Research Article

Feng Zhao*

Deep reinforcement learning enhances artistic creativity: The case study of program art students integrating computer deep learning

https://doi.org/10.1515/jisys-2023-0292
received December 02, 2023; accepted March 01, 2024

Abstract: During the artistic journey, creators frequently encounter challenges stemming from pressure, resource constraints, and waning inspiration, all of which can impede their creative flow. Addressing these obstacles requires a multifaceted strategy aimed at nurturing creativity throughout the artistic process. Procedural art generation emerges as a viable solution to invigorate artistic creativity. In this study, the deep Q-network (DQN) was constructed to solve the shortage of artistic creativity through its automatic decision-making ability. The model was trained with different types of artistic styles (abstract and minimalism) in WikiArt dataset. The model generates various artistic elements of different styles, forms, or thinking according to the input parameters or constraints, and selects specific colors, textures, or shapes to help the artist maintain focus in the creation process and expand the creativity in the creation process. In order to achieve this goal, in the process of performing the procedural art generation task with DQN, the experiment collected the generation speed, interpretability, and creativity evaluation feedback of each style of art. The feedback results show that the scores of color field painting and minimalism were 83.2, 93.5, 86.3 and 86.6, 91.5, 82.1 respectively. The research shows that employing dynamic mass spectrometry networks enables the automation of the art creation process. This innovative approach facilitates the exploration of diverse creative ideas tailored to various artistic tasks, thereby fostering advancements in art creation and nurturing creativity.

Keywords: program art generation, deep Q-network, deep reinforcement learning, artistic creation, boost creativity

1 Introduction

In the current field of art, artists face various challenges, including the repetition of creativity, limitations in expressive forms, and a lack of fresh inspiration. These challenges limit the diversity and depth of artistic creation, often leading to works of art falling into traditional categories and repetitive imitations of existing styles. In addition, artists often need to invest a lot of time and energy in seeking inspiration and experimenting with creativity in the pursuit of originality and novelty. In this context, deep learning, especially deep reinforcement learning (DRL), provides new possibilities for artistic creation.

The application of DRL technology in artistic creation has opened a new door for artists. Using DRL, artists can generate unique and complex artworks through computer models, thereby breaking through the

* **Corresponding author: Feng Zhao**, School of Art and Design, Changzhou Institute of Technology, Changzhou, 213022, China, e-mail: zhaof@czu.cn

Open Access. © 2024 the author(s), published by De Gruyter. This work is licensed under the Creative Commons Attribution 4.0 International License.

limitations of traditional creativity. Technologies such as deep Q-network (DQN) can automatically generate creative works of art by analyzing large amounts of data and learning complex patterns. This can not only help artists overcome the problem of creative exhaustion, but also inspire them to explore new forms of art and techniques of expression. In addition, the application of this technology can also help artists better understand and analyze the trends of artistic creation, so as to keep up with the times while maintaining innovation.

The lack of creativity in artistic creation is a long-standing challenge that limits artists' freedom of expression and potential for innovation in the creative process. Ishiguro and Okada [1] focused on what works of art promote the individual's inspiration for creation (inspiring works of art) and proposed a theoretical framework to explain the types of creative outcomes consistent with this dual concern. Aiming at the problem that artistic practice has become the main focus of research activities, Skains [2] proposed a methodology that takes creative practice as research. In response to the rapid development of computer technology, Al Hashimi et al. [3] explored the convergence of creativity, technology, and art and design education, and advocated the use of digital tools and repurposing of social media applications to support creative thinking. Gillam [4] and other scholars explored how participation in creative arts activities can enhance public mental health and well-being, with a particular focus on the use of music and creative writing. He combined clinical experience to explore people's understanding of the health and well-being benefits of creative arts activities. Taylor and Kaufman [5] and other scholars showed that the hierarchical value structure of creative individuals was systematically different from that of uncreative individuals in previous studies, and discussed the impact of these results on the specificity and motivation of creative domains. Kim [6] argued that the interdisciplinary practice of art courses can enhance students' abilities by fostering creativity and provide a foundation for assessment methods in contemporary art education. Anderson et al. [7] used grounded theory methods to explore how creative engagement is formed in early adolescent learners. He explores the perspectives and experiences of students participating in an integrated learning model for the arts during secondary school. The above scholars put forward their own views on the problem of insufficient creativity in artistic creation. In the creative process, the emergence of creative inspiration is often unpredictable and sometimes difficult to sustain, and the limitations of thinking may cause the creator to fall into a fixed thinking in the familiar field, and it is difficult to break through the traditional creative framework. In this context, this study introduces DRL to provide artists with a new way to deal with creative challenges.

The application of DRL in enhancing artistic creativity provides a new way to break through the limitations of traditional artistic creation. The traditional process of artistic creation often relies on the artist's intuition, experience, and personal inspiration. However, this dependence makes artists susceptible to limitations from personal experience and existing knowledge, making it difficult to explore new forms of artistic expression or innovative thinking. As an advanced machine learning technology, DRL provides artists with a new creative tool through its unique learning mechanism – learning and optimizing behavioral strategies through interaction with the environment. Through DRL, artists can transform their creative ideas into algorithmic parameters, enabling computers to generate novel and diverse works of art through exploration and experimentation. The application of DRL in artistic creation not only enables artists to break through the limitations of their personal thinking, but also stimulates new creative inspiration. By utilizing DRL generated artworks, artists can gain new perspectives and creativity, thereby incorporating these new elements into their own works. At the same time, the ability of DRL technology to process large amounts of data and identify complex patterns allows artists to explore unprecedented artistic styles and techniques, further enriching the diversity of artistic expression.

Deep Q-network is a reinforcement learning algorithm used to solve discrete action space problems, applied to procedural art generation can also solve some need to generate discrete, symbolic output problems. The application of deep learning in artistic creation is gradually demonstrating its unique value and potential. This technology is not only used for the generation of images and visual arts, but also extended to music, dance, literature, and other art forms. For example, deep learning models can analyze and learn from the works of famous artists in history, and then create new works with similar styles. This method has been used to imitate the art styles of artists such as Van Gogh or Picasso, as well as generate new music works with unique styles. In the fields of dance and performing arts, deep learning models are used to capture and analyze

the movements of dancers, and then create new dance movements and choreography based on these data. This not only provides new tools for choreographers, but also offers new analytical methods for dance research. In addition, deep learning has also been applied to literary creation, such as automatically generating poetry or stories. These systems can mimic specific writing styles or create new storylines by learning a large amount of textual data. Lin et al. and Li et al. [8,9] conducted research on DQN. The former proposed an edge-based intelligent manufacturing plant framework that extends DQN to solve the decision problem of multiple edge devices, while the latter developed an airborne DQN to reduce the packet loss rate of the overall data of sensing devices. Bo et al. [10] and other scholars discussed the application of computational aesthetics in aesthetic measurement, quantification, and generative art, using computers to judge beauty and ugliness and automatically generate aesthetic images. Cetinic and She [11] discussed the various practices of artificial intelligence (AI) art and the role of AI in artistic creation, integrating related works that deal in detail with these topics. DiPaola et al. [12] and other scholars showed how the concepts of honing theory, intrinsic motivation, and "seed events" can be implemented computationally, demonstrating their impact on the art of generating results. They discussed how the exploration of deep learning convolutional neural network generative systems can help understand human creativity. From the research given by the above scholars, it can be seen that the use of computer to create inspiration has a significant effect on enhancing artistic creativity.

