VoidScape: AI-Powered Chinese Gongbi Painting Style Transfer Tool

Student name : MINGZHAO DU
Student number: 24004238
18/03/2025
Link to git project repo:
Mingzhao-Du/VoidScape-AI-Powered-Chinese-Gongbi-Painting-Style-Transfer-Tool: An AI image processing tools

Introduction

VoidScape is an AI-driven art style transfer system specifically designed to simulate Chinese Gongbi painting—a meticulous court painting style that originated during the Song Dynasty (960-1279 AD). As a traditional Chinese painting technique, Gongbi painting incorporates the strategic use of negative space in composition, a key aspect of Chinese artistic tradition. Unlike other Chinese painting techniques such as ink wash painting, Gongbi focuses on the precision and delicacy of line work, requiring lines to be orderly, fine, and meticulous. The colours used are typically vivid, composed, bright, and elegant, with a harmonious colour palette that embodies the rich aesthetic of Chinese national traditions.

This project consists of a single-core module: AI Image Style Transfer Module – This module processes static image inputs and applies Gongbi-style transformation, preserving the delicate brushwork and colour composition characteristic of traditional Chinese fine-line painting.

The project aims to promote Chinese artistic philosophy and aesthetics, making Gongbi painting more accessible through digital means and expanding its global influence. Additionally, it serves as a creative tool for artists and designers, enabling the rapid generation of Gongbi-style visuals and broadening artistic expression.

Background

During the conceptualization of this project, I reviewed several relevant studies and literature, which provided crucial theoretical support and practical guidance for the project's development.

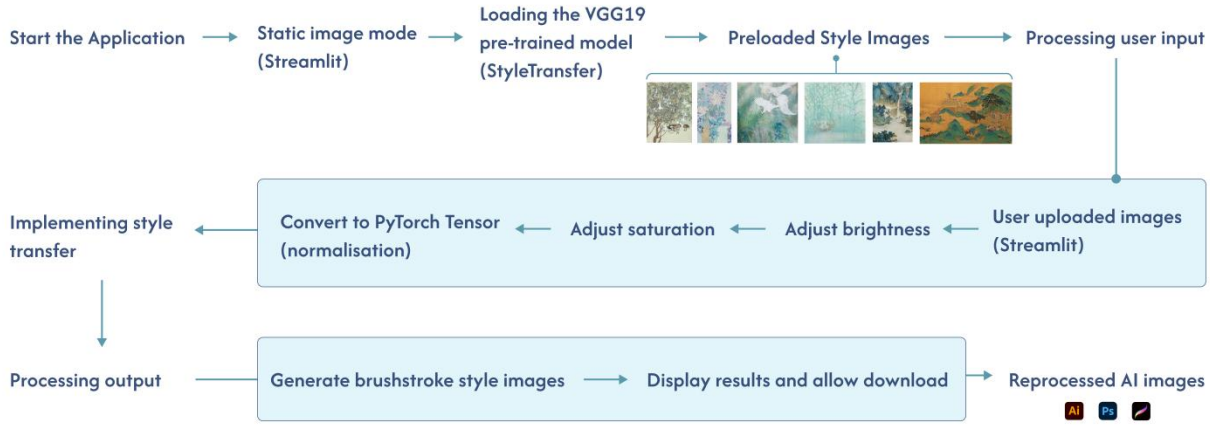
Regarding the application of Chinese painting styles in AI, a paper by Hou (2024, pp. 77-83) [1] explores the potential of AI in animating traditional Chinese painting styles, highlighting the significant role AI tools have played in advancing Chinese animation art. In recent years, Chinese painting styles have gained widespread popularity not only in animation but also across various industries such as product packaging and AAA games, achieving considerable international recognition.

Based on my research, I have contemplated the innovative direction of this project. Despite the unique technique and elegant tones of Gongbi painting, which holds a significant place in the history of Chinese art, there has been limited research in the field of artificial intelligence, particularly in the domain of style transfer. Mature applications in this area are exceedingly scarce. Therefore, I have chosen the Gongbi painting as the focal point, aiming to fill this gap in the field and promote the application of the Gongbi style in AI-generated art.

