

Learning to Doodle with Deep Q-Networks and Demonstrated Strokes

Tao Zhou¹

taozhou@cs.ucla.edu

Chen Fang²

cfang@adobe.com

Zhaowen Wang²

zhawang@adobe.com

Jimei Yang²

jimyang@adobe.com

Byungmoon Kim²

bmkim@adobe.com

Zhili Chen²

zlchen@adobe.com

Jonathan Brandt²

jbrandt@adobe.com

Demetri Terzopoulos¹

dt@cs.ucla.edu

¹ University of California, Los Angeles

Computer Science Department

Los Angeles, CA 90095, USA

² Adobe Research

345 Park Avenue,

San Jose, USA

Abstract

Doodling is a useful and common intelligent skill that people can learn and master. In this work, we propose a two-stage learning framework to teach a machine to doodle in a simulated painting environment via Stroke Demonstration and deep Q-learning (SDQ). The developed system, *Doodle-SDQ*, generates a sequence of pen actions to reproduce a reference drawing and mimics the behavior of human painters. In the first stage, it learns to draw simple strokes by imitating in supervised fashion from a set of stroke-action pairs collected from artist paintings. In the second stage, it is challenged to draw real and more complex doodles without ground truth actions; thus, it is trained with Q-learning. Our experiments confirm that (1) doodling can be learned without direct step-by-step action supervision and (2) pretraining with stroke demonstration via supervised learning is important to improve performance. We further show that *Doodle-SDQ* is effective at producing plausible drawings in different media types, including sketch and watercolor. A short video can be found at <https://www.youtube.com/watch?v=-5FVUQFQTaE>.

1 Introduction

Doodling is a common, simple, and useful activity for communication, education, and reasoning. It is sometimes very effective at capturing complex concepts and conveying complicated ideas [1]. Doodling is also quite popular as a simple form of creative art, compared to



Figure 1: Cat doodles rendered using color sketch (left) and water color (right) media types.

other types of fine art. We all learn, practice, and master the skill of doodling in one way or another. Therefore, for the purposes of building a computer-based doodling tool or enabling computers to create art, it is interesting and meaningful to study the problem of teaching a machine to doodle.

Recent progress in visual generative models—e.g., Generative Adversarial Networks [9] and Variational Autoencoders [18]—have enabled computer programs to synthesize complex visual patterns, such as natural images [9], videos [19], and visual arts [6]. By contrast to these efforts, which model pixel values, we model the relationship between pen actions and visual outcomes, and use that to generate doodles by acting in a painting environment. More concretely, given a reference doodle drawing, our task is to doodle in a painting environment so as to generate a drawing that resembles the reference. In order to facilitate the experiment setup and be more focused and expedient on algorithm design, we employ an internal Simulated Painting Environment (SPE) that supports major media types; for example, sketch and watercolor (Figure 1).

Our seemingly simple task faces at least *three challenges*:

First, our goal is to enable machines to doodle like humans. This means that rather than mechanically printing pixel by pixel like a printer, our system should be able to decompose a given drawing into strokes, assign them a drawing order, and reproduce the strokes with pen action sequences. These abilities require the system to visually parse the given drawing, understand the current status of the canvas, make and adjust drawing plans, and implement the plans by invoking correct actions in a painting environment. Rather than designing a rule-based or heuristic system that is likely to fail in corner cases, we propose a machine learning framework for teaching computers to accomplish these tasks.

The second challenge is the lack of data to train such a system. The success of modern machine learning heavily relies on the availability of large-scale labeled datasets. However, in our domain, it is expensive, if not impossible, to collect paintings and their corresponding action data (i.e., recordings of artists’ actions). This is compounded by the fact that the artistic paintings space features rich variations, including media types, brush settings, personal styles, etc., that are difficult to cover. Hence, the traditional paradigm of collecting ground truth data for model learning does not work in our case.

