Artwork Classification and Style Transfer

—— Deep Learning Final Project

Abstract:
The field of art presents intricate challenges and opportunities for the application of machine learning, particularly in the classification of artworks and the transformation of artistic styles. This report details a comprehensive project that leverages a curated dataset from Kaggle's "Best Artworks of All Time," encompassing 50 most influential artists and 17,548 distinct images.

In our research, we undertook two pivotal visual computing tasks: image classification and artistic style transfer. For classification, we aim to classify artworks based on certain criteria and employ both a bespoke Convolutional Neural Network (CNN) and a transfer learning approach based on ResNet50. For the style transfer task, we used models based on VGG19 and ResNet50 to adapt one painting's style to another. Additionally, a specialized CycleGAN model was developed to transform generic photographs into Claude Monet's distinct painting style. These methodologies were made accessible through three web interfaces, allowing for practical applications and interactive user engagement. We also conducted rigorous model evaluation and comparison at the end of each task, illuminating the inherent advantages and limitations of each approach.

Through the combination of these advanced techniques, the project delivers a nuanced artwork classification model and a versatile style transfer algorithm. The report showcases the thoughtful approach and innovative solutions employed in this endeavor.

Table of Contents

1. Problem Overview

Objective

Our project was bifurcated into two primary objectives:

1) Conduct image classification to accurately categorize paintings of different artists.
2) Conduct style transfer to mimic and transfer artistic styles.

Image Classification:

1) Constructed a custom CNN and performed extensive hyperparameter tuning to optimize its performance.
2) Constructed a Transfer learning Model based on the ResNet50 architecture, capitalizing on pre-trained weights to boost accuracy.

Style Transfer:

1) Implemented style transfer models utilizing both VGG19 and ResNet50 architectures. The goal was to capture the artistic essence of one painting and project it onto another.
2) To delve deeper into artistic transformations, CycleGAN was employed. It was trained using a diverse dataset comprising natural landscapes and architectural imagery. The overarching ambition was to metamorphose these images to resonate with Claude Monet's iconic painting style.

Model Deployment:

To bridge the gap between advanced computational models and end-users, we launched three web interfaces:

- An interface for classification based on the model we trained (custom CNN and transfer learning based on ResNet50)
- An interface for the VGG19 and ResNet50-based style transfer.
- An interface for the Monet-inspired CycleGAN style transformations.

Implications and Future Work:

Our comprehensive approach not only underscores the adaptability of deep learning in visual tasks but also paves the way for practical real-world applications. Future work might encompass model tuning or adjusting, and exploring other deep learning architectures for enhanced performance.

2. Exploratory Data Analysis

2.1 Data Overview

Our dataset is sourced from Kaggle's dataset "Best Artworks of All Time", which is a collection of paintings of the 50 most influential artists of all time. The dataset consists of a csv file and folders of artwork images by corresponding artists. Aside from this main dataset, we have also imported an image dataset for style transfer, which includes 100 Monet style images and 1672 photo images (testing images for Monet style transfer).
In the artists.csv file from "Best Artworks of All Time", we have information of the 50 most influential artists. The dataset has 50 rows and 8 columns. Its first 5 rows are sampled below:

```
First 5 rows of the artists file:
   id              name        years                        genre  \
0   0   Amedeo Modigliani  1884 - 1920                   Expressionism
1   1  Vasiliy Kandinskiy  1866 - 1944  Expressionism,Abstractionism
2   2        Diego Rivera  1886 - 1957       Social Realism,Muralism
3   3        Claude Monet  1840 - 1926                   Impressionism
4   4       Rene Magritte  1898 - 1967      Surrealism,Impressionism

  nationality                                                 bio  \
0     Italian  Amedeo Clemente Modigliani (Italian pronunciat...
1     Russian  Wassily Wassilyevich Kandinsky (Russian: Васи́...
2     Mexican  Diego María de la Concepción Juan Nepomuceno E...
3      French  Oscar-Claude Monet (; French: [klod mɔnɛ]; 14 ...
4     Belgian  René François Ghislain Magritte (French: [ʁəne...

                                    wikipedia  paintings
0  http://en.wikipedia.org/wiki/Amedeo_Modigliani        193
1  http://en.wikipedia.org/wiki/Wassily_Kandinsky         88
2       http://en.wikipedia.org/wiki/Diego_Rivera         70
3       http://en.wikipedia.org/wiki/Claude_Monet         73
4      http://en.wikipedia.org/wiki/René_Magritte        194
```

