Exploring RL Related "Teaching Computers to Paint" and Decision Transformer

Du Xiang
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA
xd00099@berkeley.edu

Zhiwei Zheng
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA
zhiwei.zheng@berkeley.edu

Abstract

In this paper, we present a study on the use of Deep Reinforcement Learning to teach a computer to paint. We aim to explore the potential of applying Decision Transformer to create visually appealing and original paintings.

To achieve this, we first reimplemented the model-free RL approach from "ICCV2019-Learning to Paint" and trained it on MNIST datasets. We successfully demonstrated the model's ability to extrapolate to other datasets, such as writing Chinese characters and Japanese characters.

Before diving into the challenge of learning to paint using Decision Transformer, we first wanted to explore how the model would perform on a more comprehensive and real-word dataset. Therefore, we trained a Decision Transformer model on the NeoRL dataset and evaluated its performance compared to behavioral cloning. The NeoRL dataset was chosen because it is a conservative and limited dataset. Since in the future we would generate a stroke dataset ourselves, testing the Decision transformer's performance on a dataset with similar characteristics first is important.

From NeoRL experiment, we had several findings. 1) In the most of cases, Decision Transformer showed better results, compared to Behaviour Cloning. 2) As the size of the dataset increases, the performance of the Decision model would increase or almost remain the same. However, the Behaviour Cloning didn't behave similarly. In datasets collected by sub-optimal policy, the Behaviour Cloning even behaved worse as the size grew up, while the Decision Transformer did not. 3) We find the quality of trajectories in one dataset is much more important than the number of trajectories in one dataset. 4) Decision Transformer performed well in each task of NeoRL benchmark, proving Decision Transformer has some ability to learn and generalize from these conservative and limited datasets. After analyzing the results, We decided to continue our study.

To teach Decision Transformer to paint, we would need a lot of offline datasets, but there is no existing offline datasets for painting strokes. Thus, we created an expert offline dataset ourselves by using a trained model from "ICCV2019-Learning to Paint" to produce trajectories during inference.

We ran an agent on the MNIST and CelebA datasets for inferences, and collected 3000 trajectories from each dataset. Each trajectory consisted of 40 steps of the agent's observations, actions and rewards. This dataset provided a source of data for evaluating the performance of Decision Transformer and comparing it

to behavioral cloning. At first, we wanted to create one large dataset as much as possible, since the large size and diversity of the dataset would allow us to train a better Decision Transformer model.

Finally, we applied the offline dataset generated from the painting agent to Decision Transformer and evaluated the results. However, one problem encountered here is that due to the large size of each episode and the limitation of computer memory, we can only use a maximum of 1000 trajectories and even so the training would take more than 4 hours per iteration. The results showed that the average return of the evaluation was 0.2673, the corresponding standard variance was 0.2788, while these numbers in dataset were 0.11 and 0.04 respectively. Although the mean was higher than that of dataset, the std was so big that in fact, the model could not fulfill the painting task.

In conclusion, our study shows Decision Transformer dose have the ability to solve some real-world problems, rather than only some games or simulated problems. However, speaking of painting task, we have to say what we did so far cannot guarantee the Decision Transformer model can completely be qualified for this job. After further reflection and analysis, we proposed these improvements. 1) Using a convolution neural network to reduce the dimension of the input of Decision Transformer. For now, the input consists of the current canvas and target image, which would be a extremely long vector after being flattened. 2) Use more time and resources, and add some noise to collect a larger and more diverse dataset. 3) Exploring transfer learning to improve Decision Transformer's performance on painting task.

1 Introduction

The ability to create visually appealing and original paintings has long been considered a uniquely human trait. However, recent advancements in artificial intelligence and machine learning have begun to challenge this notion. One promising approach is the use of Deep Reinforcement Learning to teach a computer to paint.

The idea of using reinforcement learning to teach a computer to paint emerged from the success of using this approach in other domains, such as playing games and controlling robots. The basic principle is to provide the computer with a set of rules and rewards, and allow it to learn through trial and error. By using this approach, the computer can learn to make decisions and take actions that maximize the rewards and produce desirable outcomes.

In this paper, we present a study on the use of Deep Reinforcement Learning to teach a computer to paint. We explore various RL methods to teach a computer to paint, and evaluate Decision Transformer on a more real-world dataset. In the last part of our study, we test Decision Transformer on our own dataset, but does not get achieve a great result. We reflect on it and provide some possible improvements.

2 Related Work

There has been a significant amount of research on teaching computers to paint using various machine learning techniques. One of the earliest approaches is behavioral cloning, which involves training a model to mimic the behavior of a human painter by supervised learning on a large dataset of human-generated paintings. However, this approach is limited by the quality and diversity of the training data, as the model can only learn to paint what it has seen in the training data.

