Reinforcement Learning Based Generative Art Agent

Experimental Protocol

Nathan Grabaskas
Zhizhen Wang

1. Hypothesis

The problem being discussed in this protocol is teaching an artificial intelligent agent how to draw, Drawing and creative art are a critical part of human civilisation and culture. To learn how to draw would take years of learning and practising for humans. Hence we want to explore the idea of training an artificial intelligence agent that can paint strokes on a canvas in sequence to generate a new painting.

Similar work has been done by Zhewei, Wen and Shuchange [Zhewei et al. 2018] where they trained an AI agent that can paint on a canvas to generate a painting that is similar to the target image. Apart from training for a reward policy for the agent, the team also proposed an approach to decompose the target painting into hundreds of strokes in sequence in a grid. In the end of the project, the agent is trained to be general enough to handle multiple types of images (including digits, handwritten, streetview, human portains and etc.). Another work related to artwork generation is done by Ning and his team [Ning et al. 2015]. The team is focusing on a particular type of painting, stroke drawing. They applied inverse reinforcement learning to learn the reward function from the training data.

This project is not only trying to create an artwork that is similar to an original one, but to explore the random artwork an agent can generate during the training process. Based on the goal, the agent would be measured against both quantitative and qualitative metrics.The experiment contains two parts. First part is training the agent. We will start by defining a 2D environment (canvas) for agents to simulate the painting, and each training canvas will be initialised with a set of negative rewarding grid, positive rewarding grid and hazard zones. The initial parameters for the rewarding would be based on abstract art paintings. We will be pre-processing those images to extract only the edges and shapes of the painting. Then the canvas grid point will be defined accordingly. The next step is to allow the agent to explore the environment while avoiding danger zones, maximize coverage given a set amount of energy, and maximize surface area covered if the agent had a "sprayer" attached.

2. Dataset

In this project, we use an image dataset together with Gym-MiniGrid to create the environment for the agent to explore and optimise.

The dataset that is going to be used in the project is from the Kaggle Open dataset for abstract art images provided by Greg Surma. This dataset constraints 8145 512 * 512 abstract art paintings. The team will create a method to decompose and transform the raw image into 2D grid representation. The preprocessing process includes resizing, aligning, setting threshold for detecting the edge, blurring and defining the hazard zones.

Gym-MiniGrid is used as the gridworld environment in the project, it was designed to seek to minimize software dependencies, be easy to extend, and deliver good performance for faster training. The environment comes with the existing object types wall, floor, lava, door, key, ball, box and goal. This gives the agent diverse, generated environments with tasks such as getting the key to unlock the door, hazards like lava to be avoided, and the purpose of getting to the goal.

In the project, the team will replace the default setting for the grid to image represented 2D-grid, and existing objects would be uplifted to echo the painting characteristics (for example different color, shape, and transparency). Then the agent will be trained and evaluated with a variety of image based procedural-generated environments.

3. Metrics

3.1 Quantitative Agent Comparison

Each episode has the grid setup and the agent is given a reward for reaching the goal, this reward decays for each frame of the episode. Quantitative metrics are used to compare agents trained with different algorithms in the same environment. Each agent is evaluated on 100 episodes of the procedurally generated environment. The mean reward and mean number of frames to end the episode are used to compare the agents. With a higher reward and lower frames to solve as the desired values.

3.2 Machine Learning Theory Modules

Error Analysis Diagnostics - Look at what would be perfect performance and assess where current performance is failing. Like the simple A2C optimization algorithm is able to train well on MiniGrid-MultiRoom-N2-S4-v0 but when even a trained agent is put into MiniGrid-MultiRoom-N4-S5-v0, neither algorithm is able to handle the complexity of the multiple rooms. Things such as if the agent had perfect vision of the environment, how well would it do in the complex scenario? Or what if all obstacles were removed, how well would the agent do in this scenario? Additionally, we may include screenshots of neural activity as the models are learning. To train and visualize breakdowns.

Potential Scenarios to Analysis
Agent with infinite view distance (at least to the end of the environment).