My project employs the VGG19 pre-trained model as the primary feature extraction network. Originally introduced by Simonyan and Zisserman (2015) [2], VGG19's deep structure and clear hierarchical features make it well-suited for computer vision tasks like style transfer. The model extracts multi-layer features from both content and style images, resulting in a more refined style transfer.

This project uses some content images from the *Lanscape_picture_ctGAN* dataset (2023) [3] on Kaggle for testing the generation effect. The style images were meticulously curated from publicly available resources, comprising six high-resolution Gongbi masterpieces by renowned artists. The images encompass a diverse range of themes, including landscapes and floral-bird motifs, to ensure stylistic variety in the style transfer process.

Implementation details for interactive application



(Fig 1) Flow-chart work by Mingzhao Du (2024)

As shown in the picture above (see Fig 1), the overall architecture of this project consists of an image-style transfer module designed for user interaction through static image upload.

In the image style transfer mode, users upload images (.jpg, .png, .jpeg) via a Streamlit web interface and select a target Gongbi painting style. The system provides six distinct Gongbi painting styles as references and allows users to adjust brightness and saturation to refine the visual effect. Once the 'Start conversion' button is clicked, the program processes the image, optimizing it over multiple iterations, and generates a stylized output available for JPG download.

This project employs the VGG19 pre-trained model provided by PyTorch as the core processing system for style transfer. A GitHub tutorial [4] introducing VGG19 and its associated code examples proved highly beneficial in guiding the model's implementation and integration. Within this project, VGG19 is primarily responsible for feature extraction and optimization, enabling the transfer of Gongbi painting styles. To achieve this, specific convolutional layers were selected to balance feature abstraction and computational efficiency. Low-level layers (e.g., conv1_1, conv2_1) capture fine textures, while deeper layers (e.g., conv3_1, conv4_1, conv5_1) extract more abstract stylistic features. The conv4_2 layer was designated for content preservation, ensuring that the structural integrity of the original image is maintained while allowing stylistic adaptation. These choices provide a well-structured representation of the Gongbi painting style.

The VGG19 model is implemented in 'style_transfer_model.py' and subsequently imported into 'app_VoidScape.py' for execution.During the model construction in 'style_transfer_model.py', three loss functions were used to optimize the style transfer process:

- **Content Loss**: Computes feature differences between the generated and content images at intermediate layers of VGG19 to preserve the structural integrity of the input.
- **Style Loss**: Uses Gram matrices to measure the texture similarity between the style image and the generated image, ensuring style consistency.
- **Total Variation (TV) Loss**: Enhances the smoothness of the generated image, reducing noise and improving visual aesthetics.

By leveraging PyTorch and Streamlit, this project successfully implements a VGG19-based style transfer system, providing users with an intuitive and efficient AI-powered tool for transforming static images into Gongbi-style artworks.

Details of additional experimentation

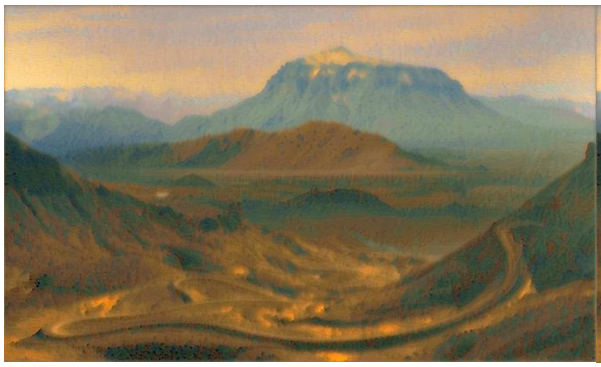
During the evaluation of the image style transfer outcomes, I observed that the generated images did not always accurately reproduce the colour tones. To rectify this issue, I incorporated adjustable parameters for brightness and grayscale, thereby allowing users to fine-tune the luminance and chromatic properties of the generated images following their individual preferences. Upon closely analyzing numerous Gongbi paintings, I discerned that they predominantly feature subdued colour saturation, with highly saturated hues selectively applied in specific regions to accentuate certain elements. This adjustment functionality facilitates a closer alignment of the generated images with the hallmark colour characteristics of the Gongbi painting.