In the creative process, artists confront challenges beyond technology and craftsmanship; they must also tap into their inspiration, embrace creative freedom, and explore their innovative potential. The intervention of DQN, as a new way of creation, can well solve the problem of insufficient creativity in artistic creation. In this study, the construction and experimental design of DQN followed a series of rigorous steps to ensure that the model can be effectively applied to program art generation. The structural design of DQN is customized for the special needs of programming art. This network includes multiple convolutional layers and fully connected layers, enabling the network to process and analyze complex visual inputs. The input layer of the network receives the original artwork data, while the output layer generates new artwork. The connections between these layers are achieved through a series of nonlinear transformations to simulate complex decisions in the process of artistic creation. The main hypothesis of the study is that DQNs can effectively simulate and enhance human creativity in program art generation. The core objective of the research is to design and optimize a special Q-network structure that can process and produce works of art with aesthetic value. The expected outcome is to achieve a model that can automatically generate high-quality artworks under given artistic styles and conditions. By testing the model under different parameter settings, the aim of the study is to find the most suitable parameter combination for artistic creation, so as to achieve the best performance of the model in the task of program art generation. Ultimately, the model should demonstrate a high degree of flexibility and adaptability, capable of generating works with creative and aesthetic value under different artistic styles and conditions. A special Q network structure is designed for the procedural art generation task to achieve the input and output of subsequent datasets. For the advantages and disadvantages of a model, a series of parameters such as learning rate, greedy strategy value, experience playback buffer, discount factor and maximum number of training rounds are set to test, and the best parameters are selected which are most consistent with the program art generation. Finally, when the model is applied to the procedural art generation task, the average reward of the model reaches up to 3,275, and the overall reward score hovers between 2,800 and 3,300. This is close to the reward obtained in parameter optimization, indicating that the model has a good effect on this task. Finally, the generated images were scored, and the scores reached over 80 points, which was at a high level, in terms of generation speed, interpretability measure, and creativity evaluation. This study demonstrates the innovation and potential of the integration of computer science and art by applying DQNs to program art generation. The innovation of the research lies in the successful application of DRL technology to the process of artistic creation. This interdisciplinary attempt not only expands the application scope of deep learning in different fields, but also provides a new methodology for artistic creation. By imitating and learning from existing art styles, DQN can automatically generate innovative and aesthetically valuable works of art, thus breaking through the limitations of traditional artistic creation.

2 DQN structure design

As a reinforcement learning algorithm [13,14], the core idea of DQN is to use deep neural network to estimate approximate functions, so as to realize the learning of optimal strategies in complex environments [15]. The main goal of DQN is to learn an optimal action value function (Q function) in the Markov Decision Process so that the agent can choose the best action to maximize the cumulative reward. By combining deep learning techniques with traditional art, DQNs can learn and imitate complex patterns and styles in traditional art, and then create new works of art based on this foundation. Models can not only replicate existing art styles, but also discover new patterns and combinations during the learning process, thus creating unprecedented works of art. By analyzing the characteristics of classical paintings, DQN can generate new paintings with similar styles but containing original elements.

2.1 Neural network architecture

DQN combines deep learning with reinforcement learning to effectively learn and optimize strategies in complex tasks [16,17]. The following are the main elements of DQN's neural network architecture [18,19].

2.1.1 Input layer

The input layer of the network takes the environment condition information as input. In the process of graphics, the graphical input is a multi-dimensional vector in pixels, because a picture has thousands of pixels and channels. Therefore, the input layer is not only the input of a picture, but also embodies the "depth" property of deep learning. The dimension of this input vector depends on the size of the image and the number of color channels. In a 200 × 200 pixel color image, the input layer of the image has 200 × 200 × 3 nodes. Each node corresponds to a pixel value or characteristic of an image.

$$X_{i,j,c}. \tag{1}$$

Each element is mathematically represented as $X_{i,j,c}$, where i and j represent row and column positions in the image, and c represents color channels. This matrix or tensor would be passed to the next layer, usually the convolution layer, for feature extraction.

The input layer is responsible for receiving, preprocessing, and representing the observed data of the environment in the procedural art generation task, providing the data for the neural network to start learning, and its core goal is to convert the raw data into a form acceptable to the neural network.

2.1.2 Convolutional layers

Convolutional layers are used to extract features from input data. In the convolution layer, the convolution kernel (also known as the filter) is used to perform convolution operations on the input data by sliding over it. The size of the convolution kernel is usually a square matrix of size 3 × 3 or 5 × 5. The number of convolution kernels is a hyperparameter that determines the number of features learned by the convolution layer [20,21]. After calculating the weighted sum of the local area, the output feature map is generated. This process network learns the features of different locations regardless of their exact location in the input.

Each convolution kernel performs convolution operations on the input data to obtain a feature map. Feature map is a new feature map obtained after processing input data. One convolution kernel can perform edge detection in the image, while the other convolution kernel can perform texture detection in the image. Because there are multiple convolution cores in the convolution layer, multiple feature maps are generated, and each feature map corresponds to a feature.

Convolution is an operation between the convolution kernel and the input data. The algorithm slides the convolution kernel over the input data, multiplies its weight with the corresponding part, and adds it to get the feature map. The mathematical representation of the convolution operation is as follows [22]:

$$Z(i, j) = (X \times W)(i, j) + b, \tag{2}$$

where Z is the generated feature map, X is the input data, W is the weight matrix of the convolution kernel, and b is the bias term. The convolution operation is performed once for each position of the input data, sliding the convolution kernel W over the input X. It can calculate the dot product for each position and add a bias term b, which generates a pixel value in the output feature map. Thus, the entire feature map is generated.

The weight of the convolution kernel is the parameter that needs to be learned. These weights determine how the convolution kernel slides over the input to extract features. Through optimization methods such as backpropagation algorithms and gradient descent, the network automatically adjusts the weight of the convolutional kernel to minimize the loss function, allowing the convolutional layer to extract features useful to the task. The initial weights of the convolution kernel are initialized according to the random distribution, in order to ensure that the initial values of the weights do not have task-related bias, so as to help the model learn the features of the task better.

The weight initialization method used in this study is Xavier/Glorot initialization [23,24]. The purpose of using Xavier/Glorot initialization is to keep the variance of input and output of each layer as equal as possible, so as to ensure that the gradient remains stable in the forward and back propagation. In a fully connected layer (or convolution layer), where the number of input channels is n_{in}, the number of output channels is n_{out}, and assuming the weights are initialized to W, then the weights W initialized by Xavier/Glorot can be expressed as follows:

$$W \sim \text{Uniform}\left(-\sqrt{\frac{6}{(n_{\text{in}} + n_{\text{out}}) \cdot k_{\text{h}} k_{\text{w}}}}, \sqrt{\frac{6}{(n_{\text{in}} + n_{\text{out}}) \cdot k_{\text{h}} k_{\text{w}}}}\right), \tag{3}$$

where k_{h} and k_{w} indicate that the convolution kernel size is $k_{\text{h}} \times k_{\text{w}}$, the above is the uniformly distributed Xavier/Glorot initialization formula, and the weights are finally initialized to a uniform distribution. The normal distribution initialization formula is as follows:

$$W \sim \text{Normal}\left(0, \sqrt{\frac{2}{(n_{\text{in}} + n_{\text{out}}) \cdot k_{\text{h}} k_{\text{w}}}}\right). \tag{4}$$

The weights of the above formula are initialized to a normal distribution with a mean of 0 and a standard deviation of $\sqrt{\frac{2}{(n_{\text{in}} + n_{\text{out}}) \cdot k_{\text{h}} k_{\text{w}}}}$. Uniformly distributed Xavier/Glorot initialization is suitable for cases where an S-type (Sigmoid) activation function is used. When the input values of two activity functions approach 0, the slope is large. Xavier's initialization scope just happens to set the weight on a slope, so that the activity function still has a slope when the input value is close to 0. The Xavier/Glorot initialization of a normal distribution is appropriate when an unrestricted excitation function, such as rectified linear unit (ReLU), is used. When the input value is greater than 0, the slope of ReLU is very large, so the normal distribution initial value method can better meet the conditions of ReLU excitation function and extend the application range of weighted initial value.