The schema overview of the artists.csv file is as below:

```
Schema of the artists file:
id              int64
name           object
years          object
genre          object
nationality    object
bio            object
wikipedia      object
paintings       int64
dtype: object
```

Number of paintings is the only integer feature in the dataset, and each artist has 168.92 paintings collected on average:

```
count     50.000000
mean     168.920000
std      157.451105
min       24.000000
25%       81.000000
50%      123.000000
75%      191.750000
max      877.000000
Name: paintings, dtype: float64
```

After checking for null values, we find that there is no missing value in the dataset:

```
Number of missing values in each column:
id             0
name           0
years          0
genre          0
nationality    0
bio            0
wikipedia      0
paintings      0
dtype: int64
```

In the image folders, we detect that there are 17548 images with 0 images unreadable or corrupted. 20 randomly selected images with corresponding artist names are shown below (fig1):

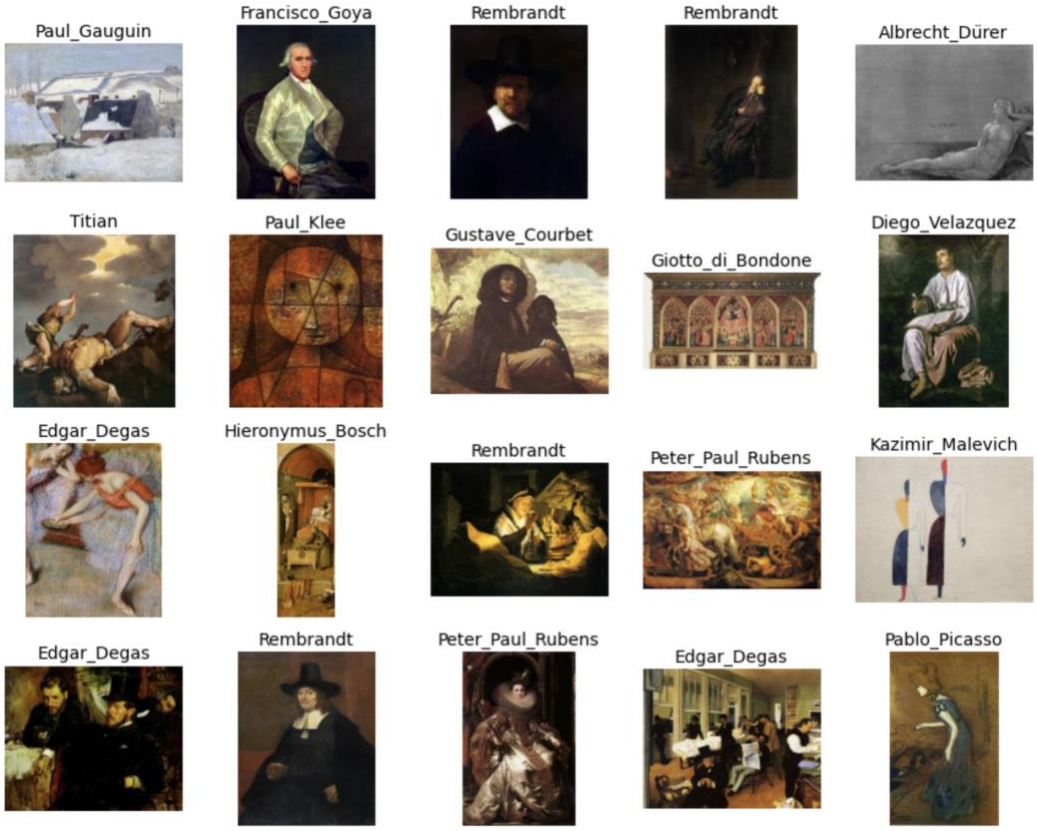


Fig1. 20 Random Images

2.2 Visualization

Painting Counts (fig2)