In subsequent testing, I identified a limitation in the user experience: the absence of a real-time preview while the image was being generated. This could result in users waiting for an extended period, only to find that the final output did not align with their expectations, thereby diminishing overall usability. To address this issue, I incorporated a real-time preview function within the 'closure()' method and implemented a progress bar in the code. These enhancements provide more immediate visual feedback, improving interactivity and optimizing the overall user experience.

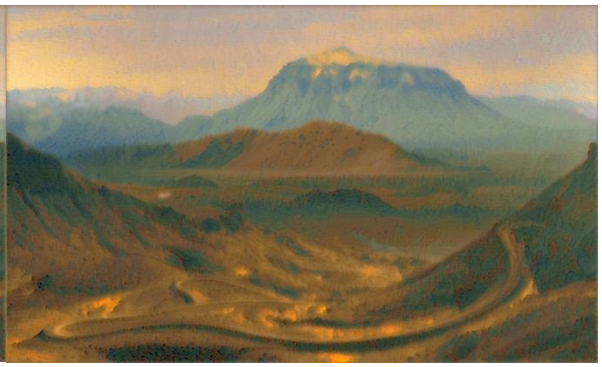
```
        def closure():
            # Real-time preview
            if run[0] % 10 == 0:
                preview_image = self.denormalize(generated_image)
                preview_placeholder.image(preview_image, caption=f"Iteration
{run[0]}", use_container_width=True)
            # Update progress
            if progress_callback and run[0] % 10 == 0:
                progress_callback((run[0] + 1) / ITERATIONS, total_loss.item())
            run[0] += 1
            return total_loss
```

```
if uploaded_file:
    # Creating progress components
    progress_bar = st.progress(0)
    status_text = st.empty()
```

The process of parameter testing and fine-tuning was a critical aspect of this project. By comparing outputs under different configurations, I aimed to determine the most effective settings. Initially, I assessed the impact of the 'STYLE_WEIGHT' parameter on style transfer quality. In the left image, 'STYLE_WEIGHT' was set to 1e5, while in the right image, it was 2e5 (see Fig 2, Fig 3). The differences were subtle, with the right image displaying slightly more refined textures upon magnification. However, higher values significantly increased computation time. Balancing visual fidelity and efficiency, I selected 'STYLE_WEIGHT = 1e5' as the optimal setting.

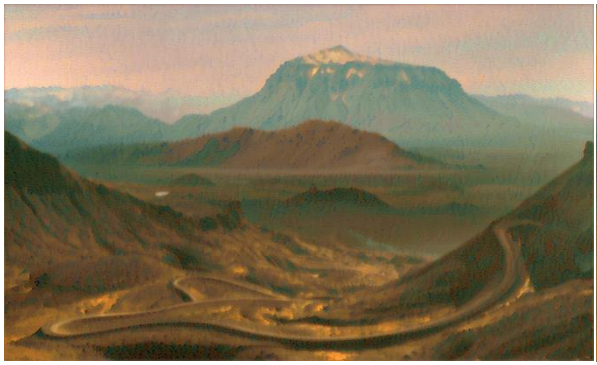


(Fig 2) 'ITERATIONS = 100, STYLE_WEIGHT = 1e5'



(Fig 3) 'ITERATIONS = 100, STYLE_WEIGHT = 2e5'

Next, I evaluated the 'ITERATIONS' parameter, testing values of 50, 100, and 200 (see Fig 4, Fig 2 and Fig 5). When 'ITERATIONS' exceeded 200, computation time increased sharply (exceeding 10 minutes), reducing efficiency. At 'ITERATIONS = 50', the generated image exhibited some colour loss but maintained overall tonal similarity to the reference style. At 'ITERATIONS = 200', colour richness and region differentiation improved, but processing time nearly doubled. Ultimately, I chose 'ITERATIONS = 100' as the optimal value, as it provided a good balance between style accuracy and processing time.