Consequently, we propose a hybrid learning framework that consists of two stages of training, which are driven by different learning mechanisms. In Stage 1, we collect stroke demonstration data, which comprises a picture of randomly placed strokes and its corresponding pen actions recorded from a painting device, and train a model to draw simple strokes in a supervised manner. Essentially, the model is trained to imitate human drawing behaviour at the stroke level with step-by-step supervision. Note that it is significantly easier to collect human action data at the stroke level than for the entire painting. In Stage 2, we challenge the model learned in Stage 1 with real and more complex doodles, for which there are no associated pen action data. To train the model, we adopt a Reinforcement Learning (RL) paradigm, more specifically Q-learning with reward for reproducing a given reference



Figure 2: Sketch Drawing Examples: **BMVC**. (top) The images produced by unrolling the Doodle-SDQ model for 100 steps. (bottom) The corresponding reference images.

drawing. We name our proposed system *Doodle-SDQ*, which stands for Doodle with Stroke Demonstration and deep Q-Networks. We experimentally show that both stages are required to achieve good performance.

Third, it is challenging to induce good painting behaviour with RL due to the large state/action space. At each step, the agent faces at least 200 different action choices, including the pen state, pen location, and color. The action space is larger than in other settings where RL has been applied successfully [19, 20, 21]. We empirically observe that Q-learning with a high probability of random exploration is not effective in our large action space, and reducing the chance of random exploration significantly helps stabilize the training process, thus improving the accumulated reward.

To summarize, Doodle-SDQ leverages demonstration data at the stroke level and generates a sequence of pen actions given only reference images. Our algorithm models the relationship between pen actions and visual outcomes and works in a relatively large action space. We apply our trained model to draw various concepts (e.g., characters and objects) in different media types (e.g., black and white sketch, color sketch, and watercolor). In Figure 2, our system has automatically sketched a colored “BMVC”.

2 Related Work

2.1 Imitation Learning and Deep Reinforcement Learning

Imitation learning techniques aim to mimic human behavior in a given task. An agent (a learning machine) is trained to perform a task from demonstrations by learning a mapping between observations and actions [15]. Naive imitation learning, however, is unable to help the agent recover from its mistakes, and the demonstrations usually cannot cover all the scenarios the agent will experience in the real world. To tackle this problem, DAGGER [22] iteratively produces new policies based on polling the expert policy outside its original state space. Therefore, DAGGER requires an expert to be available during training to provide additional feedback to the agent. When the demonstration data or the expert are unavailable, RL is a natural choice for an agent to learn from experience by exploring the world. Nevertheless, reward functions have to be designed based on a large number of hand-crafted features or rules [30].

The breakthrough of Deep RL (DRL) [23] came from the introduction of a target network to stabilize the training process and experience replay to learn from past experiences. Hasselt et al. [13] proposed *Double DQN* (DDQN) to solve an over-estimation issue in deep Q-learning due to the use of the maximum action value as an approximation to the maximum expected action value. Schaul et al. [23] developed the concept of *prioritized experience replay*, which replaced DQN’s uniform sampling strategy from the replay memory with a

sampling strategy weighted by TD errors. Our algorithm starts with Double DQN with prioritized experience replay (DDQN + PER) [23].

Recently, there has also been interest in combining imitation learning with the RL problem [8, 26]. Silver et al. [24] trained human demonstrations in supervised learning and used the supervised learner’s network to initialize RL’s policy network while Hester et al. [4] proposed Deep Q-learning from Demonstrations (DQfD), which leverages even very small amounts of demonstration data to accelerate learning dramatically.

