Learning to Paint using Reinforcement Learning

CS771A Project Presentation

Abhinav Kumar
Deepankur Kansal
Soumyadeep Datta
Sugam Srivastava

IIT Kanpur

December 12, 2020

Outline

Painting using Strokes learnt via DRL

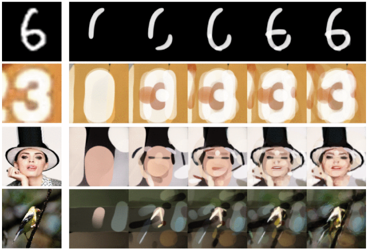


Figure 1: Target Image and painting progression

We base our project on the work done by Huang *et. al.* in [1]. The task of painting comprises the following sub-tasks:

- Describing each stroke in a continuous parameter space (location, colour, transparency etc.) and transforming into a simulation.
- Decomposing target images into an ordered sequence of strokes.
- End-to-end training without human expertise and stroke tracking.

Overall Network Architecture

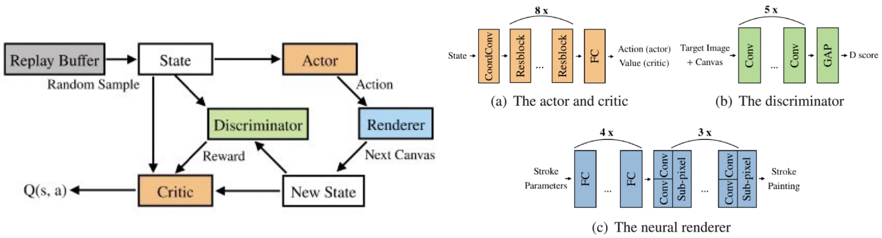


Figure 2: Overall Network Architecture

The network components are summarized as follows:

- Stroke-based renderer: Renders each action (stroke) in the environment (canvas) to produce the painting.
- Model-based DDPG: Decomposes target images into ordered sequence of strokes (actions) that best represent it.
- GAN discriminator: Provides reward for each stroke by comparing with target image.

Stroke-based Renderer

The stroke-based renderer has the following design components:

- Stroke design: quadratic Bézier curve (QBC) represented as the tuple

$$a(t) = (x_0, y_0, x_1, y_1, x_2, y_2, r_0, t_0, r_1, t_1, R, G, B)_t,$$

  where (R, G, B) controls colour, (r_0, t_0) and (r_1, t_1) control thickness of two endpoints and $P_0 : (x_0, y_0)$, $P_1 : (x_1, y_1)$ and $P_2 : (x_2, y_2)$ are control points of QBC. Equation of QBC is

$$B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2, \quad 0 \leq t \leq 1. \tag{1}$$

- Neural renderer: Employs model-based, differentiable transition dynamic $s_{t+1} = \text{trans}(s_t, a_t)$. Input - stroke parameters a_t, output - rendered image $\mathcal{S}$, producing the new state.

Model-Based DDPG

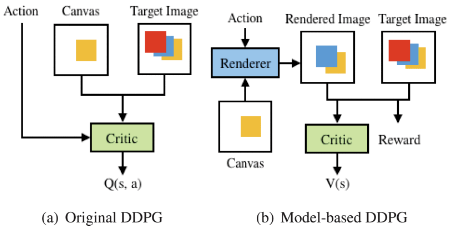


Figure 3: Original [2] and Model-Based DDPG

- Difficult for critic network to model the complex environment. Neural renderer models the action (stroke) in the environment (canvas).
- The generated image forms the next state, s_{t+1}, which is fed to GAN discriminator to obtain reward.
- Value function is used in place of Q-value as new expected reward to be maximized. (γ: discount factor)

$$V(s_{t+1}) = r(s_t, a_t) + \gamma V(s_t).$$

Discriminator

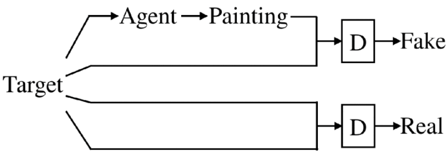


Figure 4: Discriminator Training

GAN: A Popular loss function in image restoration as it measures the distance in distribution between generated and target images.

- The Wasserstein GAN-GP [3] is employed, it maximizes the Wasserstein-I or Earth-mover distance, defined as

$$\max_{D} \mathbb{E}_{y \sim \mu}[D(y)] - \mathbb{E}_{x \sim \nu}[D(x)], \tag{2}$$

  where D: discriminator, ν/μ: distributions of fake/real samples.
- Difference of D-scores between s_t and s_{t+1}, obtained from (2), is fed as reward, $r(s_t, a_t)$, to the DDPG training module.

Improvements in the Renderer

We propose the following improvements:

- Renderer model enhancements:
  - Added dropout and convolution layers to prevent overfitting as well as learn more composable features
  - Optimized renderer to minimize L1 and L2 loss.
- Feature for conversion to Pixel Art:
  - An additional feature to baseline neural renderer
  - Make pixel art using the current DDPG model
  - Employs 2 methods :
    - pre-processing
    - post-processing.
  - Post Processing gives better results during comparison
  - Custom strokes during neural renderer can overcome difference in image quality

Improvements to the DDPG Algorithm

We implement the Prioritized Experience Replay (PER) technique [4].