(Fig 4) 'ITERATIONS = 50, STYLE_WEIGHT = 1e5'

(Fig 5) 'ITERATIONS = 200, STYLE_WEIGHT = 1e5'

Lastly, from the standpoint of user experience, I meticulously curated a selection of six reference style images, considering factors such as tonal harmony, artistic technique, and conceptual intent, to accommodate the diverse preferences of users. For example, 'Style 1' is distinguished by its sharp, defined edges and an overall harmonious colour palette, which creates a visually cohesive and balanced composition. The image below illustrates an example generated using 'Style 1' as the reference style (see Fig 6 and Fig 7).

(Fig 6) Gongbi Painting Style 1

(Fig 7) Original and AI-generated images

Critical reflection

Strengths

The project demonstrates several strengths, particularly in its flexibility and user-friendly interface. The style transfer process is highly customizable, allowing users to easily switch between different Gongbi painting styles with a simple operation in the Streamlit interface. This enhances personalization and versatility. The model performs well with landscape images, preserving the original composition while incorporating stylistic features such as colour transitions. Additionally, the Streamlit-based interface simplifies the user experience, enabling seamless image uploads, parameter adjustments, and easy result downloads.

Limitations & Challenges

Despite these strengths, several challenges remain. The primary limitation lies in the model's handling of fine details. VGG19's deep feature extraction often overlooks the delicate brushwork and intricate textures characteristic of Gongbi art, leading to outputs that lack precision and require post-processing for professional use. Another challenge is the model's suboptimal performance with portrait images, as facial features often become distorted or lose important details, limiting the model's applicability to human subjects.

Future Improvements

If I had more time to improve my project, I would focus on optimising VGG19's ability to capture finer details, as this is a crucial factor in enhancing the quality and efficiency of the generated images. Additionally, I would explore adding a camera input feature to increase interactivity and engagement. I've only taken a more in-depth look at VGG19 so far, and many of the other pre-trained models I haven't had a chance to dig deeper into yet, but future iterations will explore integrating other models that may be better suited to the specific needs of the project.

Ethical Considerations

If the project is to be used publicly, ethical and liability concerns must be addressed in addition to technical improvements. Firstly, the project must ensure that all style reference images are legally sourced, with proper attribution and necessary authorization, especially considering the use of images from online platforms. Furthermore, the application should be strictly limited to artistic creation and academic research to prevent misuse, such as generating counterfeit artworks or misleading visual content. Finally, given that the project involves the digitization of traditional Chinese Gongbi paintings, it is essential to preserve the authenticity of the art form, ensuring that its artistic integrity and historical significance are accurately conveyed.

LLM disclaimer

Code Assistance: While developing the main files, 'style_transfer_model.py' and 'app_VoidScape.py', I used ChatGPT to inquire about the implementation of unfamiliar functions and Streamlit interface modules. Additionally, ChatGPT assisted in reviewing code details, such as parameter adjustments, and provided suggestions to optimize the code structure. During debugging, I consulted ChatGPT for error resolution strategies to enhance the stability of the code.

Report Assistance: In writing the project report, I used ChatGPT solely for English grammar checks to ensure clarity and fluency. All core content, including project conceptualization, technical implementation, and analysis, was independently developed by me.

Bibliography

[1] Hou, Q. (2024) *The Chinese Painting-style Animation Innovation in the AI Era*, Journal of Education, Humanities and Social Sciences, 46, pp. 77-83. Available at: https://doi.org/10.54097/e39z4v74

[2] Simonyan, K. and Zisserman, A. (2015) *Very Deep Convolutional Networks for Large-Scale Image Recognition,* arXiv:1409.1556. Available at: https://doi.org/10.48550/arXiv.1409.1556

[3] Truong Nguyen Duy Tan (2023) *Lanscape_picture_ctGAN.* Available at: https://www.kaggle.com/datasets/truongnguyenduytan/lanscape-picture-ctgan (Accessed: 10/3/2025).

[4] Pitsenberger, J. (2023) *CNN-Image-Style-Transfer-using-VGG19-and-PyTorch.* Available at: https://github.com/Jacob-Pitsenberger/CNN-Image-Style-Transfer-using-VGG19-and-PyTorch (Accessed: 5/3/2025).