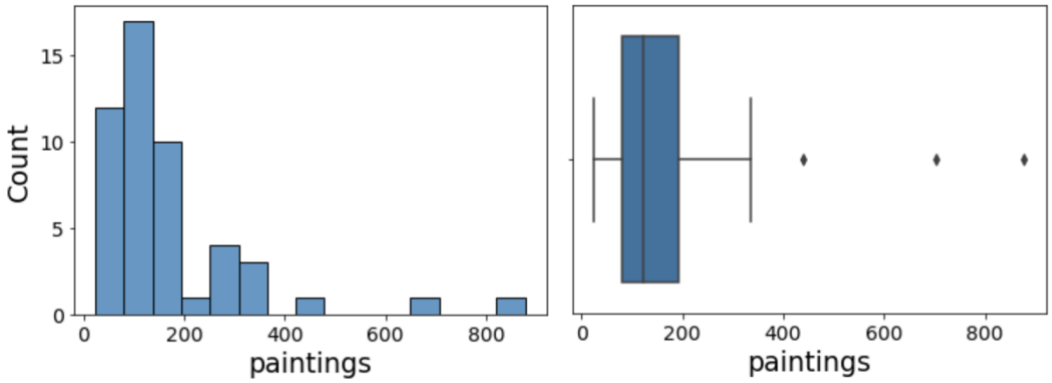


Fig2. Histogram and Boxplot of the Paintings Count

Painting Counts by Nationality (fig3)

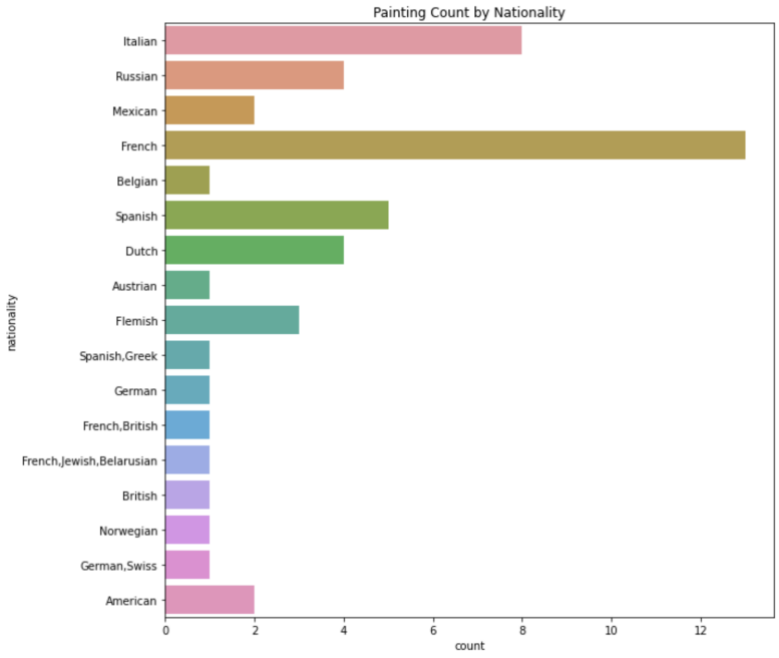


Fig3. Paintings Distribution by Nationality

Painting Counts by Artists (fig4)