The two Xavier/Glorot initialization formulas depend on the startup function used later. It is helpful to improve the gradient in the learning process and improve the learning effect of the network. In the convolution layer, the experiment chooses to use the initialization mode of uniform distribution, and the activation function uses Sigmoid activation function [25]. The Sigmoid activation function maps any real value x to the interval (0, 1), with the shape of an S-shaped curve, as follows:

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{5}$$

where $f(x)$ is the output of the activation function.

After the weights are initialized, the updated weights predict the model through forward propagation, calculate the losses, and then calculate the gradients using back propagation. Finally, the model can be gradually optimized by weight updating to make it perform better in the task.

2.1.3 Fully connected layers

Each neuron in the fully connected layer has a connection with each neuron in the previous layer, thus forming a dense structure of connections. Among them, the function of the fully connected layer is to linearly merge and nonlinearly transform the features of the upper layer, so as to obtain the feature expression of the upper layer and the feature expression of the next layer.

The input value of this fully connected layer is the input value of the previous layer (convolution or other fully connected layer), expressed as a vector. The dimension of this input vector depends on the number of neurons in the previous layer. The output of the fully connected layer is also a vector, and each unit in the vector corresponds to a single neuron in that layer.

Similar to the convolutional layer, each link in the fully connected layer has a corresponding weight that is used for linear synthesis of the output of the previous layer. Specifically, for the *i*th neuron, its input can be expressed as [26] follows:

$$Z_i = \sum_{j=1}^{n} (W_{ij} \cdot X_j) + b_i, \tag{6}$$

where Z_i is the input of the *i*th neuron, W_{ij} is the weight that connects the *j*th input neuron to the ith output neuron, X_j is the output of the *j*th neuron in the previous layer, and b_i is the bias.

Similar to the convolution layer, the normal distribution-based Xavier/Glorot initialization method is used for the weights of the fully connected layer. The ReLU function is used to activate the function, the specific formula is as follows:

$$Y_i = \mathrm{ReLU}(Z_i) = \max(0, Z_i), \tag{7}$$

where Y_i represents the output of the *i*th fully connected layer, the max function keeps the part of the input value *x* greater than zero unchanged, and the part less than or equal to zero becomes zero. This means that if the input *x* is greater than 0, the output of the excitation function is equal to the input, and if the input is less than or equal to 0, the output of the excitation function is 0.

In the whole connected layer, each neuron obtains the output signal of this neuron by calculating the input signal and applying an incentive function. This process would be repeated with each neuron of a complete connecting layer. The output of the fully connected layer is used as input to the next layer and then transmitted to the next layer, and in the last few layers, the fully connected layer is often used to produce the final network output.

2.1.4 Output layer

The output layer is the lowest layer of the neural network, and its function is to generate a final prediction or evaluation [27]. In DQN, the neurons in the output layer represent various behaviors or strategies, and their behavioral value function (Q) is called DQN. The construction of its output layer is closely related to the task.

The neurons in the output layer of DQN estimate the behavior value function $Q(a)$ for each behavior a. The behavioral value function $Q(a)$ represents the expected return of performing behavior a under given current conditions. This is a core idea of DQN, whereby an intelligent agent can choose from multiple actions and only choose those that have the greatest value.

In the output layer, the Q value of each neuron is calculated as follows:

$$Q(a) = f(WX + b), \tag{8}$$

where $Q(a)$ is the estimated Q value for action a. W is the weight matrix, which contains the connection weights of each neuron to the previous layer. X is the output of the fully connected layer, that is, the output of the previous layer. b is the offset term. f is the activation function. In the output layer, since the Q value can be any real number, the excitation function is generally not used, or the linear excitation function is used. Finally, every possible action is calculated and Q value is obtained. Q value is an important basis for the decision of the agent.

2.2 Input and output

In the procedural art generation task, the experiment uses DQN as a key component to guide the art generation process. The following is a detailed description of the input and output of DQN in this study [28].

2.2.1 Input

The input of DQN model is related to the environment of art creation and the goal of art creation. Specifically, the input includes the following elements:

Environmental state [29]: in procedural art generation, the state of the environment represents the current state or progress of the creation. It can be a picture, a component of a painting, or an illustration of the current stage of creation. In this process, the system would transmit environmental information as data to the DQN and express it in the form of a tensor.

Generative goal: generative goal refers to a vision produced by the artist in the process of creation, which includes the expectation of the art work, the style of the art work, and the expectation of the art theme. In procedural art creation, the object of creation can be a specific artistic effect, or it can be the aesthetic needs that the artist wants to achieve. Translating the resulting target into a format that the model understands is a big challenge and often requires special coding and presentation.

2.2.2 Output

The output of DQN is a strategy, that is, the work is created by the model according to the input environmental conditions and the purpose of production. In procedural art generation, the generation strategy can include the following:

Painting movements indicate which elements to add, modify, or delete in the artwork, such as patterns, lines, colors, etc.

Style adjustment describes how to change a painting's style, color, texture, etc., to meet generative goals.

Composition suggestions guide the way the artist arranges elements on the canvas to achieve the desired layout and structure.

Through training, the DQN model learns the generation strategy from the environmental state and the generation goal [30] to meet the artist's ideas to the greatest extent. This approach allows the artist to interact with the model, adjust and improve the artwork according to the model's suggestions, while retaining the artist's creativity and intuition.

2.3 Reinforcement learning parameters

Compared with ordinary neural networks, DQN introduces a series of special parameters and techniques, such as experience playback, goal network, ε-greedy strategy, and Q-Learning goal. The introduction of these

parameters and techniques is mainly due to the particularity of Q network [31]. In order to optimize Q network training so that it can stably estimate Q value functions, thereby improving the performance of reinforcement learning tasks, and helping DQN to perform well when dealing with reinforcement learning problems in discrete action spaces, the following DRL parameters need to be considered in detail in procedural art generation tasks.

2.3.1 Discount factor

The discount factor (usually expressed as γ) weighs current and future returns. Higher γ values value long-term gains, while lower γ values value short-term gains. In procedural art generation, the choice of discount factors would have a direct impact on the characteristics of the created artwork, for example, whether it focuses more on general creativity or improvisation.

2.3.2 Experience replay

Empirical playback is a training technique that first randomly samples existing experimental data and then trains them to improve the robustness and convergence of the model [32,33]. In procedural art generation, experiential playback can help models learn and improve generation strategies better. In the case of maximum use of historical experience, it reduces data correlation and enhances stability.

2.3.3 Target network update

DQN uses a set of backbone networks and a set of target networks to enhance the stability of the learning process. The weights of the destination network are not updated frequently, but periodically copy the weights of the primary network. This can reduce the deviation of the object in the training process and improve the performance of the model. In program art generation, in order to ensure the production of high-quality program art, the target network needs to be updated.

