Visual Doom AI Competition @ CIG 2016: 5vision team solution

Ivan Sorokin, Anastasiia Ignateva, Alexander Orlov, Mikhail Pavlov
5visionteam@gmail.com

September 10, 2016

1 Introduction

Our solution is based on similar principles that are used in AlphaGo from DeepMind Team [1]. In particular, the agent observes the current screen of the game and choose the action using a policy network. This deep neural network are trained by a combination of supervised learning (SL) from human games, and reinforcement learning (RL) from games of self-play.

The training procedure consists of two stage. For the first stage, we played a few hours of the game using a spectator mode in ViZDoom platform. This allowed us to collect the necessary data (pairs of screenshots and actions) for supervised learning in which the network predicts human actions. For the second stage, we continued to train network by reinforcement learning using only data from games of self-play. Moreover, we used a natural gradient descent (NGD), an optimization algorithm which is difficult to apply for high-dimensional problems. Therfore, we would like to quote the work [2], which gives insight on this subject area:

> As an analogy one can think of such powerful 2nd-order optimizers as extremely fast racing cars that need more sophisticated control systems than standard cars to prevent them from flying off the road.

2 Network Architecture

We build the deep policy network based on our previous work [3], where we presented Deep Attention Recurrent Q-Network (DARQN). The new architecture is schematically shown in Figure 1 and likewise consists of three types of networks: convolutional, attention and recurrent. Major changes are made in the attention mechanism. Now it takes four inputs: one of the feature vectors v_t^i of the last CNN layer, previous hidden state h_{t-1}^1 of the first GRU layer, previous attention map g_{t-1} and previous action a_{t-1} that has been chosen by the agent. As can be seen, we do not use any other variables of the game, s_t is a raw image of the current game screen.

We experimented only with the "soft" attention mechanism, because it's easy to train. In our case, the attention map is a softmax outputs with 24 values, which corresponds to 4×6 feature map of the last CNN layer. Other details of our architecture can be found in the source code, which we plan to publish in our github account[1].

It should also be noted that the attention mechanism greatly reduces the size of the model. For example, DQN with 18 possible actions have $1,693,362$ adjustable parameters [4], whereas the suggested model with 7 actions have only $108,432$ parameters. We tried to make the model as small as possible, because we used NGD in simple and crude way.

[1] https://github.com/5vision/

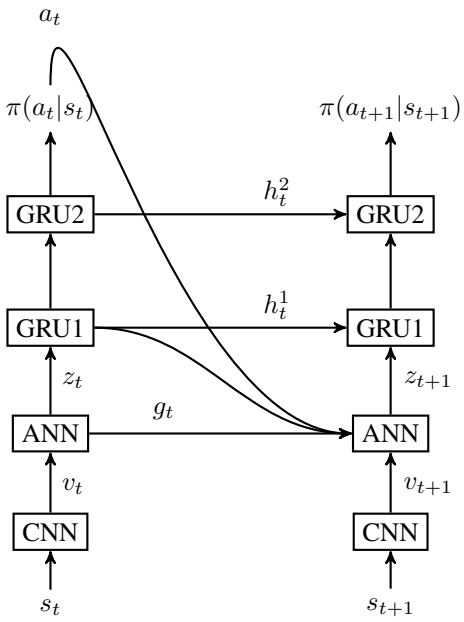

Figure 1: The Deep Attention Recurrent Policy Network

3 Training procedure

In the SL stage, we minimize the categorical cross-entropy between predictions (softmax outputs) and targets (human actions). In order to optimize this objective function, we applied the Adam updates for a mini-batch size of 16 sequences, each of which is 50 steps long.

For the Track 1, we had 9.87 hours of human play on two maps: map01 and horizontally mirrored map01. The pre-training lasted about 8 hours, which corresponds to about $365,000$ updates.

For the Track 2, we had 18.27 hours of human play on four maps: map01, map02 and corresponding horizontally mirrored version. The pre-training lasted about 24 hours, which corresponds to about 1M updates.

In the RL stage, we decided to use a batch mode with the size of 128 sequences. In particular, we trained the policy network in the similar way as done in the work [5], but except for a few things. First of all, we split all parameters of the model into two parts ($\approx$ 50k CNN parameters and $\approx$ 58k others), and estimated the Fisher Information Matrix (FIM) independently for each part. Moreover, we used unlabeled data for estimation [6]. In other words, at each training step we compute the gradient and the metric on the different samples. Secondly, in order to solve a linear system, we replaced the Conjugate Gradient algorithm by MinResQLP [7]. We also applied the constant momentum and the weights averaging [2].

In both stages, the agent is playing againts the bots from Doom engine. Nevertheless, we have noticed that the bots on the map01 behave poorly. Therefore, for the Track 1 we have decided to additionaly train the model against itself. The RL stage with Doom's bots took about 3000 updates and 46 hours. The stage with self-play games took about 600 updates and 58 hours. In the case of Track 2, we have not noticed a significant improvement in RL stage. The average reward is growing, but the behavior is almost unchanged. Most likely due to the complex reward function and low capacity of the model.

4 Doom specific knowledge

ViZDoom platform provides a rich set of possible actions. Unfortunately, we have not experimented with this enough, and simply choose the easiest set of actions: ATTACK, TURN_LEFT, TURN_RIGHT, MOVE_FORWARD. It is worth noting that for the Track 1 we defined only 7 possible actions, i.e. when the agent is attacking, he can not move. Maybe it is main mistake, but on the other hand it simplifies the estimation of FIM. Respectively, for the Track 2 we used 12 actions.

As we mentioned, we do not use the variables of the game to make decisions, but we use them to specify a reward function. We have also added a little trick, which selects the best weapon in the Track 2.

5 Conclusion

A straightforward way to compete in this challenge is just to apply SL. But our desire was to develop an RL algorithm that would improve the model which simply mimics a human. For a simple environment, such as Track 1, we have been able to succeed, but the agent is still reactive in his behavior, it does not have a long planning strategy. For example, to run for a medical kit only when a health is low or to freely navigate in a maze with enemies.

The use of NGD imposes the limit to our model sizes. Perhaps this is the main drawback of our approach. Fortunately, there are better ways to incorparate 2nd-order information (K-FAC, KFC, PRONG, etc.) and in this direction a lot of things can be done.

We also developed the new architecture that much better than our previous DARQN model. Unfortunately, we have not shown a comparison of these two models in this report, and we also have not proven advantages of RL stage. In case of successful participation in the competition, we will publish more evidence and a detailed description of our solution.

References

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[2] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. *Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: WCP volume 37., arXiv:1503.05671*, 2015.

[3] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network. *Deep Reinforcement Learning Workshop, NIPS 2015, arXiv:1512.01693*, 2015.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] John Schulman, Sergey Levine, Philipp Moritz, Michael I Jordan, and Pieter Abbeel. Trust region policy optimization. *Proceedings of the 31st International Conference on Machine Learning, Lille, France, 2015. JMLR: WCP volume 37, arXiv:1502.05477*, 2015.

[6] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv:1301.3584*, 2013.

[7] Sou-Cheng T Choi, Christopher C Paige, and Michael A Saunders. Minres-qlp: A krylov subspace method for indefinite or singular symmetric systems. *SIAM Journal on Scientific Computing*, 33(4):1810–1836, 2011.