TRANSFER LEARNING FOR RELATED REINFORCEMENT LEARNING TASKS VIA IMAGE-TO-IMAGE TRANSLATION

Peter Xuanyuan Chai, Danang Prawira Nugraha & Nagaarjun Nagarajan
University of Southampton
{xc5g15,dpn1n18,nn5g15}@soton.ac.uk

ABSTRACT

The original paper explored the possibility of transfer learning instead of retraining deep RL agents, such that the RL agent is able to work effectively through the GAN generated domain from another domain without retraining. Our experiment set out to re-implement the original author's code, validate its functionality and performance and analyse the advantages of the proposed method compared to re-training an RL agent.

1 INTRODUCTION

The paper (Gamrian & Goldberg, 2018) proposes an approach to reduce the training time of a deep reinforcement learning (RL) agent using the idea of transfer learning. The author first pointed out that a deep reinforcement learning fails to adapt to simple variations of the original game it trained on. A RL agent is very unstable to small, trivial changes in the game whereas a human can easily adapt. A GAN is then used to create a visual mapping between the original game and the variation. The RL agent then is played by using the output from the GAN as the input so that it can still score highly instead of total failure. This report evaluates the principle of transfer learning of this type and analyses the methods used by the original author. We tried to implement the overall system with the author's code but failed to do due to reasons explained in later sections, and only managed to individually train the RL part and the GAN part.

1.1 RELATED WORK

Deep reinforcement learning algorithms recently have achieved human level performance in a variety of tasks. One of the domains is on playing games like Atari 2600 games where the agent can be trained by using the pixels data along with the other related information (Mnih et al., 2013). These results can then be improved by using the Asynchronous Advantage Actor-Critic method (A3C) (Mnih et al., 2016) which produces much better scores with the same training time. Later on, OpenAI (2017) introduces the synchronous version called Advantage Actor-Critic method (A2C) which produces equal performances while also being cost effective due to its usage on a single GPU rather than A3C that only runs using multiple CPU threads.

As an effort to address the generalization problem of an RL agent, the author use Generative Adversarial Network (Goodfellow et al., 2014) that learns to visually map from a target image back to the source image. UNIT (Liu et al., 2017) is used to train a GAN that performs unsupervised learning since there are no paired corresponding images between the original game and the target variation of the game.

1.2 TARGET QUESTIONS

- Is it plausible to train an image-to-image translation GAN that can map a variation of the game back to the original version of the game?
- Is it plausible to use the image-to-image translation GAN such that the RL agent can still score highly on the game variation?

2 EXPERIMENTAL SETUP

2.1 IMPLEMENTATION DETAILS

The authors have made their code public [1] and the implementation uses PyTorch. We fully use the author's code to reproduce the experiment results. However, there are inconsistencies both in the documentation and the code implementation. For example, some arguments are actually disabled by default or code was out-dated. The train implementation of Breakout A3C also neither shows any reward logging nor save the model's checkpoints during the training.

2.2 GAN

As per the original code, two UNIT GANs were trained on datasets generated by untrained RL agents. One GAN was designed to transfer the source task to the target task, and the other vice-versa in accordance to the Cycle-Consistency Principle in Gamrian & Goldberg (2018). Target iterations were set to about 300k to match the original author's specifications. However due to very long training times and limited resources, we were only able to train one GAN to that specification.

2.3 RL

The RL agents (A2C and A3C) needs to be trained under certain constraints so they can score reasonably high in their corresponding game. Therefore, the training was set to terminate when one of the following conditions are satisfied:

- **Maximum number of updates reached:** The number of updates depends on three factors: **Number of processes**, **Number of Frames** and **Number of Steps**.
  The performance of the model is mainly affected by the number of frames collected. Increasing the number of processes decreases the training time. The following values were chosen: *Number of frames:* $10e^6$ *Number of steps:* 20 *Number of processes:* 8 (depends on the computational resources available). Using these values gives rise to 625,000 updates. This was expected to take 26 hours using the available computational resources.
- **High Score reached:** Gamrian & Goldberg (2018) suggests that upon reaching a score of 300 in Breakout or 10,000 in RoadFighter, it is safe to assume that the agent has learnt to play the game.

2.4 REPRODUCIBILITY COST

The paper does not give any details about hardware specification used to perform the experiments. We describe the cost to train both reinforcement learning agent and the GAN as below.

- Breakout
  - **A3C training**: Since the algorithm uses only CPUs, it also needs a lot of memory during the training. After training for 12 hours, it uses 10 GB of RAM. We performed the training on a 8-cores CPU, 30 GB of RAM and using 6 threads[2].
  - **A2C training**: We also tried to train the Breakout using A2C algorithm [3]. It is able to reach mean reward of $\sim$ 300 after 3 hours of training using a single GPU on the yann GPU server [4].
  - **UNIT GAN training**: The average training time for the UNIT GAN was $\sim$14,778 iterations per hour for each variation, using either the yann GPU server or personal GTX970 enabled desktop.[5]

[1] https://github.com/ShaniGam/RL-GAN

[2] using a GCP free tier instance

[3] https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail

[4] yann.ecs.soton.ac.uk

[5] Estimates were used because we could not track exact times on yann server due to running out of GPU memory or other errors, causing us to resume the training and not have a clear running time.