More recently, model-based Reinforcement Learning (RL) has been applied to the task of teaching computers to paint. In particular, the work of [2] has shown that a model-based RL approach can be used to train a painting agent to generate high-quality paintings that are similar to human-generated paintings. This approach has the advantage of allowing the model to explore and learn to paint novel patterns that may not be present in the training data. Their work also introduce differentiable neural

renderers for efficient painting and flexible support of different stroke designs, such as Bezier curves, triangles, and circles. This allows the model to be trained in an end-to-end fashion and improves the painting quality.

Another recent development in the field is the Decision Transformer, which has been proposed as a more sample-efficient and effective approach for RL [1]. This model combines the strengths of both transformer-based models and model-free RL, allowing it to learn more efficiently and generalize better to unseen situations.

In this paper, we build upon the work of [2] and [1] by applying Decision Transformer to the task of teaching a computer to paint. We use the model-based RL approach of [1] to generate an offline dataset for training and evaluating painting agents. We compare the performance of Decision Transformer to behavioral cloning and the model-based RL approach of [1] on various datasets. Our work contributes to the existing literature by showing the potential of Decision Transformer for this task and providing a new offline dataset for training and evaluating painting agents.

3 Dataset and Data Collection

3.1 MNIST – Paint

The MNIST dataset [3] is a widely used dataset for evaluating machine learning algorithms on the task of handwritten digit recognition. It consists of 70,000 grayscale images of handwritten digits, with 60,000 images for training and 10,000 for testing. Each image is 28x28 pixels in size and contains a single handwritten digit from 0 to 9. The dataset is balanced, with an equal number of images for each digit.

Figure 1: MNIST Dataset

We use the MNIST dataset to train our painting agent to generate images of handwritten digits. This allows us to evaluate the ability of the agent to learn to paint novel patterns and generalize to unseen digits.

3.2 CelebA – Paint

In this work, we also use the CelebA dataset [4] for training and evaluating our painting agent. The CelebA dataset is a large-scale dataset of celebrity face images, with over 200,000 images. Each image is centered on the face and is cropped to remove the background. The images are then resized to 64x64 pixels. The dataset contains images of over 10,000 different celebrities, representing a wide range of facial appearances and expressions.

Figure 2: CelebA Dataset

3.3 NeoRL – Decision Transformer

The NeoRL dataset [5] is a new reinforcement learning benchmark that addresses the reality gaps commonly found in previous benchmarks. The evaluation of state-of-the-art offline RL algorithms on the NeoRL dataset reveals that some algorithms may be less effective in real-world tasks than in previous benchmarks, and current offline policy evaluation methods are unable to effectively select the best policy. This work aims to provide insight into future research and the deployment of RL in real-world systems.

The NeoRL dataset is obtained by using the SAC algorithm to train a policy on each environment until convergence, and recording a policy at every epoch. The policy with the highest episodic return is denoted as the expert policy, and other policies with around 25%, 50%, and 75% expert performance are stored to simulate sub-optimal policies. Data is collected by sampling from the trained Gaussian policies with probability 20%, and using the mean of the Gaussian policy for the rest of the data collection. Training and test datasets are then created by selecting four similar policies for each level, randomly selecting three for training data, and using the remaining policy for test data. The default size of the test data is 1/10 of the training data for each task, and the extra test dataset can be used for offline evaluation and model and hyperparameter selection.

The NeoRL dataset we used include three specific domains: **IB, FinRL, and CityLearn**.

The **IB** domain simulates industrial control tasks and includes problems commonly encountered in real-world industrial environments, such as high-dimensional continuous state spaces, delayed rewards, complex noise patterns, and high stochasticity of multiple reactive targets. The behavior policy in this environment is deterministic.

The **FinRL** domain provides a trading simulator that replicates the real stock market and supports backtesting with market frictions such as transaction costs, market liquidity, and investor risk aversion. In this environment, agents can trade once per day for 30 stocks in the pool, and the reward is based on the difference in the total asset value between the end of the day and the previous day. The environment may evolve over time. The training dataset for this domain is too large, so only 102 and 103 trajectories are provided in the NeoRL dataset.

The **CityLearn (CL)** domain involves controlling energy storage in different types of buildings to reshape the aggregation curve of electricity demand. The goal is to coordinate the control of domestic hot water and chilled water storage by the buildings to reshape the overall curve of electricity demand. This environment is highly stochastic and has a high-dimensional space.

3.4 Generated Offline Paint Strokes

We use the model-based Reinforcement Learning approach from [2] to generate a dataset of paint strokes. The RL model is pre-trained on the CelebA datasets. The paint strokes are generated by running the RL model in an inference mode, where the target image is provided and the model generates a sequence of paint strokes that are applied to an empty canvas.