2.2 Sketch and Art Generation

There are outstanding studies related to drawing in the fields of robotics and AI. Traditionally, a robot arm is programmed to sketch lines on a canvas to mimic a given digitized portrait [28]. Calligraphy skills can be acquired via Learning from Demonstration [27]. Recently, Deep Neural Network-based approaches for art generation have been developed [9, 8]. An earlier work by Gregor et al. [10] introduced a network combining an attention mechanism with a sequential auto-encoding framework that enables the iterative construction of complex images. The high-level idea is similar to ours; that is, updating only part of the canvas at each step. Their method, however, operates on the canvas matrix while ours generates pen actions that make changes to the canvas. More recently, a SPIRAL model [7] used Reinforced Adversarial Learning to produce impressive drawings without supervision; however, the model generates control points for quadratic Bezier curves, rather than directly controlling the pen’s drawing actions.

Rather than focusing on traditional pixel image modeling approaches, Zhang et al. [5] and Simhon and Dudek [25] proposed generative models for vector images. Graves [11] focused on handwriting generation with Recurrent Neural Networks to generate continuous data points. Following the handwriting generation work, a sketch-RNN model was proposed to generate sketches [12, 16], which was learned in a fully supervised manner. The features learned by the model were represented as a sequence of pen stroke positions. In our work, we process the sketch sequence data and, using an internal simulated painting environment, render onto the canvas as in the reference images.

3 Methodology

Given a reference image and a blank canvas for the first iteration, our Doodle-SDQ model predicts the pen’s action. When the pen moves to the next location, a new canvas state is produced. The model takes the new canvas state as the input, predicts the action based on the difference between the current canvas and the reference image, and repeats the process for a fixed number of steps. (Figure 3a).

3.1 Our Model

The network has two input streams (Figure 3b-A). The global stream has 4 channels, which comprise the current canvas, the reference image, the distance map and the color map. The distance map and the color map encode the pen’s position and state. The local stream has 2 channels—the cropped patch of the current canvas centered at the pen’s current location with size equal to the pen’s movement range, and the corresponding patch on the reference image. Unlike the classical DQN structure [20], which stacks four frames, the input in this model includes only the current frame and no history information.

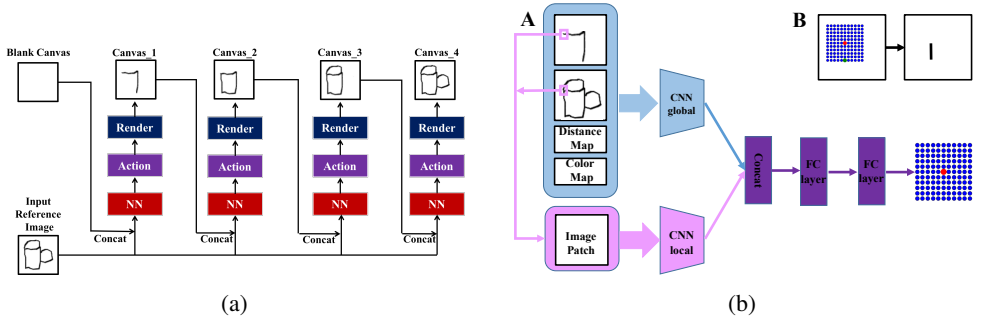


Figure 3: Doodle-SDQ structure. (a) The algorithm starts with a blank canvas and an input reference image. The neural network predicts the action of the pen and sends rendering commands to a painting engine. The new canvas and the reference image are then concatenated and the process is repeated for a fixed number of steps. (b) A: Two CNNs extract global scene-level contextual features and local image patch descriptors. The local and global features are concatenated for action prediction. B: Given the current position (red dot) and the predicted action (green dot), the painting engine renders a segment to connect them. The rectangle of blue dots represents the movement range, which is the same size as the local image patch.

The convnet for global feature extraction consists of three convolutional layers [20]. The first hidden layer convolves $32\ 8 \times 8$ filters with stride 4. The second hidden layer convolves $64\ 4 \times 4$ filters with stride 2. The third hidden layer convolves $64\ 3 \times 3$ filters with stride 1. The only convnet layer of the local CNN stream convolves $128\ 11 \times 11$ filters with stride 1. The two streams are then concatenated and input to a fully-connected linear layer, and the output layer is another fully-connected linear layer with a single output for each valid action.