- In *experience replay*, default: uniform sampling from replay buffer.
- PER assigns weights to the samples based on their importance. More important samples are used more frequently in training.
- The idea is to pick those samples first which have a higher error magnitude, $|\delta_i|$, where the error is defined as

$$\delta_i = r(s_t, a_t) + \gamma V^{'}(s_{t+1}) - V(s_t).$$

The priorities, p_i, are defined *proportionally* as $p_i = |\delta_i| + \epsilon$. The weights assigned are of the form

$$w_i = \left(\frac{1}{N}\frac{1}{P(i)}\right)^{\beta},$$

where the probabilities, $P(i)$ are calculated as

$$P(i) = \frac{p_i^{\alpha}}{\sum_k p_k^{\alpha}}.$$

Improvements to the Discriminator

Originally, discriminator of WGAN-GP (weight normalized)[3] was used. We experimented with different architectures of discriminators independently as followed:

- DRAGAN[5] (slightly worse performance)
- FisherGAN[6] (slightly better performance)
- CramerGAN[7] (slightly better performance)
- SNGAN[8] (better performance)
- SAGAN[9]'(best performance till experimentation)

The performance is tested on training loss for a fixed number of iterations (5k) as a reward metric to the critic to check for the fastest learning.

Simulation Results:Renderer

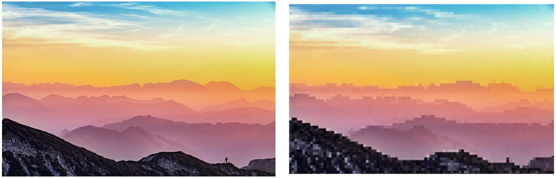
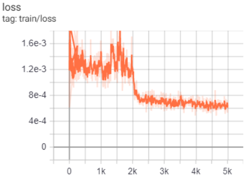


Figure 5: Normal and Pixel Art made by Models



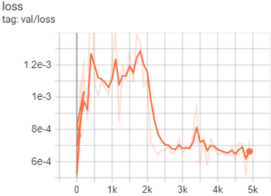


Figure 6: Final Train and Validation Loss vs Steps

Simulation Results: DDPG with PER

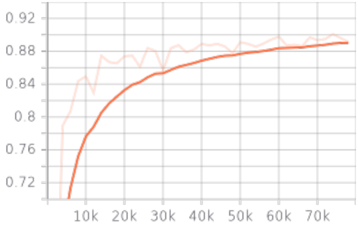


Figure 7: DDPG with PER (test reward)

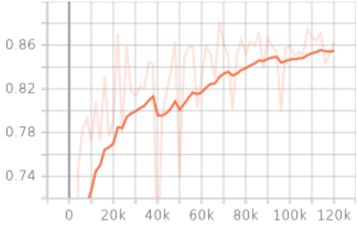
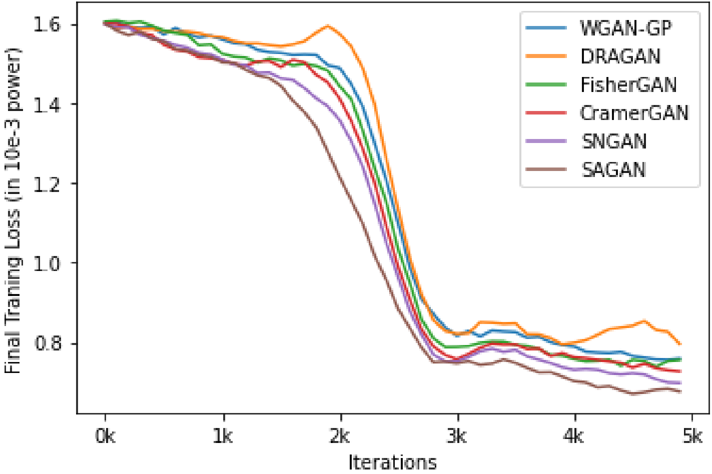


Figure 8: DDPG without PER (test reward)

Simulation Results:Discriminator Loss



Comparison of different GANs using final training loss for a fixed number of iterations(5k) under the purview of fastest convergence.

Summary and Issues Faced

- We improve Renderer.
- Added an Pixel Art Stroke.
- We improve the model-based DDPG algorithm by employing *Prioritized Experience Replay*, which hastens the learning process.
- We improve Discriminator by using SAGAN architecture as the discriminator for reward which makes learning faster.
- Possible extensions:
  - Make even better renderer.
  - Employ more advanced versions of DDPG, notably Twin-Delayed DDPG (TD3) [10], for more stability in learning.
  - Make even better discriminator by using improvements to SAGAN which are heavy in computational constraint.
  - Extend same techniques to some other application.
  - Add some more types of strokes
- DDPG implementation with Parameter Noise [11] did not learn anything as here a model-based DDPG with CNN layers is used whereas as original paper uses model-free DDPG.

References I

Z. Huang, W. Heng, and S. Zhou, "Learning to paint with model-based deep reinforcement learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *CoRR*, vol. abs/1704.00028, 2017. [Online]. Available: http://arxiv.org/abs/1704.00028

T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2016.

References II

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *CoRR*, vol. abs/1704.00028, 2017. [Online]. Available: http://arxiv.org/abs/1704.00028

Y. Mroueh and T. Sercu, "Fisher GAN," *CoRR*, vol. abs/1705.09675, 2017. [Online]. Available: http://arxiv.org/abs/1705.09675

M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The cramer distance as a solution to biased wasserstein gradients," *CoRR*, vol. abs/1705.10743, 2017. [Online]. Available: http://arxiv.org/abs/1705.10743

References III

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *CoRR*, vol. abs/1802.05957, 2018. [Online]. Available: http://arxiv.org/abs/1802.05957

H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," 2019.

S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of Machine Learning Research*, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1587–1596. [Online]. Available: http://proceedings.mlr.press/v80/fujimoto18a.html

References IV

M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, “Parameter space noise for exploration,” 2018.