The dataset of paint strokes consists of the painting trajectories of randomly sampled images. Each trajectory represents a single painting episode. The dataset contains a total of 6,000 paint-

ing episodes, 3,000 were sampled from MNIST and 3,000 were sampled from CelebA respectively, each consisting of observation, next_observation, action, reward and terminal flag. The details of each property would be explained in Sec 4.1.

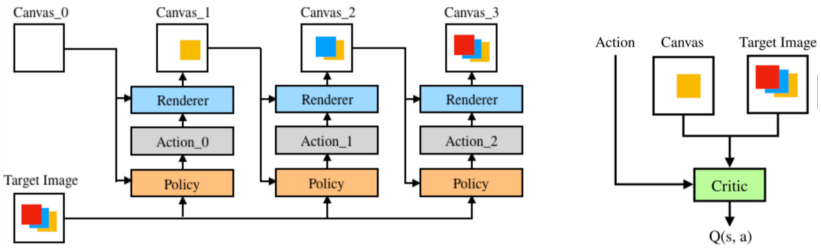
Figure 3: Writing a 5, Generated Offline Paint Dataset with skipped steps

4 Models

4.1 Model-free RL for Painting

For the starting point of our experiments, we re-implemented the model-free Reinforcement Learning (RL) approach [2] to teach a computer to paint. The painting task is modeled as a Markov Decision Process (MDP) with a state space S, an action space A, a transition function $trans(s_t, a_t)$ and a reward function $r(s_t, a_t)$.

The state space is constructed by all the information that the agent can observe in the environment. It is separated into three parts: the states of the canvas, the target image, and the step number. Formally, $s_t = (C_t, I, t)$, where Ct and I are bitmaps and t is the step number. The transition function, $s_t + 1 = trans(s_t, a_t)$, gives the transition process between states, which is implemented by painting a stroke on the current canvas, see part a, Figure 4.



(a) Overall Architecture (b) model-free DDPG

Figure 4: Visual Demonstration of Paint Model from Huang et al. [2]

An action at of the painting agent is a set of parameters that control the position, shape, color, and transparency of the stroke that would be painted at step t. The behavior of the agent is defined as a policy function π that maps states to deterministic actions, i.e. π: S $\rightarrow$ A. At each step, the agent observes the state st and predicts the parameters of the next stroke at. The state then evolves based on the transition function $s_t + 1 = trans(s_t, a_t)$, which runs for n steps.

The reward function is designed to measure the difference between the current canvas and the target image. It is defined as $r(st, at) = L_t - L_{t+1}$, where $r(s_t, a_t)$ is the reward at step t, Lt is the measured loss between I and the Ct, and Lt+1 is the measured loss between I and the Ct+1. In this work, the loss L is formulated as the discriminator score. To make the final canvas resemble the target image, the agent should be driven to maximize the cumulative rewards in the whole episode. At each step, the objective of the agent is to maximize the sum of discounted future rewards.

4.2 Decision Transformer

Decision Transformer [1] is a framework for solving reinforcement learning (RL) problems by modeling trajectories using an autoregressive model. The framework uses the Transformer architecture, which is commonly used in natural language processing tasks, to model the temporal dependencies between states and actions in an RL problem.

The Decision Transformer models the trajectory as a sequence of triplets, each consisting of a return-to-go value, a state, and an action. The return-to-go value represents the remaining rewards to be

obtained in the episode, and is used to specify the desired performance at test time. The state and action are the observations and actions taken at each time step in the episode.

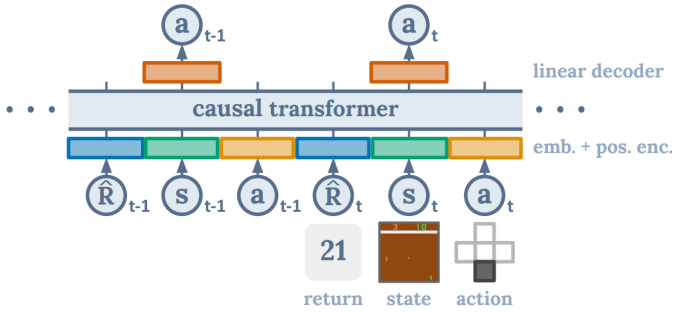


Figure 5: Decision Transformer Illustration [1]

5 Experiments and Results

5.1 Model-free RL on Painting and Extrapolation

The original motivation for the experiment was to explore the potential of using a model-free reinforcement learning approach, especially the Decision Transformer, to teach a computer to paint. To do this, we first reimplemented the model-free RL approach from [2] and trained it on MNIST datasets.