2.3.4 Q-learning objectives

The DQN network uses Q learning algorithm to update Q function, and the purpose of Q learning is to estimate Q function based on Bellman equation [34]. Specifically, Q-learning goals (often expressed as TD goals) are as follows:

$$Q(s, a) = R(s, a) + \gamma \times \max(Q(s', a')), \tag{9}$$

where $Q(s, a)$ is the Q value of action a performed in state s, $R(s, a)$ is the instant reward obtained after performing action a in state s, γ is a discount factor that balances the importance of immediate and future rewards, $\max(Q(s', a'))$ represents the maximum Q value for selecting the optimal action a' in the next state s'.

The goal of Q-learning is to gradually approach the Q-value in the process of Q-learning to the best Q-function, so as to guide individuals to make optimal decisions. In procedural art generation, the determination of Q learning object would directly affect the learning and improvement of production strategy, so that the project has higher creativity and artistry.

2.3.5 ε-greedy strategy

ε-greedy strategy is a way to balance development and use. In DQN, the agent selects actions according to the ε-greedy strategy to conduct behavior selection [35]. ε is the likelihood of discovery and $1 - \varepsilon$ is the likelihood of using a known strategy.

When the random variable is less than ε, the agent would randomly select a behavior in order to explore various states and behaviors in the environment. When the random variable is greater than or equal to ε, the agent chooses the behavior with the largest Q value according to the existing optimization strategy.

The choice of ε-greedy strategy needs to be based on the characteristics of the task. The higher the ε, the task is more inclined to explore, and the lower the ε, the task is more inclined to use it. In the process of programming and generation, the reasonable selection of ε directly affects the diversity and stability of the generation strategy.

2.4 Network training

2.4.1 Loss function

By evaluating the model at the cost of mean squared error (MSE), DQN minimizes the deviation between the evaluation results of the model and the target results. The loss function is as follows:

$$L(\theta) = E[(Q(s, a; \theta) - (r + \gamma \times \max(Q(s', a'; \theta_{\text{target}})))^2], \tag{10}$$

where θ is the parameter of the current network. θ_{target} is the parameter of the target network. $Q(s, a; \theta)$ is the predicted action value of the model taking action a under state s; r is the immediate reward returned by the environment. γ is the discount factor. $\max(Q(s', a'; \theta_{\text{target}}))$ is the maximum Q value of all possible actions in the next state s' given by the target network.

2.4.2 Optimization algorithm

DQN is a method that uses Stochastic gradient descent (SGD) and its variants to minimize the loss function in the network. The purpose of this method is to use backpropagation technology to modify the parameters in the network, so that the loss function is gradually reduced, and the behavior value function is gradually close to the actual value function. SGD related calculation formula is as follows:

$$\theta = \theta - \alpha \nabla \theta L(\theta), \tag{11}$$

where θ represents the parameter vector that needs to be updated, and α (learning rate) is a hyperparameter that controls the length of the update step and determines how much the parameter changes at each update step. $\nabla \theta L(\theta)$ represents the gradient (derivative) of the loss function $L(\theta)$ with respect to the parameter θ.

2.4.3 Training strategy

The DQN training process includes the following key strategies:

Experience replay: In order to achieve the robustness of training and the effective use of samples, the experience-based replay method is often used to store the existing state, behavior, reward, and the next state and other information, and then randomly extract it. This method effectively reduces the relationship between samples and avoids the model falling into local minima.

Update of target network: In order to enhance the stability of learning, two kinds of networks are generally used: one is the existing Q-network (the Q-network described here), and the other is the target Q-network. On this basis, the existing Q-network is copied periodically to obtain its parameters, reduce the volatility of the model, and achieve the purpose of stabilization.

Exploration strategy: In the initial stage, the ε-greedy strategy is used to randomly select the behavior, and the probability of $1 - \varepsilon$ is selected for the current optimal behavior. As training progresses, the value of ε usually decreases so that the model relies more on what has been learned.

3 DQN construction generated by program art

DQNs learn how to execute tasks to maximize rewards through interaction with the environment. In the application of artistic creation, DQN first understands different artistic styles and elements by observing the data (images or music) of the artwork. These data are considered as environmental states, and the goal of DQN is to generate new works of art by taking specific "actions." During the training process, DQN learns what kind of actions would produce high-quality artwork by trying different actions and observing the results. For example, when the image generated by DQN matches a certain artistic style, it can receive positive feedback (reward). DQN uses these rewards to update its internal decision-making strategies, thereby generating works that better align with the target style in the future. The key technical elements include experience replay and target network. Experience replay allows DQN to learn from past attempts, breaking the correlation between data and improving learning efficiency by randomly extracting previous experiences. The target network is used to stabilize the learning process. It is a copy of the current network, regularly updated, and used to calculate the expected reward value, so that the algorithm remains stable during the training process.

3.1 Dataset preparation

The preparation of the dataset is a worthy part of the experiment when performing the task of procedural art generation. High-quality datasets can affect the performance of DQNs and the quality of the generated artwork. In this study, WikiArt, an open source dataset, is chosen for the experiment. The dataset is a large dataset of paintings, which contains a large number of art works, covering various periods, styles, and artists. A total of 195 paintings by artists are included, including 42,129 for training and 10,628 for testing, for a total of 52,757. The works are divided into eight categories, including landscapes, figures, still life, and abstractions. It is a collection of data that is well worth exploring for art lovers, researchers, or practitioners. This experiment would focus on the experiment of two types of artistic data: color field painting and minimalism.

The selection of the WikiArt dataset is crucial for the success of this study, as it provides rich and diverse art samples for DQN. This dataset is chosen because it has several key characteristics that make it an ideal choice for programming art generation tasks. First, WikiArt contains a wide range of art works from different historical periods, which means it covers a variety of art styles from classical to modern. This diversity provides DQN with rich learning resources, enabling the network to learn and imitate various different artistic styles. In addition, the WikiArt dataset offers a wide variety of art works, ranging from landscape paintings, figure paintings, still life paintings to more abstract art forms such as abstract and minimalist art. These different types make the dataset not only visually diverse but also equally rich in concepts and presentation techniques. Especially, the art styles of color block painting and minimalism, due to their unique visual features and expressive techniques, provide a challenge for the DQN model and an opportunity for researchers to evaluate the model's ability to handle different art styles. In order to ensure the quality and representativeness of the dataset, strict standards are adopted in the selection process of WikiArt. The selected artwork not only needs to have sufficient visual resolution and clarity, but also needs to ensure diversity and balance between samples. This means that the number of works in each category needs to be representative enough to avoid bias in the learning process of the model. Finally, the historical depth and cultural breadth of the WikiArt dataset provide a comprehensive artistic worldview for the DQN model. By learning and analyzing the works in this dataset, DQN can not only imitate existing artistic styles, but also create new forms of artistic expression based on this foundation.

Although WikiArt covers a wide range of painting art styles for the purposes of this article, its format is not suitable as a data input for DQN. Therefore, translating the WikiArt dataset into a data format suitable for DQN model inputs became an urgent problem. The experiment needs to conduct data preprocessing and feature extraction on the collected dataset to match the experiment format and model.

Since DQN combines Q-learning, a reinforcement learning algorithm, the model takes the environmental state as the input and the action as the output. It is sensitive to the scale and distribution of the input, so it is necessary to process the environmental status and scale of the original data first.