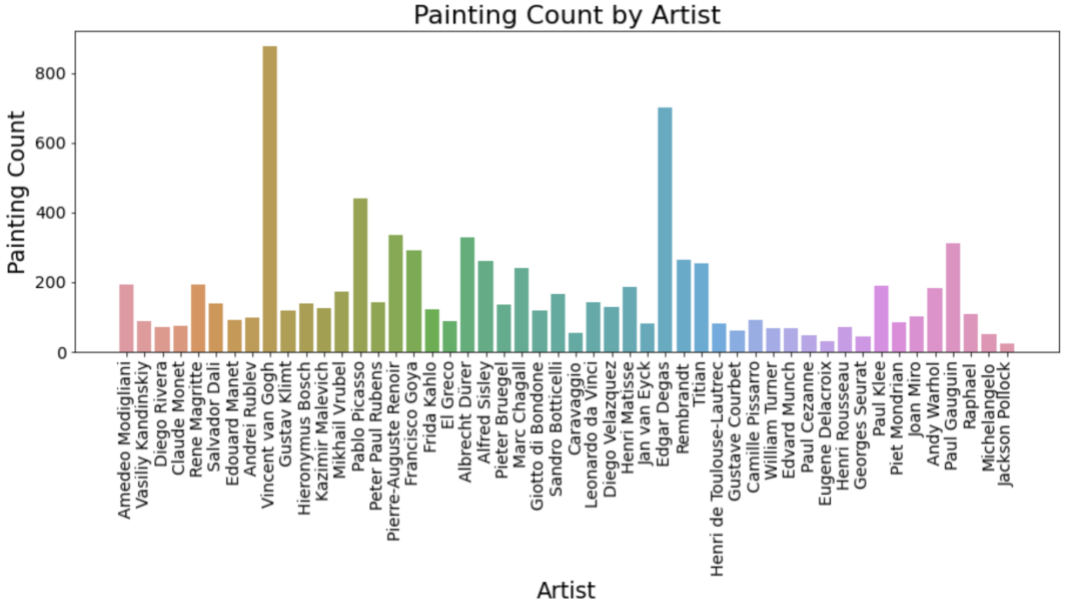


Fig4. Paintings Distribution by Artists

3. Task 1: Classification

3.1 Data Sampling and Augmentation

3.1.1 Data Sampling

In this task, we subsampled the dataset to fit our computing capacity by only selecting the top artists who have 200 or more paintings. In this way, we get 11 artists(classes) in total, and here is the histogram of the count of paintings for these top artists (fig5).

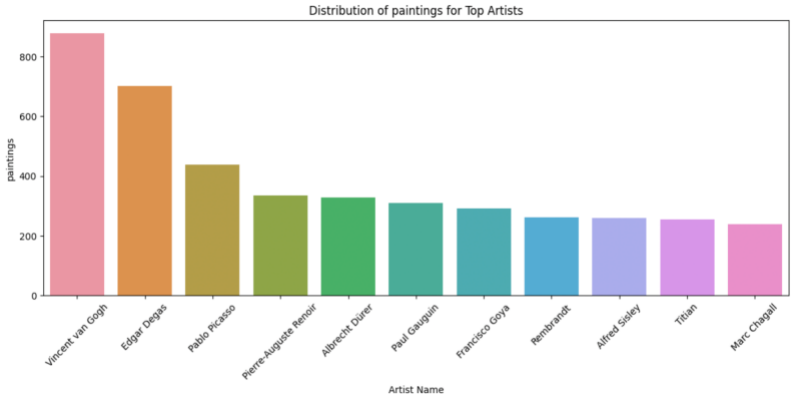


Fig5. Distribution of Paintings for Top Artists

3.1.2 Data Augmentation

To expand the diversity of the training dataset and reduce overfitting, we employ data augmentation techniques using the ImageDataGenerator class from Keras. These augmentation techniques include:

- Rotation: Images are randomly rotated by up to 45 degrees in both clockwise and counter-clockwise directions.
- Width and Height Shifting: The images are shifted along their width and height by a maximum of 20%, creating slightly translated versions of the original images.
- Shear: Random shear transformation is applied, altering the shape of the images to introduce additional variations.
- Zooming: Random zooming within a range of 20% is performed to mimic different scales and perspectives.
- Horizontal Flipping: Images are horizontally flipped with a probability, introducing mirror-image variations.
- Rescaling: Pixel values are normalized to the range [0, 1], reducing the impact of differing pixel intensity scales.