The model-free model 4.1 was trained on MNIST dataset for 12 hours on GPU, and 20 hours on CelebA data. As a reminder, we measured the loss by the pixel-wise distance. Here are the results from training, 5

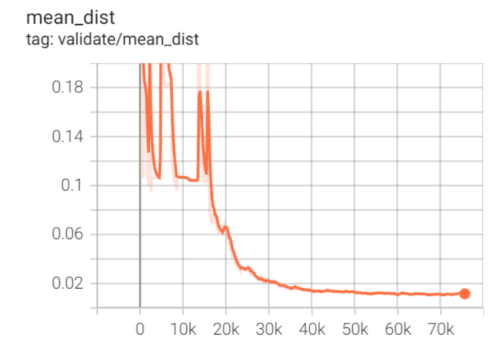
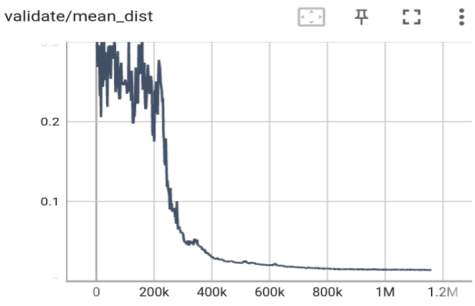
MNIST	CelebA
	

Table 1: Training Results

The trained model was able to perform reasonably well on all the test images, see Table 2.

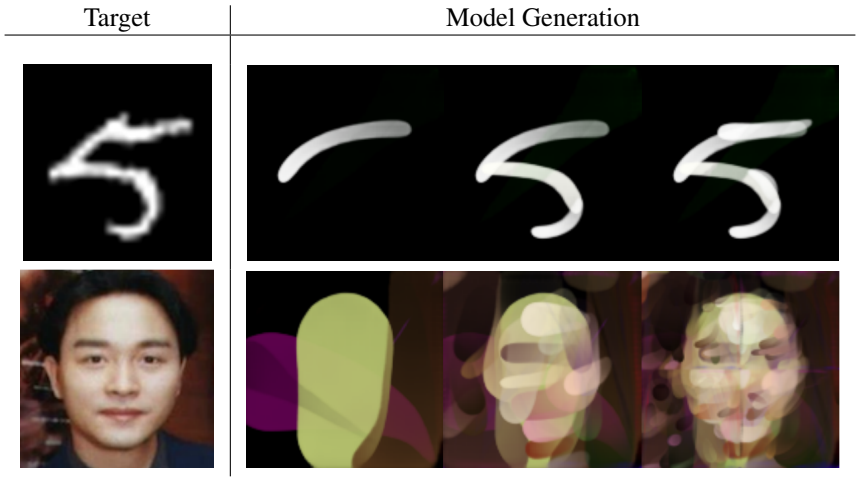


Table 2: Test Results on: row1 - MNIST model, row2 - CelebA model

We also noticed that the model was able to extrapolate really well, which means it did 'learn' how to draw images that have similar styles compared to the training set. See performance on writing Chinese, Figure 6 and Japanese Figure 7.

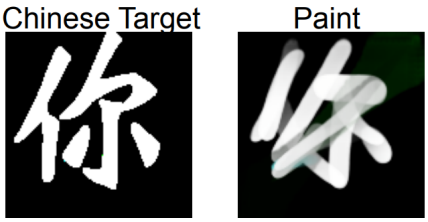


Figure 6: Pretrained MNIST perform on writing Chinese

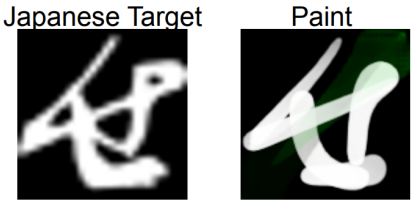


Figure 7: Pretrained MNIST perform on writing Japanese

5.2 Decision Transformer on NeoRL

In this section, we trained and evaluated the Decision transformer model's performance on NeoRL dataset. The hyperprameters for these tasks remain the same as [1] describes in the gym part, except the reward-to-go are set as the following: 167,000 for CityLearn task, 850 for FinRL task and -210,000 for IB task.

For those who are not familiar with NeoRL benchmark, here we also provide the distribution of these three tasks we used in this study 8. We choose to show representative sub-datasets of each task generated by the most optimal policy. Specifically, we use the Expert-10000 subset for CityLearn task, Expert-1000 subset for FinRL task and High-10000 subset for IB task.

After 10 iterations, we had the following results9. Considering the simplicity of figures, we only provide the average return of evaluation, ignoring the standard variance. More details about the evaluation can be found here link.