Environment with all obstacles (keys, locked doors, lava, etc.) removed.
Large, open environment with small and large view distance (This can show how lack of seeing hinders exploration and how intrinsic reward can help this)

Ablative Analysis - Ideally, we would like to do some study on how freezing or stimulating different areas of the CNN would affect the quantitative metrics. However, due to time constraints of the course, we will look at how adding in an LTSM (memory) could aid the agent training, especially in multi-room, multi-task environments.

3.3 Qualitative Comparison

Ultimately, we want to generate new pieces of art from these trained agents. To get the image visualizations we will take a 100 episode heatmap of the agents movements over the grid environment generated from the existing artwork. These images will be placed in front of human raters and asked which one (A/B comparison test) they prefer. Raters will also be given a single image and asked to rate its interestingness on a scale of 1 to 5. Rating is conducted through the use of crowdsourcing through theHive.ai. Due to time constraints, we may only get one pass at this and will be limited to a small number of raters. This will give an answer to the difficult question, “how good is this art”.
The major challenge with the qualitative comparison is the additional time required to complete agent training, environment generation, rating questions, and get results back from the raters. We will be unable to make multiple iterations of this trying to improve each time.

Rater Question	**Possible Answers**
With a single image, “How interesting do you find this abstract artwork?”	1, 2, 3, 4, 5 with 5 being very and 1 being nor
With an A/B comparison of images, “Which piece of artwork do you find more interesting?”	A, B, Neither

4. Models

4.1 Model Inputs

Input to the model is the agent’s view of the grid environment. For example if the view distance is set to 7, then the input would be an array of size (3,7,7). Each tile is encoded as a 3 dimensional tuple: (OBJECT, COLOR, STATE) [Chevalier-Boisvert et al, 2018].

Dimension	**Represented Values**
STATE	open, closed, and locked

OBJECT	wall, floor, lava, door, key, ball, box, and goal
COLOR	red, green, blue, purple, yellow, and grey

4.2 Model Policies

A2C (Advantage Actor-Critic) [Mnih et al, 2016]- The Advantage function captures how much better an action is compared to others at a given state, while the value function captures how good it is to be at this state. This way the evaluation of an action is based not only on how good the action is, but also how much better it can be. The benefit of the advantage function is that it reduces the high variance of policy networks and stabilizes the model.

Advantage actor-critic (A2C) maintains a policy and an estimate of the value function. A2C operates in the forward view and uses the same mix of time step returns to update both the policy and the value function. The policy and the value function are updated after every t_max actions. The update performed by the algorithm is update_loss = policy_loss - entropy_coef * entropy + value_loss_coef * value_loss. We use a convolutional neural network that has one softmax output for the policy (actor component) and one linear output for the value function (critic component) with the CNN layers shared.

A2C Intrinsic Rewards - After backward propagation of the model using the update_loss, there is a second round of backward propagation using loss = L2 Norm (weights[S_t+1] - weights[S_t]) for all layers in the CNN [Raileanu and Rocktäschel, 2016]. This intrinsic reward is to encourage the agent to take actions that lead to significant changes in its learned state representation.

4.3 Model Architecture [Lucas Williams, 2018]

```
(image_conv): Sequential(
   (0): Conv2d(3, 16, kernel_size=(2, 2), stride=(1, 1))
   (1): ReLU()
   (2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0)
   (3): Conv2d(16, 32, kernel_size=(2, 2), stride=(1, 1))
   (4): ReLU()
   (5): Conv2d(32, 64, kernel_size=(2, 2), stride=(1, 1))
   (6): ReLU()
)
(actor): Sequential(
   (0): Linear(in_features=64, out_features=64, bias=True)
   (1): Tanh()
   (2): Linear(in_features=64, out_features=7, bias=True)
)
(critic): Sequential(
   (0): Linear(in_features=64, out_features=64, bias=True)
```

```
    (1): Tanh()
    (2): Linear(in_features=64, out_features=1, bias=True)
)
```

5. General Reasoning

To achieve the goal that has been mentioned in the hypothesis section, the project would leverage a few existing libraries, packages and algorithms to train the agent.