Then the dataset is divided into training and validation subsets. We use an 80-20 split, where 80% of the data is used for training, and the remaining 20% is reserved for model validation. And for both the training and validation subsets, we create data generators using the ImageDataGenerator class. A random image and its augmentation vision can be seen in fig6.

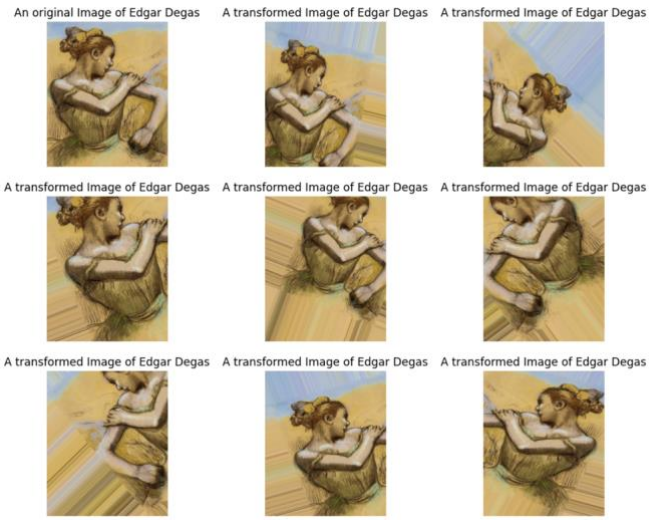


Fig6. A ransom sample and its augmented version

3.2 Custom CNN

3.2.1 Model Architecture

- **Convolutional Layers:** Three Conv2D layers with 32, 64, and 128 filters respectively, apply 3x3 kernels to capture salient features from the input images. ReLU activation functions introduce non-linearity to enhance feature extraction, while MaxPooling2D layers downsample the feature maps.
- **Flatten Layer:** Following the Convolutional and MaxPooling layers, a Flatten layer converts the 3D feature maps into a 1D vector, preparing the data for fully connected layers.
- **Fully Connected Layers:** Two Dense (fully connected) layers with 64 neurons and a Dropout layer with a rate of 0.5 are employed to facilitate complex feature combinations and reduce overfitting. The final Dense layer has 11 neurons, corresponding to the number of artist classes for multi-class classification. The softmax activation function converts the model's outputs into class probabilities.

To be more specific, the complete model architecture is shown in fig7.