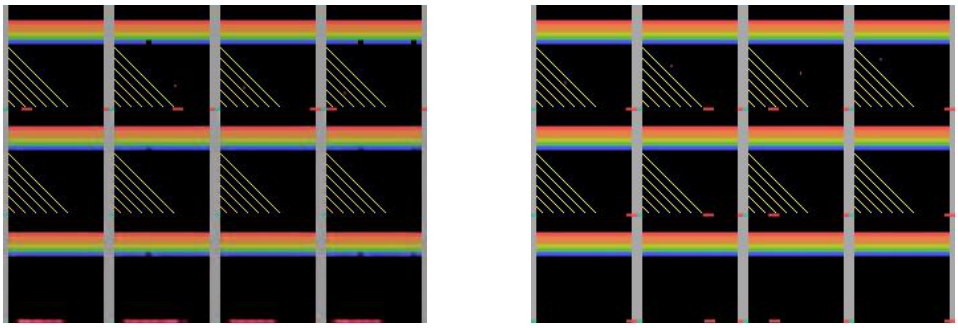
  - diagonals variation - ∼22hrs total (325k iterations)
  - green-lines variation - ∼8hrs total (120k iterations)
  - moving-squares variation - ∼15hrs total (220k iterations)

- RoadFighter
  - **A2C training**: We trained the RoadFighter A2C using the TPU platform of Google Colab [6]. To reach 20,000 updates, we need to run it for 10 hours.
  - **UNIT GAN training**: Trained level 2 to level 1 for 9,200 iterations for about 5 hours. Failed to train other levels due to lack of computational resources and long training times.

3 RESULTS

3.1 GAN

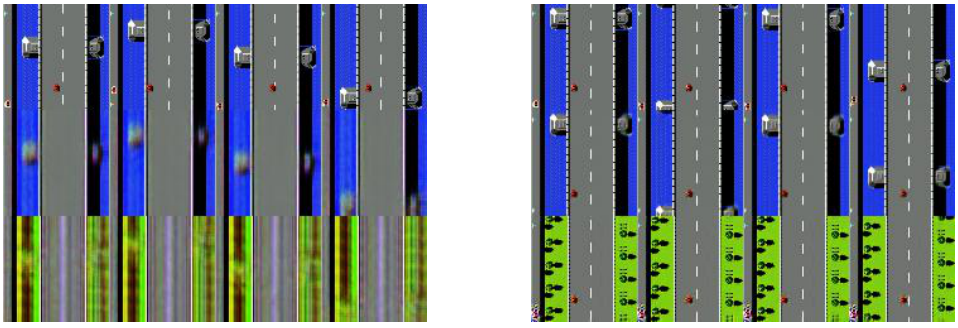
For both Breakout and RoadFighter, we managed to train the GANs to map the target domain to the source domain. In Breakout, we trained three out of the four variations, and the results are stored as GIFs in the repository. Figure 1 shows outputs of UNIT GAN trained on the Breakout dataset.



(a) Inputs and outputs of GAN at iteration 900 (b) Inputs and outputs of GAN at iteration 21500

Figure 1: The top row of the images represent the target domain, bottom row is the reconstructed source domain from target domain, and middle row represents the opposite

You can notice in the diagonals, early on in the training, the GAN was unable to fully reconstruct the source image from the target domain, especially when looking at the paddle at the bottom. After about 21500 iterations, the paddle was reproduced correctly, although smaller features that move every frame, like the ball, still does not appear to be learned until at least iteration 100k.



(a) Inputs and outputs of GAN at iteration 100 (b) Inputs and outputs of GAN at iteration 9200

Figure 2: The top row of the images represent the target domain, bottom row is the reconstructed source domain from target domain, and middle row represents the opposite

[6] http://colab.research.google.com

It is clear on examining Figure 2 that the initial outputs of the GAN at early iterations produce extremely vague outlines of the source domain, without any details on the car and full of artefacts. The GAN trains quickly however, and manages to extract details of the trees, the car, and even the dotted lines by iteration 2000. Despite this, turns in the road or existence of other cars on the road fail to be reproduced by the transfer learning at this number of iterations.

3.2 Reinforcement Learning Agent

For Breakout, we trained the A3C model for 10 hours and the model able to achieve mean reward of 360. However, when we tried to perform the transfer learning to play the diagonals variation, it is only able to reach a score of 40.

A2C model on RoadFighter was left to run for about 24000 out of 625000 updates (10 hours) until hitting problems like running out of memory and storage, timeouts and bus-errors were encountered. During training, most epochs scored 0, and there were some instances where the agent manaaged to score points with a mean reward around 300 and a maximum of 4000.

4 Conclusion

The paper provide enough details complemented with their implementation code to reproduce the experiments. We trained the RL agent and the UNIT GAN for both Breakout and RoadFighter. We did not manage to fully reproduce the results for reinforcement learning with transfer learning using GAN to perform the image to image translation.

We thought that the approach described from the paper does not entirely solve the RL generalization problem. In reality, the policy to play the game does not generalise; it only tricks the agent to be able to play using the existing policy. Also, we thought that all the hassles and time spent training both initial RL agent and the GANs were not rewarding enough since the end RL agent performance that play the game variations is still sub-par compared to when we just fully train the RL agent from the beginning.

We propose the following to improve our current results:

- Use better hardware to achieve better training results in shorter time. For example, Mnih et al. (2016) use a 16 cores of CPU to achieve above 300 score in Breakout with only 4 hours of training. Using a better GPU to train the UNIT GAN also helps.
- Train the GAN using more iterations to produce better mapping from the target to the source image.

References

Shani Gamrian and Yoav Goldberg. Transfer Learning for Related Reinforcement Learning Tasks via Image-to-Image Translation, 2018. URL https://openreview.net/forum?id=rkxjnjA5KQ.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, pp. arXiv:1406.2661, jun 2014.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised Image-to-Image Translation Networks. *CoRR*, abs/1703.00848, 2017. URL http://arxiv.org/abs/1703.00848.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. 2013.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783.

OpenAI. OpenAI Baselines: ACKTR & A2C, 2017. URL https://openai.com/blog/baselines-acktr-a2c/.