1. Use existing abstract artworks from Kaggle open dataset as the raw image, then create a general method to generate new grid environments that can represent the artwork for agents to explore.
2. Integrate the new grid environments with the Gym-MiniGrid project. And use it as the experiment baseline.
3. During the training period, expand on the algorithm Advantage actor-critic (A2C) and Proximal Policy Optimization (PPO) with intrinsic rewards for state changes, hence encouraging the agent to explore the generated environments.
4. Based on a variety of generated environments, test on a combination of different algorithms and reward functions.
5. Use both quantitative and qualitative metrics to evaluate the agent performance.
6. Compare the agent's performance between default MiniGrid environments and artwork derived environments. The comparison should also be conducted when training on different policy optimisation algorithms.

The above process would provide a consistent and stable method to train the agent. After sufficient interaction, the agent would gain enough knowledge to generate meaningful sequential steps which would be rendered to artworks.

In order to generate an artistically pleasing art, the team also needs to build a separate painting environment to render the agent's movement into visual arts. This task has been considered optional since the goal of this project is to find the best policy for agents to learn how to draw, rather than creating meaningful arts.

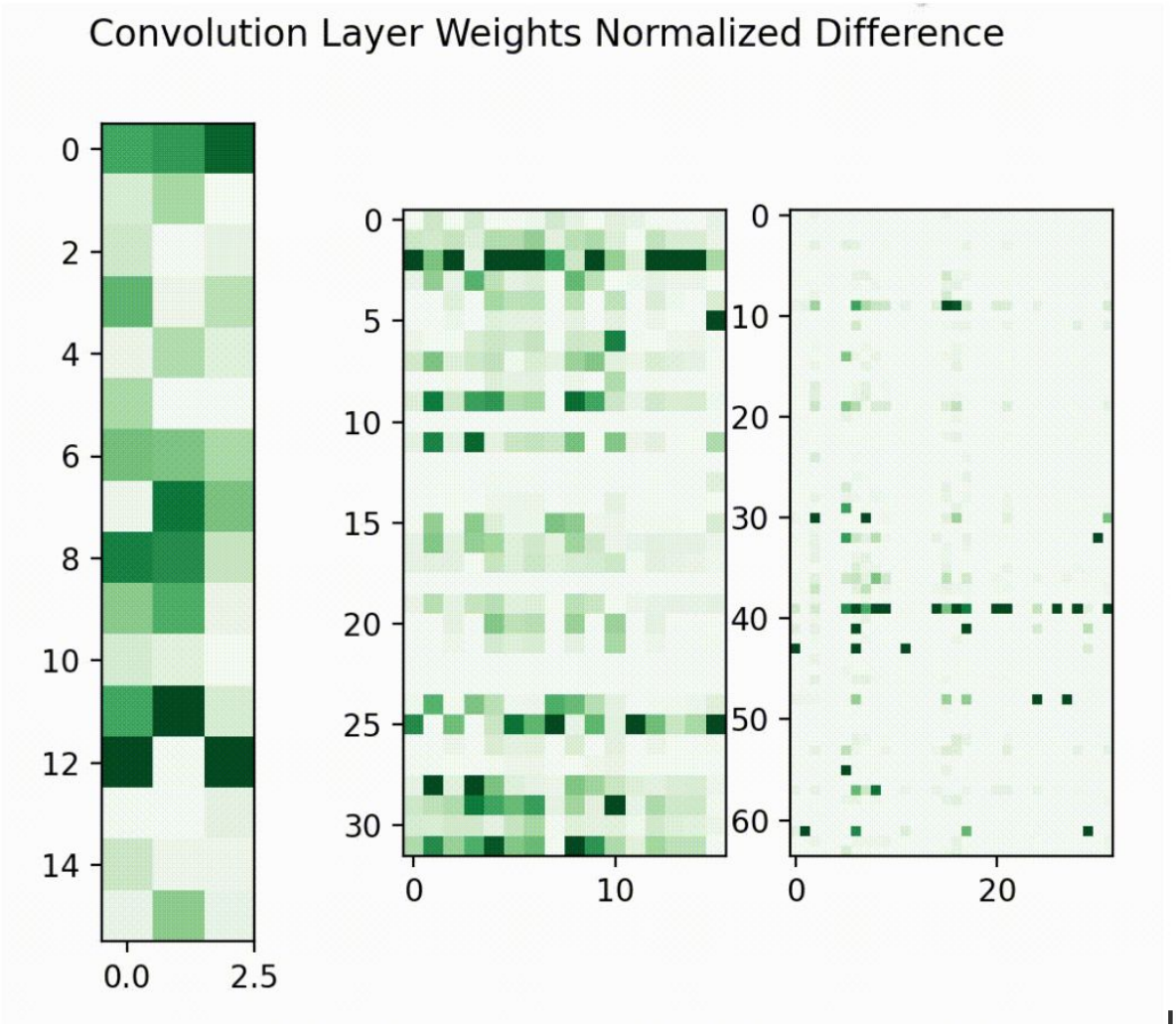
6. Summary of progress so far

The team has set up the base repo to build up the training environment. There are three repositories we combined to lay the foundation for our experiments.

1. Gym-MiniGrid [Chevalier-Boisvert et al, 2018] - contains the framework for the grid environment and obstacles/interactions the agent is able to perform.
2. Torch-AC [Lucas Williams, 2018] - contains the base algorithm implementations for PPO and A2C.
    a. A2C and PPO learning algorithms expanded to reward the agent when larger weight changes occur to encourage learning.

3. RL-Starter-Files [Lucas Williams, 2017] - contains the framework for training the agent on each environment, storing model states, and evaluating agent performance.
    a. Expanded to include real time model weight change visualization.

Below is the real-time heatmap generated by the agent when the team is experimenting on different algorithms.