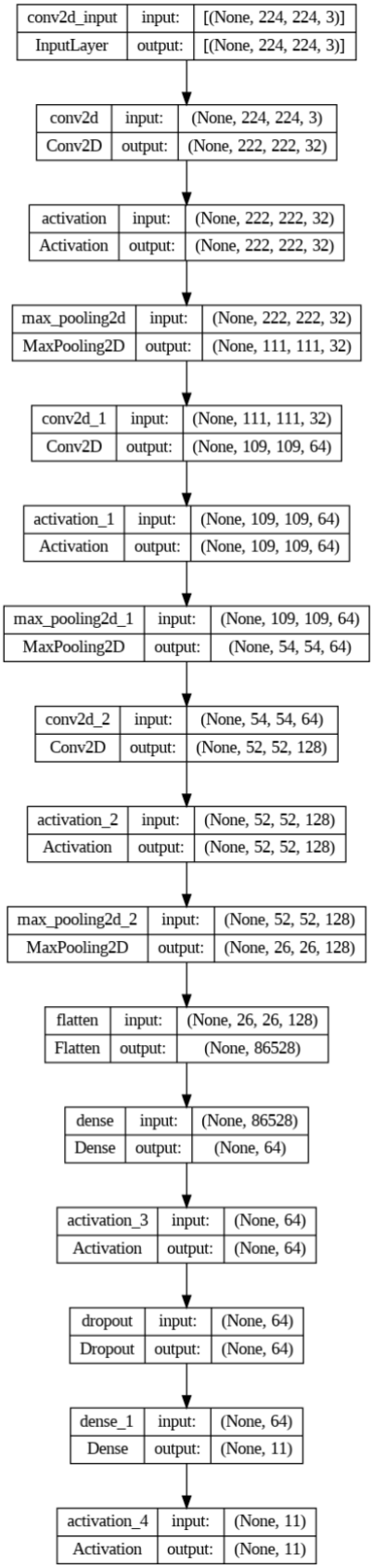


Fig7. Custom CNN Architecture

3.2.2 Model Training

The training process aims to optimize the model's parameters using the Adam optimizer and categorical cross-entropy loss function. To ensure efficient convergence and prevent overfitting, we introduce two callbacks:

- EarlyStopping: With a patience of 3, EarlyStopping monitors the validation loss and halts the training if no significant improvement is observed within three epochs. This prevents unnecessary computations and saves training time while ensuring optimal model performance.
- TensorBoard: We use TensorBoard, a visualization tool, to monitor and analyze the model's training progress.

Fig8 is how the train and validation accuracy and loss change over the epoch. (The blue line represents the training accuracy/loss, and the orange line represents the validation accuracy/loss)

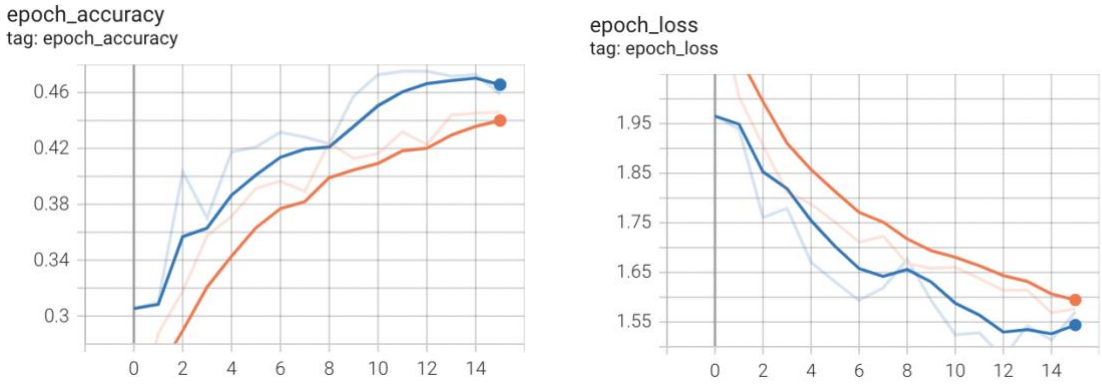


Fig8. Train/Validation Accuracy/Loss via Epoch

3.2.3 Hyperparameter Tuning

We adopt the Kerastuner library to facilitate hyperparameter optimization, and the key hyperparameters we explored are the number of dense units and the learning rate for 'Adam' optimizer. In detail, we conduct a RandomSearch over five different hyperparameter combinations, each trained twice (10 epochs for once) to obtain robust results. Here is the result:

- The best number of units in the dense layer is 384
- The best learning rate in the optimizer is 0.0001

Assume the tuned model 'best_model'. Fig9 is how the train and validation accuracy and loss of 'best_model' change over the epoch. (The blue line represents the training accuracy/loss, and the orange line represents the validation accuracy/loss)