(1) Data cleaning: In the collected data, it needs to detect and remove low-quality images or abnormal data in the dataset. Data with damage, incomplete information, large image contrast differences, and obvious edge defects need to be cleaned. It can detect whether there are duplicate art works in the dataset, ensuring the diversity of the data. On this basis, this article proposes a new method for image classification of art works.
(2) Normalization of image attributes: Standardization requires all images to have the same format, such as Joint Photographic Experts Group or Portable Network Graphics. This can ensure consistent identification of image data by subsequent models. It can adjust the size and resolution of images to maintain consistency. This method not only reduces the computational complexity of the system, but also ensures the consistent dimensionality of the input data of the system. The color, brightness, contrast, and other parameters of an image can be standardized to have the same visual characteristics during training. On this basis, the analysis of relevant metadata (author, style, creation time, etc.) can be used to achieve accurate and consistent metadata, providing a basis for subsequent data annotation and reward functions.
(3) Feature extraction: Next it is necessary to extract features related to states, actions, rewards, and the next state from each artwork.

State: It can use the current picture to represent a state. Finally, the convolutional neural network can be used to extract the features of the image and carry out feature extraction. This eigenvector is the state expression of DQN.

Action: In procedural art generation, action is represented as editing or transforming an existing art image. These techniques include color adjustment, style change, shape adjustment, blur effect, and line enhancement. Each action can be assigned a unique identifier and mapped to a form that the model can understand. In the experiment, for the adjustment of color, style change, shape adjustment, blur effect, and line enhancement of the five editing operations, this study uses 0–5 integers for each operation.

Reward: The design of the reward function determines how the model evaluates the generated artwork. Rewards are defined based on factors such as the quality of the generated work, its closeness to the target style, and its innovation. If the generated work closely matches the target style, a higher reward can be allocated.

Next state: The next state is the Art Nouveau image after the action is performed. Based on the editing of the current state by the action, a new image can be generated and used as the next state.

3.2 Parameter tuning

When using DQN for training, this article adjusts the following hyperparameters to optimize the performance of the algorithm: Learning rate (α), ε greedy strategy (ε), experience playback buffer size (buffer size), discount factor (γ), and maximum training epoch (Epoch). The experiment tested these five parameters separately, and the specific parameters are shown in Table 1:

Among the above parameters, the learning rate α determines the step size at which the model updates weights in each iteration. Greedy strategy ε is a balance between exploring new ideas and utilizing known effective strategies. The size of the experience playback buffer is used to store and reuse previous experiences; the discount factor γ is used to balance the importance of current rewards with future rewards. Maximum number of training rounds: the maximum number of steps or rounds specified for the training algorithm to run is an important hyperparameter.

3.3 Training process

The training process of DQN in program art generation:

Table 1: DQN parameter table

Sequence	α	ε	Buffer size	γ	Epoch
1	0.01	0.01	10,000	0.95	500
2	0.005	0.01	10,000	0.95	500
3	0.001	0.01	10,000	0.95	500
4	0.005	0.01	10,000	0.95	500
5	0.005	0.03	10,000	0.95	500
6	0.005	0.05	10,000	0.95	500
7	0.005	0.05	10,000	0.95	500
8	0.005	0.05	20,000	0.95	500
9	0.005	0.05	30,000	0.95	500
10	0.005	0.05	20,000	0.99	500
11	0.005	0.05	20,000	0.89	500
12	0.005	0.05	20,000	0.69	500
13	0.005	0.05	20,000	0.69	500
14	0.005	0.05	20,000	0.69	500
15	0.005	0.05	20,000	0.69	500
16	0.005	0.05	20,000	0.69	500
17	0.005	0.05	20,000	0.69	1,500
18	0.005	0.05	20,000	0.69	3,000

(1) Initialize the network and experience replay buffer: Before training, the weights and optimizers of the DQN neural network can be initialized. It needs to create an experience replay buffer to store samples of previous states, actions, rewards, and the next state for training purposes.
(2) Environmental interaction: During each training cycle (epoch), DQN interacts with the program art generation environment. The model selects an action from the current environmental state, executes the action, and observes the reward returned by the environment and the next state. This process is repeated until the predetermined number of time steps are reached or the generated artwork is considered complete.
(3) Experience replay: At the end of each time step, the status, behavior, rewards, and data for the next stage are saved in an experience replay cache. Its main purpose is to disrupt the interrelationships between data and enhance the stability of training. In each batch of training, a random batch would be selected from an experienced playback buffer for training.
(4) Calculate Q value and target value: Existing neural networks can be used to calculate Q value for each behavior in each state. These Q values are calculated based on the weighting of the current network. At the same time, the target Q value is calculated based on the weight of the target network (usually a copy of the current network, regularly updated). After completing various actions, the target Q value is used to evaluate long-term benefits.
(5) Calculate loss function: By calculating the target Q value and combining it with the actual Q value of the existing network, the corresponding loss function is obtained. The experiment used MSE to measure the difference between the predicted values of the model and the actual indicator values.
(6) Optimize the network and update the target network: Backpropagation algorithm can be used and combined with an optimizer to update the weights in the network to minimize the loss function in the network. The image generation algorithm can be optimized and the weights of the target network can be periodically updated. Under a given number of training steps, the weights of the target network can be copied, which can improve the stability and efficiency of training.
(7) Repeat training cycle: The above steps can be repeated until the predetermined number of training cycles is reached or the stop condition (performance convergence) is met. The number of training cycles is usually a hyperparameter that needs to be adjusted based on the complexity of the task and computational resources.

In this study, the reasons for selecting specific hyperparameters and their impact on model performance are the core components of experimental design. Choosing the appropriate learning rate α is crucial as it determines the speed at which the model absorbs new information in each iteration. An excessively high learning rate may cause the model to fail to converge during the training process, while an excessively low learning rate may slow down the training progress. In the experiment, different learning rates are tested to find the optimal balance point, enabling the model to learn effectively without losing valuable information too quickly. The selection of the greedy strategy value ε is crucial for balancing the exploration and utilization of the model. A higher ε value encourages the model to explore more, while a lower value makes the model more inclined to utilize known information. Determining the appropriate ε value can help the model better find a balance between the two, thereby achieving optimal performance during the learning process. The size of the experience replay buffer is also a key factor. A larger buffer can store more experience, providing a richer learning material for the model. However, an excessively large buffer may cause the model to use outdated information during training, affecting learning efficiency. The setting of the discount factor γ affects the model's emphasis on future rewards. A higher γ value makes the model more focused on long-term rewards, while a lower value makes the model more inclined to pursue immediate rewards. The adjustment of this parameter helps to control the behavior strategy of the model, making it more suitable for specific artistic creation tasks.

Next experimental testing would be conducted on the various parameters selected above, with different interval ranges set for each parameter. The purpose is to select the most suitable set of parameters for the experiment. First, the learning rate parameter was analyzed and three different settings were set, namely, 0.01, 0.005, and 0.001. The other parameters are temporarily initialized using the established parameters, with ε set to 0.01, Buffer size set to 10,000, γ set to 0.95, and epoch selected as 500 for the experiment.

Next α was tested, with four parameters set: 0.5, 0.1, 0.05, and 0.01. The variance of performance uncertainty under different parameters, exploration-exploitation balance, and convergence speed were collected. The higher the value of variance, the greater the volatility of model performance. The convergence speed refers to the speed at which a model achieves optimal or near optimal performance during the training process. Exploring utilizing balance is a strategy in which algorithms find a balance between exploring new actions (finding potentially better strategies) and utilizing known actions (proven effective strategies). If the value of the exploration-exploitation balance is 0.3, it means that the algorithm has a 30% probability of choosing to explore new actions, and a 70% probability of choosing to utilize known and proven actions.

This article would analyze buffer size. The experience replay buffer is used to store experience samples of previously observed states, actions, rewards, and the next state of the agent. The experiment selected 10,000, 20,000, and 30,000 as parameter choices, as shown in Table 2.