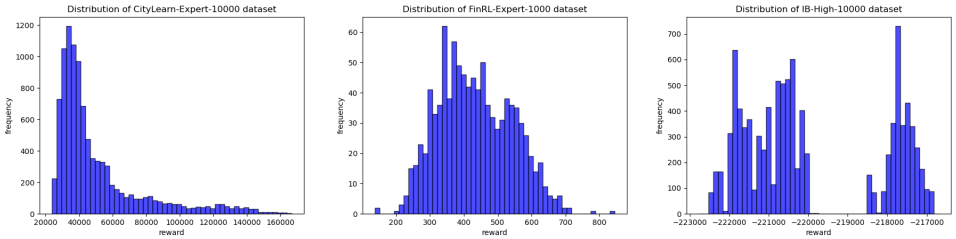


Figure 8: Distribution of datasets

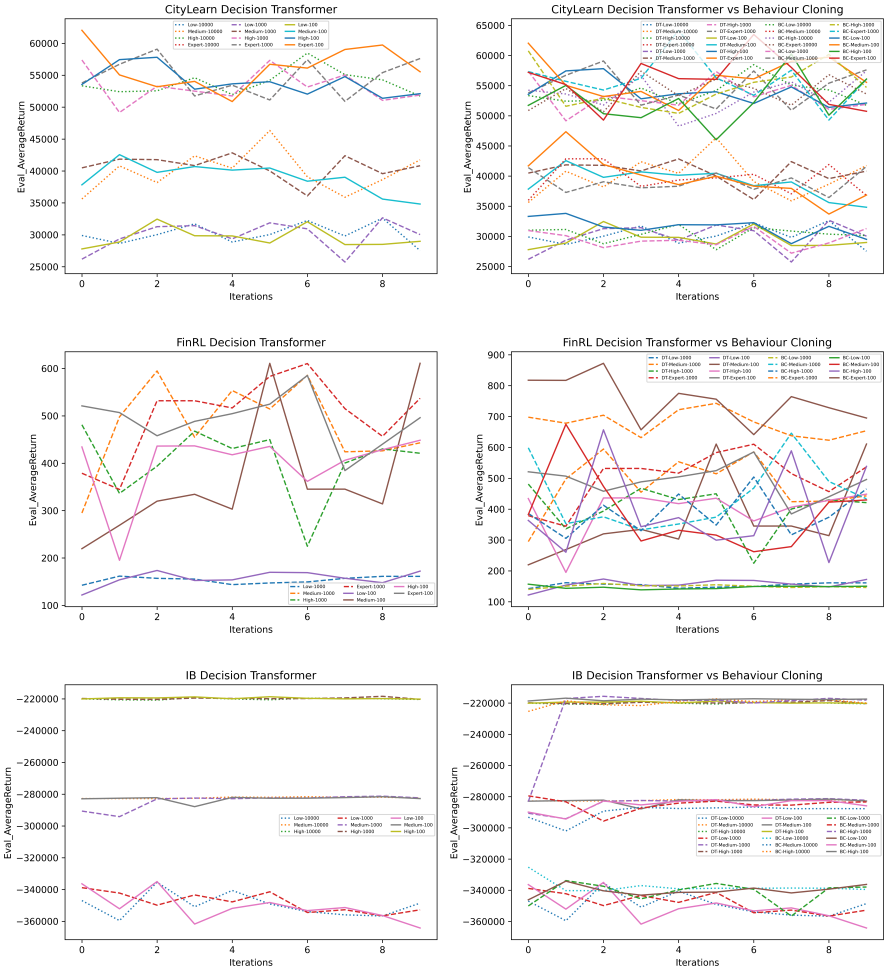


Figure 9: Evaluation on NeoRL Dataset

In the most cases, Decision Transformer had a better performance than Behaviour Cloning. Besides, compared to the distribution of each task, we can find that Decision transformer performed quite well in each task, achieving a higher reward than the average reward of datasets. These results show that Decision Transformer can perform quite well in these real-world tasks, and supports our following study.

Besides, from the experiment results on each task, we can make a conclusion that the quality of dataset is much more important than the size of dataset. Although some dataset might only have limited high-reward trajectories, models can still learn a lot from them and succeed generalizing.

This finding suggests us when we generate paint data, sampling high-reward trajectories is much important than getting more trajectories.

Another finding is that as the size of the dataset increases, the performance of the Decision model would almost always increase or remain the same. However, In datasets collected by sub-optimal policy, the Behaviour Cloning messed up all things and got a worse result as the size grew up, which shows Behaviour Cloning has a higher standard for datasets.