At each time step, the pen must decide where to move (Figure 3b-B). The pen is designed to have maximum 5 pixels offset movement horizontally and vertically from its current position.¹ Therefore, the movement range is 11×11 and there are in total 121 positional choices.

The pen’s state is determined by the type of reference image. Specifically, the pen’s state is either up or down (i.e., draw) for a grayscale image. For a color image, the pen’s state can be up or down with a color selected from the three options (i.e., red, green, and blue).² Therefore, the dimension of the action space is 242 for grayscale images and 484 for color images. Figure 3b-B shows a segment rendered given the pen’s current position and the predicted action.

Rather than memorizing absolute coordinates of a pen on a canvas, humans tend to encode the relative positions between points. To represent the current location of the pen, an L2 distance map is constructed by computing

$$D(x, y) = \frac{\sqrt{(x - x_o)^2 + (y - y_o)^2}}{L}, \quad \forall (x, y) \in \Omega, \quad (1)$$

where Ω denotes the canvas which is an $L \times L$ discrete grid, L being the length of the canvas’ side, and (x_o, y_o) is the current pen location. In terms of a color map, all elements are 1 when the pen is put down and 0 when the pen is lifted up for grayscale images. For an image with red, green, and blue color, all elements are 0 when the pen is lifted up, 1 for red

¹The maximal offset movement of the pen is set arbitrarily; it could also be 4 or 6.

²The painting engine allows more colors; however, to simplify our experiments, we limit it to three colors.

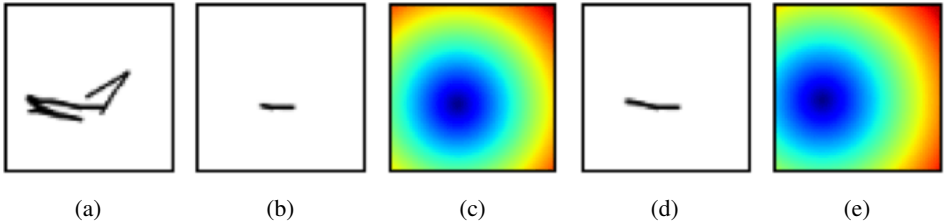


Figure 4: Data preparation for pre-training the network. (a) A reference image comprising two strokes randomly placed on the canvas; (b) the current canvas as part of the reference image; (c) the distance map of the current canvas, whose center is the pen’s location on the current canvas; (d) the next step canvas after a one step action of the pen; (e) The distance map of the next step canvas, which represents the pen’s location on the next step canvas.

color drawing, 2 for green color and 3 for blue color. The size of distance map and the color map is the same as the canvas size, which is 84×84 (Figure 3b-A). Table 1 summarizes the dimensionalities of the input and output for grayscale or color reference images.

Image	Input global stream	Input local stream	Output action space
Grayscale	$84 \times 84 \times 4$	$11 \times 11 \times 2$	$11 \times 11 \times 2 = 242$
RGB	$84 \times 84 \times 8$	$11 \times 11 \times 6$	$11 \times 11 \times 4 = 484$

Table 1: Input and output dimensionalities.

3.2 Pre-Training Networks Using Demonstration Strokes

DRL can be difficult to train from scratch. Therefore, we pre-train the network in a supervised manner using synthesized data with ground truth actions. The synthetic data are generated by randomly placing real strokes on canvas (Figure 4a). The real strokes are collected from recordings of a few artist paintings.

In the learning from demonstration phase, each training sample consists of the reference image (Figure 4a), the current canvas (Figure 4b), the color map, the distance map (Figure 4c), the small patch of the reference image, and the current canvas. The ground truth output will be the drawing action producing Figure 4d from Figure 4b. After training, the learned weights and biases are used to initialize the Doodle-SDQ network in the RL stage.