Table 2: Impact of different buffer sizes

Buffer size	Stability score	Memory usage	Training time
10,000	63.3	4.6	73.6
20,000	89.3	8.3	93.5
30,000	93.1	10.6	133.1

In Table 2, it can be seen that with the continuous increase in buffer size, there has been varying degrees of growth in stability, memory usage, and training time.

From the perspective of precision rate, the precision rate with a buffer of 20,000 is 0.927, which is higher than the precision rate of the other two parameters. The discount factor (γ) determines the importance of future rewards. A larger discount factor places greater emphasis on long-term rewards, while a smaller discount factor places greater emphasis on immediate rewards. Based on different γ values, the model was tested for reward, learning speed, stability standard deviation, and exploration-exploitation balance, as shown in Table 3.

Table 3: Discount factor impact table

γ	Reward	Learning speed	Stable standard deviation	Exploration-exploitation balance
0.99	1,365	96	0.34	0.58
0.89	2,963	172	0.21	0.35
0.69	2,763	192	0.18	0.30

From Table 3, it can be seen that among the three different γ values, the comprehensive performance of the 0.69 interval is higher than the other two. From the perspective of obtaining rewards, the reward with an γ value of 0.69 is 2,763, which is much higher than the γ value of 0.99 and not significantly different from the γ value of 0.89. From the perspective of learning speed, the γ value of 0.69 is 192, the γ value of 0.89 is 172, and the γ value of 0.99 is 96. It can be seen that the γ value of 0.69 is higher than the other two; from the perspective of stability standard deviation, the stability standard deviation of γ value 0.69 is 0.18, the stability standard deviation of γ value 0.89 is 0.21, the stability standard deviation of γ value 0.99 is 0.34, and the stability standard deviation of γ value 0.69 is leading in the experiment. Exploration-exploitation balance also presents the same effect: the γ value of 0.69 is 0.30, the γ value of 0.89 is 0.35, and the γ value of 0.99 is 0.58. The experiment finally determined that the γ value is 0.69.

Epoch defines the number of times the entire training dataset is fully transmitted to the neural network model in the neural network model, and sets appropriate parameter ranges to achieve effective training and excellent performance of the model. The initial training frequency given in the experiment is 500 to avoid wasting resources and time caused by early training. After determining other complex parameters, this parameter can be adjusted and two different epochs can be set for experimentation.

During the process of Epoch from 500 to 1,500 and then to 3,000, both the training and validation sets exhibited similar curves in terms of loss value and accuracy. It can be clearly seen that with the increase in epoch, the trend of these two performance indicators gradually flattens out. Finally, the final epoch of 1,500 was selected for the experiment.

Based on the above tests for each parameter, the optimal parameters were determined in the experiment as shown in Table 4.

Table 4: Experimental determination parameters table

α	0.005
ε	0.05
Buffer size	20,000
γ	0.69
Epoch	1,500

4 Case experiment: Program art generation using DQNs

4.1 Experimental results

In this study, the trained DQN model not only demonstrated its effectiveness in color block painting and minimalist styles but was also applied to various other art styles to demonstrate its adaptability and versatility. In the experiment, the model was used to generate impressionist style artworks. After inputting classic works of impressionism, the model can output images with typical impressionist features, such as blurred contours, vivid light and shadow effects, and vivid color applications. These works reflect the unique handling of light and color by impressionists. Similarly, the model was also tested for its effectiveness in surrealist styles. Surrealist works are known for their dreamy scenes and illogical images. The model generated a series of

imaginative and creative works in this style of testing, showcasing the unique dreamlike and irrational expression of surrealism. In addition, in the testing of abstract art, the model successfully created a series of abstract works that showcase the non-concrete aesthetics of abstract art through the free combination of color, shape, and line. This indicates that the DQN model can understand and mimic the core concepts of abstract art, i.e., conveying emotions and concepts through visual language.

This article would test the tested model and input different styles of painting works into the model. Through learning and testing the model, the task of program art generation can be achieved, as shown in Figure 1.

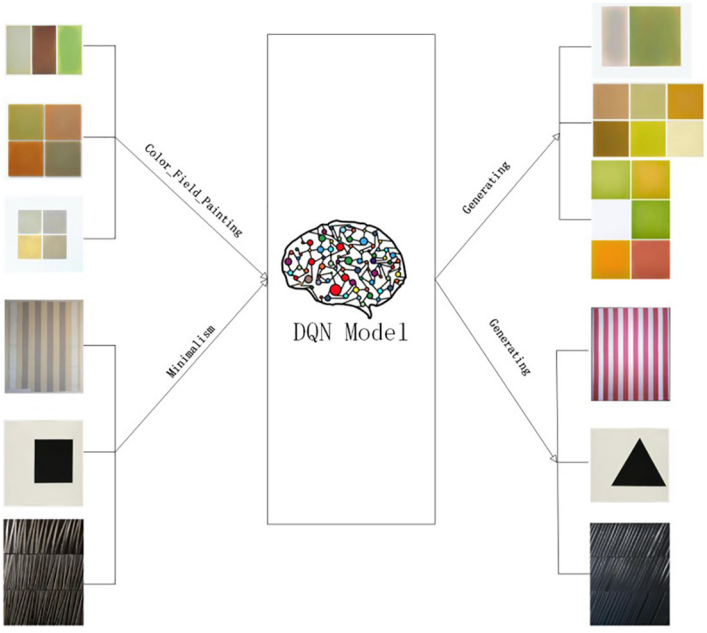

Figure 1: Generation of two styles.

In Figure 1, the left side shows the target image of the input model, and the right side shows the output image of the model. It can be seen that for the model trained in the experiment, after inputting art images of a specific style, the model would output results of a specific style.

The experiment also collected five DQN performance indicators for each of the two styles, and recorded the average reward for each experiment, which is the average reward obtained by the model in the environment. Convergence indicates whether DQN has successfully converged to a stable strategy. If it converges, it is marked as “convergent,” otherwise it is marked as “unstable.” The exploration rate records the probability of the model choosing random actions in each experiment, which gradually decreases with training; Q-value convergence indicates whether the Q-value has successfully converged during the training process. If the Q value converges, it can be marked as “convergent”; stability is used to evaluate the stability and reproducibility of DQN performance. If the performance is stable, it can be marked as “stable,” otherwise it is marked as “unstable.” DQN was applied in the process of artistic creation, and various methods were utilized to improve and innovate existing models. More complex and customized network architectures were used to better capture and simulate the subtle features of specific artistic styles. For art styles with complex textures and rich details, deeper network structures were designed and specially designed levels and activation functions were added to improve the model’s learning and generation capabilities on these details. Considering the impact of the exploration rate of the model on performance, the learning process was optimized by dynamically adjusting the exploration strategy. An adaptive exploration rate adjustment mechanism based on performance indicators was implemented to enable the model to conduct extensive exploration in the initial

stage, and then gradually reduce exploration based on learning progress and performance improvement, in order to improve learning efficiency and stability.

In Table 5, experiment numbers 1–5 represent the evaluation results during the process of drawing color field painting styles. Experiment numbers 6–10 show the evaluation results during the process of drawing minimalist styles. From the entire table, it can be seen that the overall average reward remains between 2,800 and 3,300, with a minimum exploration rate of 0.01.