5.3 Decision Transformer on Generated Paint Data

As mentioned in section 3.4, we used the pretrained renderer and agents from [2] to run the inference code on randomly sampled MNIST dataset. We obtained 3,000 trajectories with each trajectory/episode being 40 steps long and contain information including observations, rewards, actions, and terminals. Here is the detailed metadata for our generated dataset:

	Observation	Action	Reward	Terminal
Dimension	[7, 128, 128]	[65]	float	boolean
Description	[:3,128,128]: current state [3:7, 128, 128]: target [7, 128, 128]: rendering data	params for a stroke	dist(target-current) - dist(target-prev)	True if last step, else False

Table 3: Data Configurations

For training input, we feed our collected trajectories into the Decision Transformer[1], using the offical code template on Github(https://github.com/kzl/decision-transformer), each trajectory/episode being a 40-step sequence of [state, action, reward, terminal] transitions. In each of the timestep, as seen from the metadata, we feed the data with our current state along with the target. We did this as a reinforce step to make the Transformer model remember our goal. Our original proposal was that the Transformer model will be able to memorize and learn the patterns and steps needed to reach the target image.

Due to the computational limit, we only used 1000 trajectories from our collected dataset. The size of the each trajectory input is about 18MB, which amounts up to 54GB for 3,000 trajectories. With a single computational node, we were only able to maximally load 1,000 trajectories into the model. Therefore, we trained the the Decision Transformer with 2 configurations as follow:

DT Model	Config 1	Config 2
Time on GPU	43 hrs	50 hrs
Total Iter	10	8
Dataset	500 trajectories of MNIST	1000 trajectories of MNIST
Batch Size	64	64
Num Steps/Iter	10,000	10,000

Table 4: Training Configurations

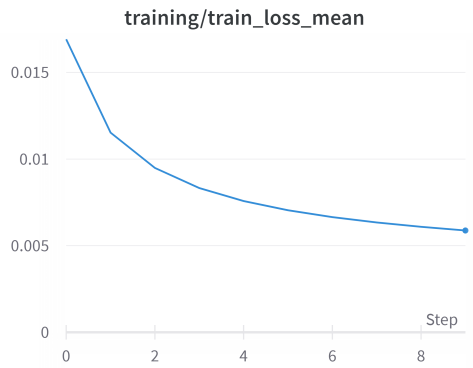
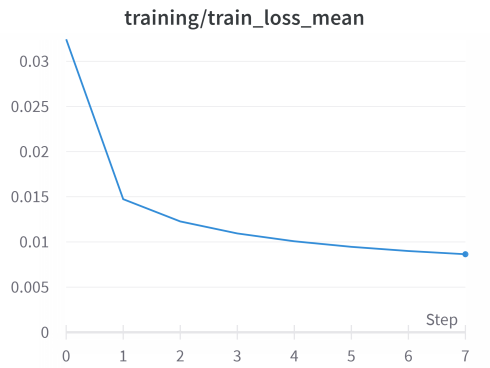
Config 1	Config 2
	

Table 5: Training Results

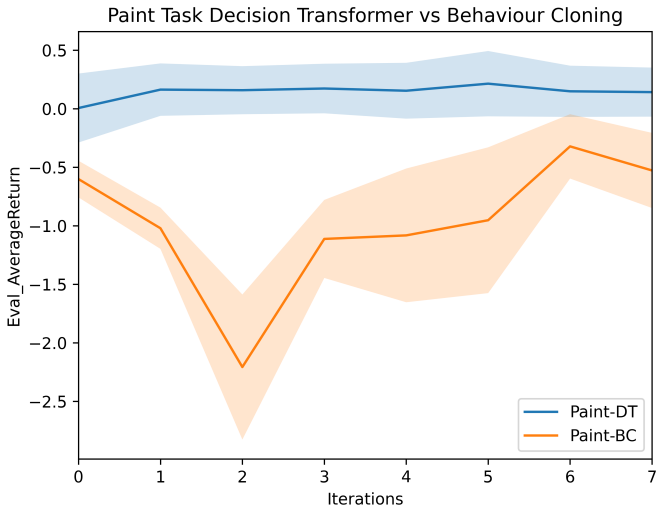


Figure 10: Comparison Between Decision Transformer and Behaviour Cloning

The training loss, defined as the squared distance between predicted actions and ground truth actions, as seen from plots 4 is decreasing. What's more, Decision Transformer has a better performance than the Behaviour Cloning as expected in Figure 10. This means that our algorithms are working as expected. However, looking at the qualitative report from the evaluations, the transformer model doesn't seem to be learning properly.