3.3 Doodle-SDQ

To encourage the agent to draw a picture similar to the reference image, the similarity between the k^{th} step canvas and the reference image is measured as

$$s_k = \frac{\sum_{i=1}^L \sum_{j=1}^L (P_{ij}^k - P_{ij}^{\text{ref}})^2}{L^2}, \quad (2)$$

where P_{ij}^k is the pixel value at position (i, j) in the k^{th} step canvas and P_{ij}^{ref} is the pixel value at that position in the reference image.

The pixel reward of executing action at the k^{th} step is defined as

$$r_{\text{pixel}} = s_k - s_{k+1}. \quad (3)$$

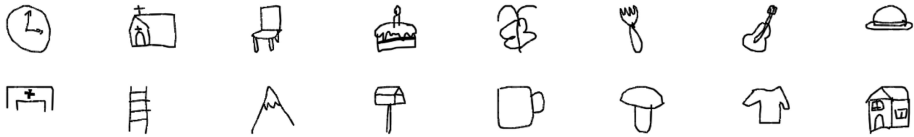


Figure 5: Reference images for training and testing. 16 classes are randomly chosen from the QuickDraw dataset [16]: clock, church, chair, cake, butterfly, fork, guitar, hat, hospital, ladder, mountain, mailbox, mug, mushroom, T-shirt, house.

An intuitive interpretation is that r_{pixel} is 0 when the pen is up and increases with the similarity between the canvas and reference image.

To avoid slow movement or pixel by pixel printing, we penalize small steps. Specifically, if the pen moves less than 5 pixels/step when the pen is drawing or if it moves while being up, the agent will be penalized with P_{step} . If the input is an RGB image, we additionally penalize the incorrectness of the chosen color P_{color} .

Thus, the final reward is

$$r_k = r_{\text{pixel}} + P_{\text{step}} + \beta P_{\text{color}}, \quad (4)$$

where P_{step} and P_{color} are constants set based on the observation, and β depends on the input image type: 0 for a grayscale image and 1 for a color image.

In the RL phase, we use QuickDraw [16], a dataset of vector drawings, as the input reference image. Since the scale of the drawings in QuickDraw varies across samples, the drawing sequence data is processed such that all the drawings can be squeezed onto an 84×84 pixel canvas. We randomly selected sixteen classes, and each class includes 200 reference images (Figure 5). For RL training, the images except for the ‘house’ class are applied. Therefore, 3,000 reference images are adopted for training.

4 Experiments

During the pretraining phase, we use a softmax cross entropy loss for the classification task. The loss is minimized using Adam [17] with minibatches of size 128 for optimization with the initial step size $\alpha = 0.001$, and gradually decays with the training step. Instead of using random initialization, the learned weights from the pretrained classification model are used to initialize Doodle-SDQ’s network. Due to the large action space, the pen is likely to draw a wrong stroke following a random action in the RL phase. Thus, exploration in action space is rarely applied unless the pen is stuck at some point.³ For the RL stage, we train for a total of 0.6M frames and use a replay memory of 20 thousand frames. The weights are updated based on the difference between the Q value and the output of the target Q network [23]. The loss is minimized using Adam with $\alpha = 0.001$. Our model is implemented in Tensorflow [4]. We plan to release our code, data, and the painting engine to facilitate the reproduction of our results.

To visualize the effect of the algorithm, the model is unrolled for 100 steps starting from an empty canvas. We chose 100 steps because more steps do not lead to further improvement. Figure 6 shows the drawing given the reference images from different categories in the test set using different media types. Additional sketch drawing examples are presented (Figure 7)

³From our observations, the pen is likely to stop moving at some location or move back and forth between two spots. Only in these scenarios, the pen will be given a random action to avoid local minima.