Table 5: Model performance evaluation indicators table

Number	Average reward	Astringency	Exploration rate	*Q* value convergence	Stability
1	3,053	Convergent	0.1	Convergent	Stable
2	2,963	Convergent	0.05	Convergent	Stable
3	3,275	Unstable	0.2	Convergent	Unstable
4	3,096	Convergent	0.02	Convergent	Stable
5	3,152	Unstable	0.21	Convergent	Unstable
6	3,245	Unstable	0.15	Convergent	Unstable
7	3,026	Convergent	0.01	Convergent	Stable
8	2,998	Convergent	0.03	Convergent	Stable
9	2,865	Unstable	0.2	Convergent	Unstable
10	2,986	Convergent	0.02	Convergent	Stable

At the end of the experiment, three scoring items were selected: generation speed, interpretability measurement evaluation, and creativity analysis to obtain user satisfaction and feedback on the generated work. Among them, the generation speed measures the efficiency of the model in generating artistic works. This indicator is crucial for practical applications, especially in scenarios where a large number of works need to be generated quickly. An efficient model can produce high-quality works in a short period of time, which is an important measure for practicality and commercial applications. Interpretability measurement evaluation focuses on whether the artwork generated by the model can be understood and appreciated by human audiences. This indicator reflects the model's ability to capture and imitate subtle differences and complexity in human artistic creation. An artwork generated by a highly interpretable model is more comparable to the works of human artists and is more easily accepted and appreciated by the audience. Creativity analysis measures whether a model can produce novel, unique, and creative works of art. This indicator is particularly important as it directly relates to the originality and innovation of artistic works. A highly creative model can break through the limitations of traditional thinking and style, generating works of art with innovative elements and unique styles. The experiment collected the evaluation scores of 100 viewers, with a maximum score of 100 points. The average values of these three items are shown in Figure 2.

In Figure 2, the horizontal axis coordinates represent two different styles of art works (one is color field painting and the other is minimalism). From Figure 2, it can be seen that the generation speed, interpretability, and creativity scores of color field paintings are 83.2, 93.5, and 86.3, respectively. The generation speed, interpretability, and creativity scores of minimalism are 86.6, 91.5, and 82.1, respectively.

4.2 Discussion

From Table 5, it can be seen that the average reward of the experiment is around 3,000 under both stable and convergent conditions. However, it can be observed that the average reward fluctuates significantly under unstable convergence and unstable stability conditions. The minimum reward is 2,865, and the maximum reward is 3,275. At the same time, the exploration rate is higher than other groups, with a fluctuation of around 0.2.

In the ratings shown in Figure 1, the ratings for color field painting and minimalism are shown separately. From the perspective of generation speed, it can be seen that the former has a slightly lower score of 83.2 than

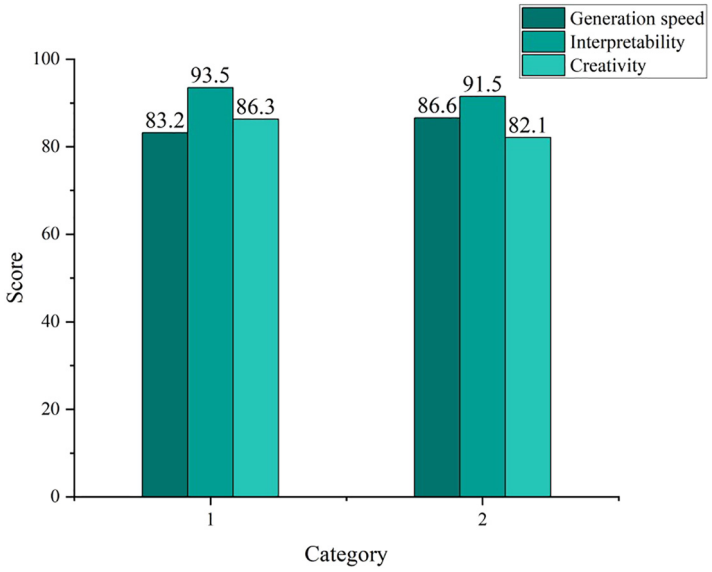

Figure 2: Program art generation rating.

the latter, but in terms of interpretability and creativity, the score of color field painting is higher than that of minimalism. In terms of explanatory indicators and innovation ability, due to the manually drawn dataset, there are limitations in novelty and innovation ability.

From the data in Table 5 and the ratings in Figure 2, it can be observed that there are differences in the performance of models with different artistic styles, which are mainly influenced by various factors. Color block painting and minimalist style have significant differences in visual characteristics. Color block paintings usually contain more details and color variations, which may provide DQN with richer learning materials, thereby making the model perform better in this style. In contrast, the minimalist style is characterized by simplicity and limited color usage, which may result in models facing fewer changes during the learning process, and may explain its relatively low ratings in interpretability and creativity. The different exploration rates also affect the performance of the model. A higher exploration rate means that the model attempts more new actions, leading to fluctuations in performance in specific tasks. In experiments, unstable convergence and stability are often accompanied by higher exploration rates, which leads to fluctuations in the average reward.

In this study, the advantage of the DQN model lies in its ability to learn and adapt to the characteristics of different art styles, generating works of art with high aesthetic value. By adjusting key parameters, the model can flexibly transition between different styles, demonstrating good adaptability and creativity. However, the model also has limitations. For example, high exploration rates may lead to unstable performance, while the model's dependence on the dataset means that its creativity and novelty are limited by the provided training data. The aim of this article is to improve the model's adaptability and creativity to different art styles by adopting more advanced network architectures and learning algorithms. Second, by enhancing and diversifying the training dataset, the learning range of the model can be expanded, improving its ability to generate novel and creative works. Finally, the exploration strategy of the model is further optimized to reduce performance fluctuations and improve the overall stability and reliability of the model while ensuring learning efficiency. Through these improvements, the application effect of the model in program art generation has been significantly improved.

5 Conclusion

This study successfully explored the application of DQN in program art generation, demonstrating the enormous potential of deep learning technology in improving artistic creativity and promoting automation in

artistic creation. By constructing and applying DQN based models, the experiment not only achieved significant improvements in the quality of artistic works, but also demonstrated enhanced innovation and diversity, indicating that deep learning techniques can effectively imitate and expand human artistic creativity. Key findings include the effectiveness of the DQN model in capturing and imitating different artistic styles, as well as its potential in generating works with high aesthetic value. These achievements are of great significance for understanding the application of AI in the field of art, and provide new paths for the integration of art and technology. Although this study has achieved positive results, there is still potential for further optimization of the model and expansion of its application scope. Future research can focus on optimizing hyperparameters in models to enhance the diversity of generated works, which would expand the understanding of DRL in art creation and promote deeper integration of art and technology.

Funding information: This work was supported by the Industry, University, and Research Project Construction of the "Qiyun" Online Art Course Teaching Platform at Art Museum Qiyun (Horizontal project number: E4-6112-22-447).

Author contributions: F.Z. was responsible for the manuscript writing, research framework design, mode creation, data analysis, proofreading language, and processing images. The author has read and agreed to the published version of the manuscript.

Conflict of interest: The author declares that there is no conflict of interest regarding the publication of this article.

Data availability statement: The data used to support the findings of this study are available from the corresponding author upon request.