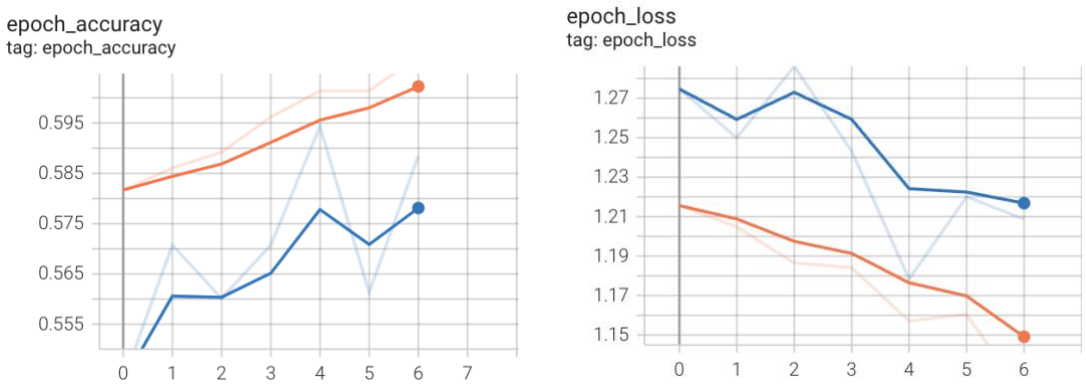


Fig9. Train/Validation Accuracy/Loss via Epoch

3.2.4 Experimental Results

The train accuracy and validation accuracy before and after tuning are listed in the table1

	Train Accuracy	Validation Accuracy
Before Tuning	0.512	0.451
After Tuning	0.658	0.576

Table1. Train/Validation Accuracy Before & After Hyper-Parameter Tuning

- Overall, the hyperparameter tuning process has effectively fine-tuned the CNN architecture, resulting in a considerable performance gain in both training and validation accuracy.
- The training accuracy increased significantly to 65.8%, showcasing the effectiveness of the optimized hyperparameters in better capturing complex patterns and features within the training dataset. Furthermore, the validation accuracy showed remarkable improvement, reaching 57.6%. This indicates that the tuned model generalizes better to unseen data, making it more adept at recognizing artists from diverse artworks.
- However, there is still ample room for further enhancement in the artist recognition model. The obtained training accuracy validation accuracy indicates that the model is yet to reach its full potential.
- Furthermore, we could find the confusion matrix in fig10, which could also support our findings. And we could conclude that in this case, the model performs the best in classifying Albercht_Durer's work (may be due to his strong personal style) while performs worst in classifying Titan's work.