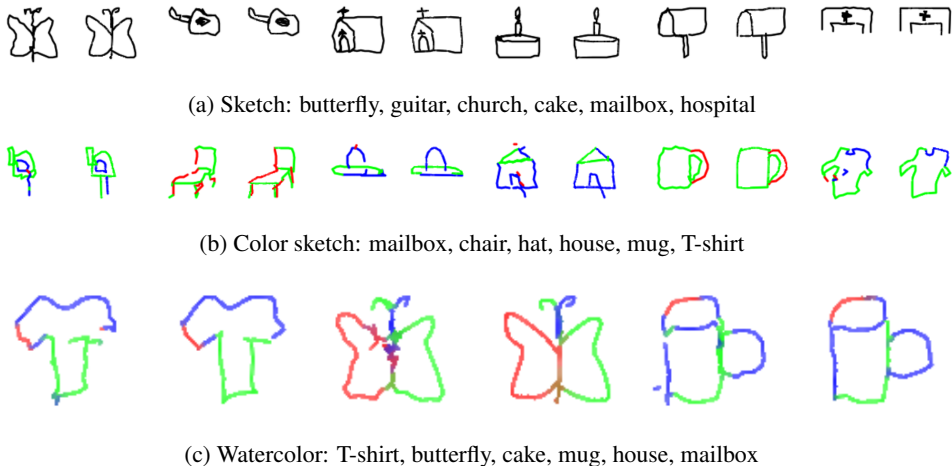


Figure 6: Comparisons between drawings and reference images in different media types: (a) sketch, (b) color sketch, (c) watercolor. The left image in each pair is the drawing after 100 steps of the model and the right is the reference image. The drawings in watercolor mode are enlarged to visualize the stroke distortion and color mixing

and the algorithm was tested on reference images not in the QuickDraw dataset, where we found that, although it was trained on QuickDraw, the agent has the ability to draw quite diverse doodles. For a reference image, the reward from each step is summed up and the accumulated reward is a quantitative measure of the performance of the algorithm. The maximum reward is achieved when the agent perfectly reproduces the reference image. In the test phase, we used 100 house reference images and 100 reference images randomly selected from the test sets belonging to the training classes.

		Naive SDQ	SDQ + Rare exp	Pretrain on random	Pretrain on QuickDraw	SDQ + Rare exp + weight init	Max reward
House Class	Sketch	93	1,404	1,726	1,738	1,927	2,966
	Color Sketch	-13	1,651	1,765	1,747	1,808	3,484
	Water Color	-162	407	596	620	670	1,492
Training Classes	Sketch	67	1024	1,539	1,521	1,805	2,645
	Color Sketch	-15	1,464	1,669	1,683	1,731	3,533
	Water Color	-182	363	446	473	509	1527

Table 2: Average accumulated rewards for the models tested.

Table 2 presents the average accumulated rewards and the average maximum rewards across reference images. In the table, the ‘Naive SDQ’ model is the Doodle-SDQ model trained from scratch following a ϵ -greedy strategy with ϵ annealed linearly from 1.0 to 0.1 over the first fifty thousand frames and fixed at 0.1 thereafter. The ‘SDQ + Rare exp’ is the Doodle-SDQ model trained from scratch with rare exploration. The ‘Pretrain on random’ model is the model with supervised pretraining on the synthesized random stroke sequence data (Figure 4). The ‘Pretrain on QuickDraw’ model is the model with supervised pretraining on the QuickDraw sequence data. The ‘SDQ + Rare exp + weight init’ model is the Doodle-