The above heatmap was derived by the CNN three layers added to the model. The darker the color the larger the difference in the weight change for the model update. Shows which areas of the agent's "brain" are learning. This is an agent training using A2C with Intrinsic rewards on MiniGrid-MultiRoom-N4-S5-v0.

Initial testing with original algorithms and implemented algorithms intrinsic reward. In these early experiments, agents trained with intrinsic rewards to perform slightly better than the baseline.

Method	**Environment**	**Avg Reward**	**Avg Frames to Finish**

PPO	MiniGridMR-N2-S4-v0	0.77375	10.1
PPO_I	MiniGridMR-N2-S4-v0	0.7775	9.9125
A2C	MiniGridMR-N2-S4-v0	0.675	14.1
A2C_I	MiniGridMR-N2-S4-v0	0.76	10.65

7. References

- Maruan Al-Shedivat, Lisa Lee, Ruslan Salakhutdinov, Eric Xing. On the Complexity of Exploration in Goal-Driven Navigation.arXiv preprint arXiv:1811.06889. 2018.
- Andres De Biase and Mantas Namgaudis. Creating safer reward functions for reinforcement learning agents in the gridworld. University of Gothenburg Chalmers University of Technology. Sweden 2018.
- Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.
- Zhewei Huang, Wen Heng, Shuchang Zhou. Learning to Paint With Model-based Deep Reinforcement Learning.arXiv preprint arXiv:1903.04411 . 2019.
- Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pp. 1928–1937, 2016.
- John Schulman, Filip Wolski, Praffula Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. 2017.
- Istvan Szita and Andras Lorincz. Learning Tetris using the noisy cross-entropy method. Neural computation 18.12, pp. 2936–2941. 2006.
- Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. International Conference on Learning Representations, 2020.
- Lucas Williams. PyTorch Actor-Critic deep reinforcement learning algorithms: A2C and PPO. https://github.com/lcswillems/torch-ac. 2018.
- Lucas Williams. Reinforcement Learning Starter Files. https://github.com/lcswillems/rl-starter-files. 2017.
- Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample Efficient Actor-Critic with Experience Replay. arXiv preprint arXiv:1611.01224. 2016.
- Ning Xie, Tingting Zhao, Feng Tian, Xiaohua Zhang and Masashi Sugiyama. Stroke-based stylization learning and rendering with inverse reinforcement learning. IJCAI'15: Proceedings of the 24th International Conference on Artificial IntelligenceJuly 2015 Pages 2531–2537