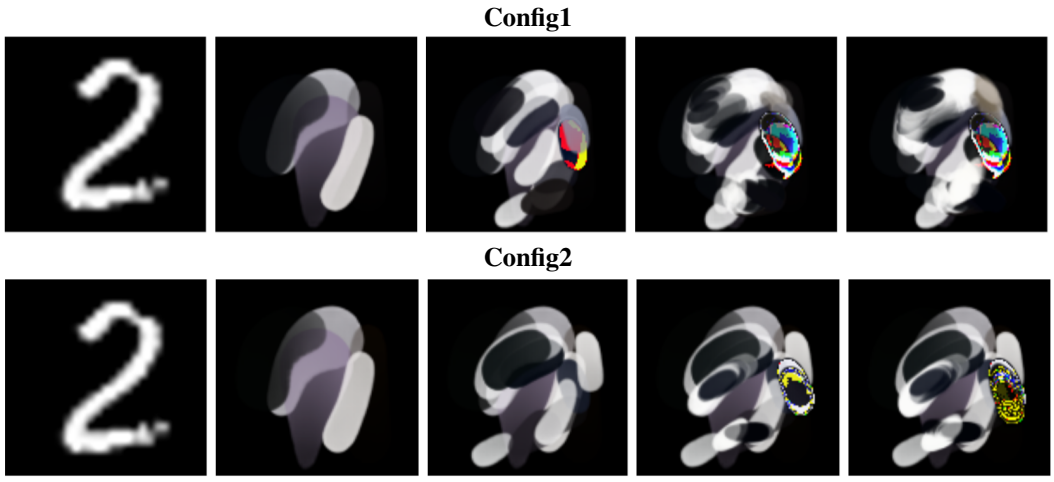


Figure 11: Inference results on writing a 2. The first picture is the target image.

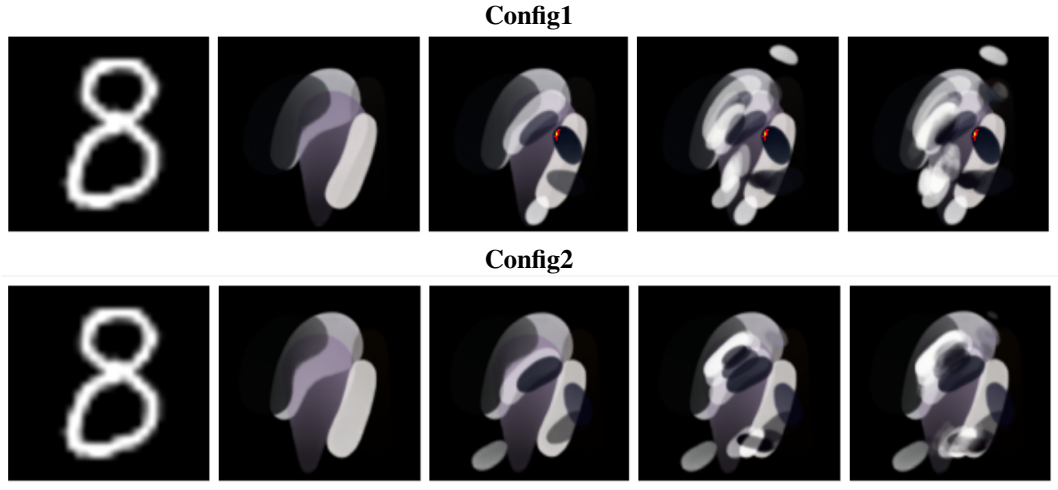


Figure 12: Inference results on writing an 8. The first picture is the target image.

In this experiment, we attempted to teach a computer to paint using a Decision Transformer model. However, despite our efforts, the model's performance was unsatisfactory, Figure 11 and Figure 12. The generated paintings were not of high quality and did not accurately capture the styles of the training data.

Our analysis suggests that this poor performance was likely due to a combination of factors, including the limited amount and quality of the training data, as well as the inherent difficulties in using a Decision Transformer model for this task. As for limited data, the training data was collected and produced by another agent, without enough time for validation, the quality of the data is not guaranteed. In particular, we looked into how the training data on writing the same digit. It turns out that in Figure 13, we found the same digit is actually written very differently whether it's in the size of each stroke or the sequence of the strokes. In this case, there are five similar targets '6', but the training data are completely different sequences of actions and observations. The model will likely to have difficulties learning the patterns of drawing a '6'.