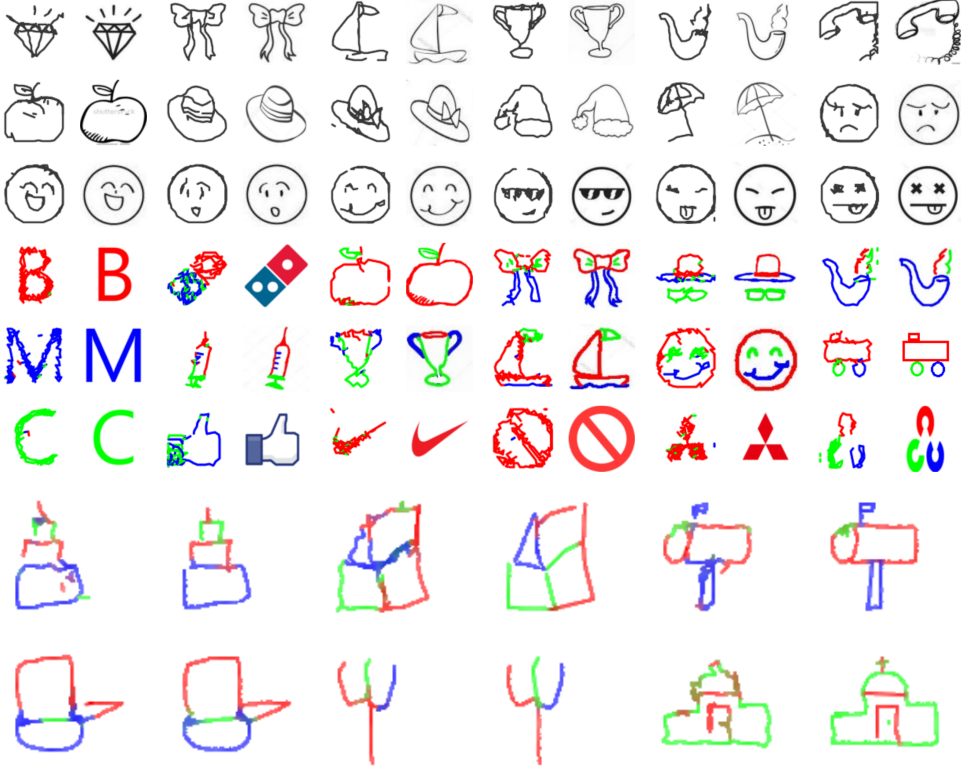


Figure 7: Additional sketch drawing examples.

SDQ model with rare exploration and weight initialization from the ‘Pretrain on random’ model. Based on the average accumulated reward, Doodle-SDQ with weight initialization is significantly better than all the other methods. Furthermore, pretraining on the QuickDraw sequence data directly does not lead to superior performance over the RL method. This indicates the advantage of using DRL in the drawing task.

5 Discussion

We now list several key factors that contribute to the success of our Doodle-SDQ algorithm and compare it to the DDQN + PER model of Schaul et al. [23] (Table 3).

Since Naive SDQ cannot be directly used for the drawing task, we first pretrain the network to initialize the weights. Referring to Table 2, pretraining with stroke demonstration via supervised learning leads to an improvement in performance (Columns 4 and 7). Based on our observations, the 4-frame history used in [23] introduces a movement momentum that compels the agent to move in a straight line and rarely turn. Therefore, history information is excluded in our current model. In [23], the probability for the exploration of random action decays from 0.9 to 0.1 with increasing epochs. Since we pretrained the network, the agent does not need to explore the environment with a large rate [4]. Thus, we initially set the exploration rate to 0.1. However, Doodle-SDQ cannot outperform the pretrained model until

we remove exploration.⁴ The countereffect of the exploration may be caused by the large action space. The small patch in the two streams structure (Figure 3) makes the agent attend to the region where the pen is located. More specifically, when the lifted pen is within one step action distance to the target drawing, the local stream is able to move the pen to a correct position and start drawing. Without this stream, the RL training cannot be successful even after removing the exploration or pretraining the network. The average accumulated rewards for the global stream only network varies around 100 depending on the media types.

	Doodle-SDQ	DDQN + PER [23]
History	No	Yes
Exploration	Rare	Yes
Pretrain	Yes	No
Input stream	2	1

Table 3: Differences between the proposed method and [23].