References

[1] Ishiguro C, Okada T. How does art viewing inspires creativity? J Creative Behav. 2021;55(2):489–500. doi: 10.1002/jocb.469.
[2] Skains RL. Creative practice as research: discourse on methodology. Media Pract Educ. 2018;19(1):82–97. doi: 10.1080/14682753.2017.1362175.
[3] Al Hashimi S, Al Muwali A, Zaki Y, Mahdi N. The effectiveness of social media and multimedia-based pedagogy in enhancing creativity among art, design, and digital media students. Int J Emerg Technol Learn (iJET). 2019;14(21):176–90. doi: 10.3991/ijet.v14i21.10596.
[4] Gillam T. Enhancing public mental health and wellbeing through creative arts participation. J Public Ment Health. 2018;17(4):148–56. doi: 10.1108/JPMH-09-2018-0065.
[5] Taylor CL, Kaufman JC. Values across creative domains. J Creative Behav. 2021;55(2):501–16. doi: 10.1002/jocb.470.
[6] Kim H. An analysis of creative effect on interdisciplinary practices in art education. Int J Educ Art. 2018;14(2):179–96. doi: 10.1386/eta.14.2.179_1.
[7] Anderson RC, Haney M, Pitts C, Porter L, Bousselot T. "Mistakes can be beautiful": Creative engagement in arts integration for early adolescent learners. J Creative Behav. 2020;54(3):662–75. doi: 10.1002/jocb.401.
[8] Lin CC, Deng DJ, Chih YL, Chiu HT. Smart manufacturing scheduling with edge computing using multiclass deep Q network. IEEE Trans Ind Inform. 2019;15(7):4276–84. doi: 10.1109/TII.2019.2908210.
[9] Li K, Ni W, Tovar E, Jamalipour A. On-board deep Q-network for UAV-assisted online power transfer and data collection. IEEE Trans Veh Technol. 2019;68(12):12215–26. doi: 10.1109/TVT.2019.2945037.
[10] Bo Y, Yu J, Zhang K. Computational aesthetics and applications. Vis Comput Ind Biomed Art. 2018;1(1):1–19. doi: 10.1186/s42492-018-0006-1.
[11] Cetinic E, She J. Understanding and creating art with AI: Review and outlook. ACM Trans Multimed Comput Commun Appl. 2022;18(2):1–22. doi: 10.1145/3475799.
[12] DiPaola S, Gabora L, McCaig G. Informing artificial intelligence generative techniques using cognitive theories of human creativity. Procedia Comput Sci. 2018;145(2):158–68. doi: 10.1016/j.procs.2018.11.024.
[13] François-Lavet V, Henderson P, Islam R, Bellemare MG, Pineau J. An introduction to deep reinforcement learning. Found Trends® Mach Learn. 2018;11(3–4):219–354. doi: 10.1561/2200000071.

[14] Wang HN, Liu N, Zhang YY, Feng DW, Huang F, Li DS, et al. Deep reinforcement learning: a survey. Front Inf Technol Electron Eng. 2020;21(12):1726–44. doi: 10.1631/FITEE.1900533.
[15] Padakandla S. A survey of reinforcement learning algorithms for dynamically varying environments. ACM Comput Surv (CSUR). 2021;54(6):1–25. doi: 10.1145/3459991.
[16] Zhang D, Han X, Deng C. Review on the research and practice of deep learning and reinforcement learning in smart grids. CSEE J Power Energy Syst. 2018;4(3):362–70. doi: 10.17775/CSEEJPES.2018.00520.
[17] Li D, Zhao D, Zhang Q, Chen Y. Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]. IEEE Comput Intell Mag. 2019;14(2):83–98. doi: 10.1109/MCI.2019.2901089.
[18] Yun J, Goh Y, Chung JM. DQN-based optimization framework for secure sharded blockchain systems. IEEE Internet Things J. 2020;8(2):708–22. doi: 10.1109/JIOT.2020.3006896.
[19] Zhong C, Lu Z, Gursoy MC, Velipasalar S. A deep actor-critic reinforcement learning framework for dynamic multichannel access. IEEE Trans Cognit Commun Netw. 2019;5(4):1125–39. doi: 10.1109/TCCN.2019.2952909.
[20] Iqbal A, Tham ML, Chang YC. Convolutional neural network-based deep Q-network (CNN-DQN) resource management in cloud radio access network. China Commun. 2022;19(10):129–42. doi: 10.23919/JCC.2022.00.025.
[21] Chen L, Hu X, Tang B, Cheng Y. Conditional DQN-based motion planning with fuzzy logic for autonomous driving. IEEE Trans Intell Transp Syst. 2020;23(4):2966–77. doi: 10.1109/TITS.2020.3025671.
[22] Ma C, Huang JB, Yang X, Yang MH. Robust visual tracking via hierarchical convolutional features. IEEE Trans Pattern Anal Mach Intell. 2018;41(11):2709–23. doi: 10.1109/TPAMI.2018.2865311.
[23] Tomasev N, Glorot X, Rae JW, Zielinski M, Askham H, Saraiva A, et al. A clinically applicable approach to continuous prediction of future acute kidney injury. Nature. 2019;572(7767):116–9. doi: 10.1038/s41586-019-1390-1.
[24] De Fauw J, Ledsam JR, Romera-Paredes B, Nikolov S, Tomasev N, Blackwell S, et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. Nat Med. 2018;24(9):1342–50. doi: 10.1038/s41591-018-0107-6.
[25] Mourgias-Alexandris G, Tsakyridis A, Passalis N, Tefas A, Vyrsokinos K, Pleros N. An all-optical neuron with sigmoid activation function. Opt Express. 2019;27(7):9620–30. doi: 10.1364/OE.27.009620.
[26] Zhao M, Guo X, Zhang X, Fang Y, Ou Y. ASPW-DRL: assembly sequence planning for workpieces via a deep reinforcement learning approach. Assem Autom. 2019;40(1):65–75. doi: 10.1108/AA-11-2018-0211.
[27] Chen Q, Zhao W, Li L, Wang C, Chen F. ES-DQN: A learning method for vehicle intelligent speed control strategy under uncertain cut-in scenario. IEEE Trans Veh Technol. 2022;71(3):2472–84. doi: 10.1109/TVT.2022.3143840.
[28] Fu Q, Li K, Chen J, Wang J, Lu Y, Wang Y. Building energy consumption prediction using a deep-forest-based DQN method. Buildings. 2022;12(2):131. doi: 10.3390/buildings12020131.
[29] Yang Y, Juntao L, Lingling P. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. CAAI Trans Intell Technol. 2020;5(3):177–83. doi: 10.1049/trit.2020.0024.
[30] Wang X, Zhang Y, Shen R, Xu Y, Zheng FC. DRL-based energy-efficient resource allocation frameworks for uplink NOMA systems. IEEE Internet Things J. 2020;7(8):7279–94. doi: 10.1109/JIOT.2020.2982699.
[31] Du H, Han P, Xiang Q, Huang S. Monkeyking: Adaptive parameter tuning on big data platforms with deep reinforcement learning. Big Data. 2020;8(4):270–90. doi: 10.1089/big.2019.0123.
[32] Lin Y, Huang J, Zimmer M, Guan Y, Rojas J, Weng P. Invariant transform experience replay: Data augmentation for deep reinforcement learning. IEEE Robot Autom Lett. 2020;5(4):6615–22. doi: 10.1109/LRA.2020.3013937.
[33] Prianto E, Kim M, Park JH, Bae JH, Kim JS. Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor–critic with hindsight experience replay. Sensors. 2020;20(20):5911. doi: 10.3390/s20205911.
[34] Shi Q, Lam HK, Xiao B, Tsai SH. Adaptive PID controller based on Q-learning algorithm. CAAI Trans Intell Technol. 2018;3(4):235–44. doi: 10.1049/trit.2018.1007.
[35] Bourebia NEH, Li C. A greedy energy efficient clustering scheme based reinforcement learning for WSNs. Peer-to-Peer Netw Appl. 2022;15(6):2572–88. doi: 10.1007/s12083-022-01368-7.