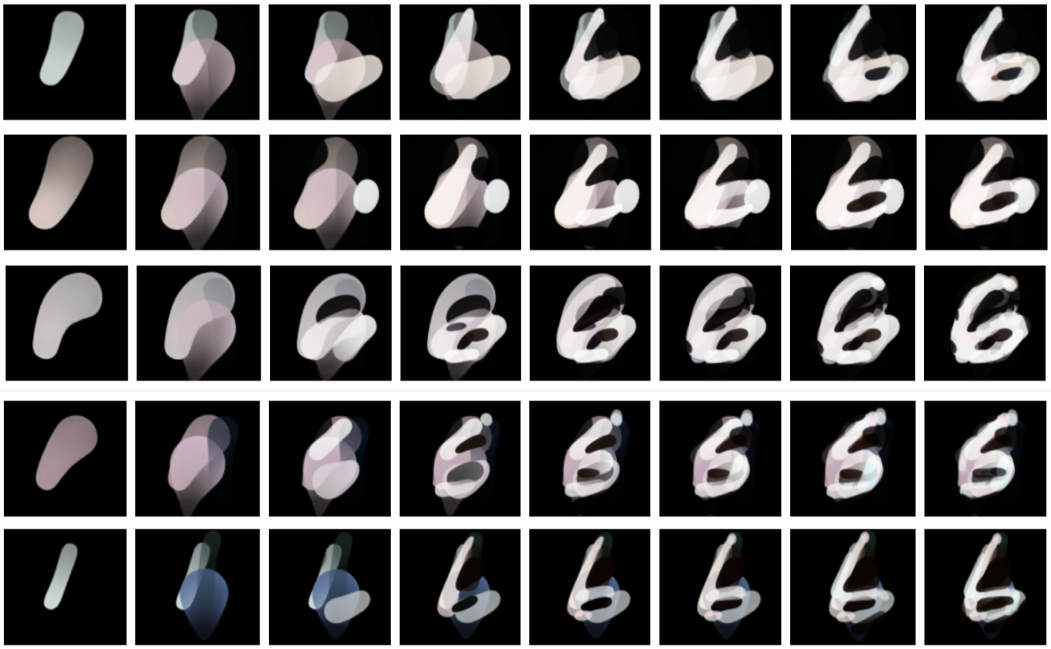


Figure 13: Training data that has '6' drawn in drastically different ways.

Meanwhile, since we are capturing loss and reward in pixel distances, the model doesn't have the ability to classify or recognize images. It will try to do anything to reduce the distance between the target image and the predicted strokes. This is not good since our observation space is huge given that it's 7 X 128 X 128. It will be difficult for the model to even memorize the unstructured patterns, let alone generalizing on unseen test data.

6 Discussion and Future Work

In this project, we explored the use of deep reinforcement learning to teach a computer to paint. We first reimplemented the model-free RL approach from "ICCV2019-Learning to Paint" and trained it on the MNIST dataset. We then tested the performance of the Decision Transformer on the NeoRL dataset and compared it to behavioral cloning. Our results showed that the Decision Transformer performed well on the NeoRL dataset, demonstrating its potential for solving real-world problems.

However, when we applied the Decision Transformer to the painting task, we encountered challenges. The size of the expert dataset was limited, and the computer's memory was not sufficient to train the model on the full dataset. Our analysis suggests that this poor performance was likely due to a combination of factors, including the limited amount and quality of the training data, as well as the inherent difficulties in using a Decision Transformer model for this task. In particular, we found that the training data on writing the same digit was written very differently, making it difficult for the model to learn the patterns of drawing a specific digit.

In the future, there are several experiments that could be done to improve the performance of the Decision Transformer on the painting task. One approach could be to explore different neural network architectures for the Decision Transformer. For example, using a convolutional neural network (CNN) could improve the model's ability to extract features from the input images and reduce the dimensionality of the input. This could make it easier for the model to learn the patterns of drawing specific digits or strokes.

Another potential approach for future work is to use transfer learning to improve the performance of the Decision Transformer on the painting task. Transfer learning involves using a pre-trained model on a related task as a starting point for training on a new task. In this case, the pre-trained model could be trained on a similar task, such as drawing strokes or characters, and then fine-tuned on the

painting task. This could help the model learn more efficiently and improve its performance on the painting task.

Finally, further research could be done to improve the quality and diversity of the training data for the painting task. In this study, we were limited by time and computational resources, which limited the amount of data we were able to collect and the complexity of the models we were able to train. In the future, additional time and resources could be dedicated to generating a larger and more diverse dataset, which could improve the performance of the Decision Transformer on the painting task.

7 Team contributions

Du Xiang: Model-free Paint Rimplementation training on MNIST, Data Collection from Pretrained Model (MNIST trajectories), NeoRL environment set up and Training using Decision Transformer, Paint experiments using Decision Transformer config 1, collaboration on report.

Zhiwei Zheng: Model-free Paint Rimplementation training on Celeb, Data Collection from Pretrained Model (Celeb trajectories), NeoRL environment set up and Training using Decision Transformer, Paint experiments using Decision Transformer config 2, collaboration on report.

References

[1] Lili Chen et al. "Decision Transformer: Reinforcement Learning via Sequence Modeling". In: *arXiv preprint arXiv:2106.01345* (2021).

[2] Zhewei Huang, Wen Heng, and Shuchang Zhou. "Learning to paint with model-based deep reinforcement learning". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2019.

[3] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[4] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[5] Rong-Jun Qin et al. "NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning". In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022. URL: https://openreview.net/forum?id=jNdLszxdtra.