G., Grosse, R. B., and Norouzi, M. Understanding posterior collapse in generative latent variable models. In *DGS@ICLR*, 2019.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL http://arxiv.org/abs/1509.06461.