Despite the success of our SDQ model in simple sketch drawing, there are several limitations to be addressed in the future work. On the one hand, the motivation of this paper is to design an algorithm to enable machines to doodle like humans, rather than competing with GAN [9] to generate complex image types, at least not at the current stage. However, it has been demonstrated that an adversarial framework [10] interprets and generates images in the space of visual programs. Therefore, it will be a promising direction to mimic human drawing by combining adversarial training technique and reinforcement learning. On the other hand, although the SDQ model works in a relatively large action space due to rare exploration, the average accumulated rewards introduced by the component of reinforcement learning still suffers from the increase of the dimension of action space by allowing colorful drawing as shown by a comparison between sketch and color sketch (Column 6 and 7 in Table 2). Since our future work will incorporate more action variables (e.g., the pen’s pressure and additional colors) and explore doodling on large canvases, the actions might be embed in a continuous space [5].

6 Conclusion

In this paper we addressed the challenging problem of emulating human doodling. To solve this problem, we proposed a deep-reinforcement-learning-based method, Doodle-SDQ. Due to the large action space, Naive SDQ fails to draw appropriately. Thus, we designed a hybrid approach that combines supervised imitation learning and reinforcement learning. We train the agent in a supervised manner by providing demonstration strokes with ground truth actions. We then further trained the pre-trained agent with Q-learning using a reward based on the similarity between the current drawing and the reference image. Drawing step-by-step, our model reproduces reference images by comparing the similarity between the current drawing and the reference image. Our experimental results demonstrate that our model is robust and generalizes to classes not presented during training, and that it can be easily extended to other media types, such as watercolor.

⁴A random movement will be generated only when the agent gets stuck at some position, such as moving back and forth or remaining at the same spot.

References

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.
- [2] Sunni Brown. *The Doodle revolution: Unlock the power to think differently*. Penguin, 2014.
- [3] Gabriel V Cruz Jr, Yunshu Du, and Matthew E Taylor. Pre-training neural networks with human demonstrations for deep reinforcement learning. *arXiv preprint arXiv:1709.04083*, 2017.
- [4] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.
- [5] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.
- [6] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks, generating "art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*, 2017.
- [7] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. *arXiv preprint arXiv:1804.01118*, 2018.
- [8] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.
- [9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- [11] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.
- [12] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.
- [13] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI Conference on Artificial Intelligence*, pages 2094–2100. AAAI Press, 2016.

- [14] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, et al. Deep q-learning from demonstrations. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.
- [15] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):21, 2017.
- [16] Jongejan J, Rowley H, Kawashima T, Kim J, and Fox-Gieg N. The quick, draw! - ai experiment. <https://quickdraw.withgoogle.com>, 2016.
- [17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [19] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.
- [20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fiedjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Dan Wiestra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–33, 2015.
- [21] Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):81, 2016.
- [22] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.
- [23] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.
- [24] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [25] Saul Simhon and Gregory Dudek. Sketch interpretation and refinement using statistical models. In *Rendering Techniques*, pages 23–32, 2004.
- [26] Kaushik Subramanian, Charles L Isbell Jr, and Andrea L Thomaz. Exploration from demonstration for interactive reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 447–456. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

- [27] Yuandong Sun, Huihuan Qian, and Yangsheng Xu. Robot learns chinese calligraphy from demonstrations. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4408–4413. IEEE, 2014.
- [28] Patrick Tresset and Frederic Fol Leymarie. Portrait drawing by paul the robot. *Computers & Graphics*, 37(5):348–363, 2013.
- [29] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.
- [30] Ning Xie, Hirotaka Hachiya, and Masashi Sugiyama. Artist agent: a reinforcement learning approach to automatic stroke generation in oriental ink painting. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1059–1066. Omnipress, 2012.
- [31] Xu-Yao Zhang, Fei Yin, Yan-Ming Zhang, Cheng-Lin Liu, and Yoshua Bengio. Drawing and recognizing chinese characters with recurrent neural network. *IEEE transactions on pattern analysis and machine intelligence*, 2017.