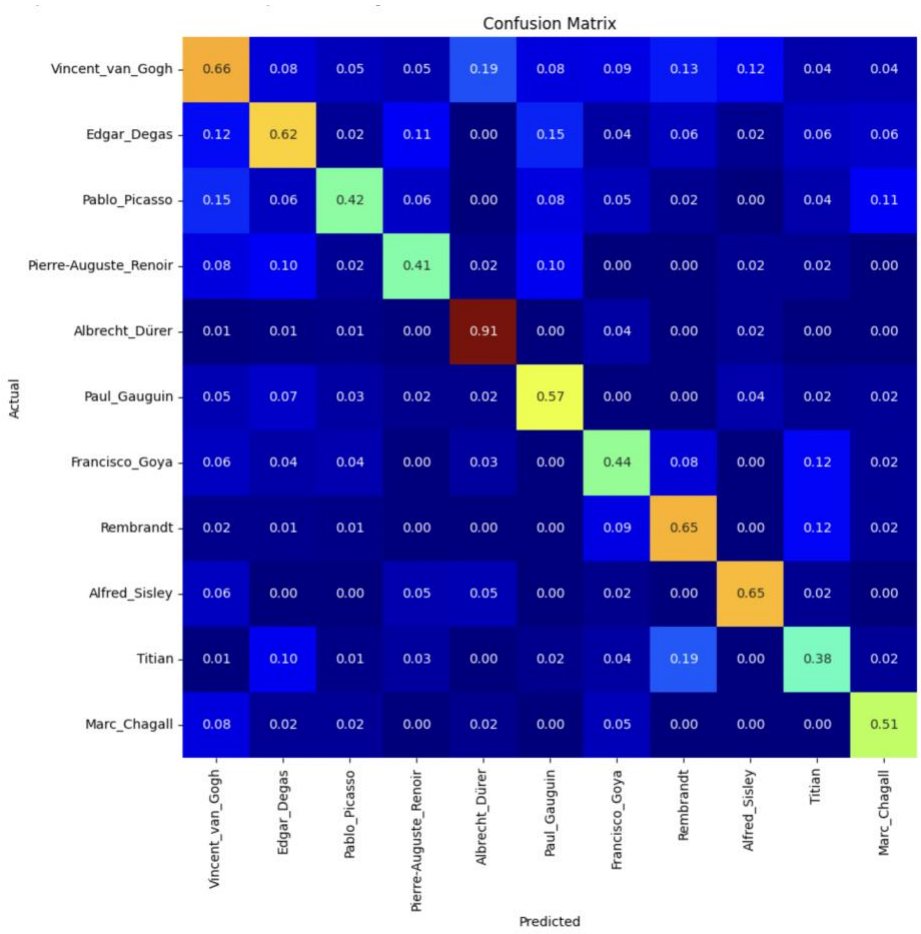


Fig10. Confusion Matrix (Result of Custom CNN after Hyper-Parameter Tuning)

3.3 Transfer Learning - Based on ResNet50

In this part, we build and train a deep learning model for image classification using the ResNet50 architecture with transfer learning.

3.3.1 Phase 1 - Full Model Traning

- Load pre-trained ResNet 50 and add some additional layers:
  - Flatten the output of the base model to prepare it for the fully connected layers.
  - Add a Dense (fully connected) layer with 512 units, using ReLU activation and batch normalization.
  - Add another Dense layer with 16 units, ReLU activation, and batch normalization is added.
  - Finally, a Dense layer with the number of classes and softmax activation.
- During this phase, we train the entire model including both the pre-trained ResNet50 layers and the additional custom layers added for this task.

In this phase, all layers in the model are set to be trainable, allowing the model to adjust both the pre-trained features and the new task-specific features. The primary goal of this

phase is to adapt the model to the dataset while leveraging the knowledge learned from the ImageNet dataset on which the ResNet50 was originally trained.

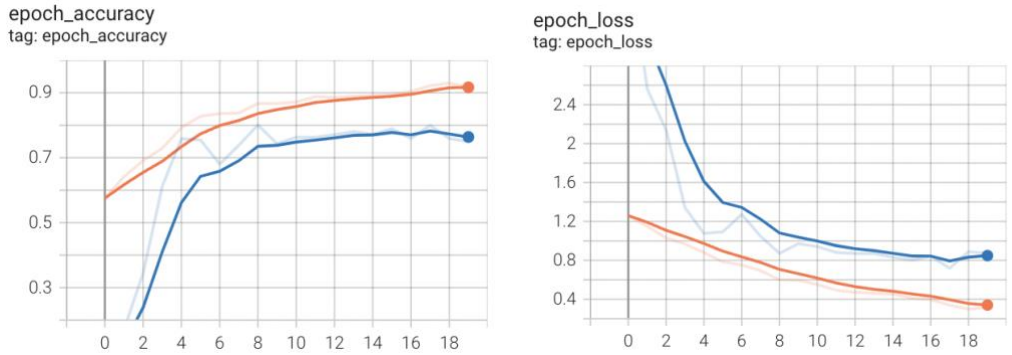


Fig11. Train/Validation Accuracy/Loss via Epoch

3.3.2 Phase 2 - Fine Tuning

- Freeze (set to non-trainable) the core layers of the pre-trained ResNet50.
- Only trained the additional custom layers and the first 50 layers of the base model (top layers).

The custom layers are more task-specific and responsible for the final classification, while the first 50 layers of the base model can be fine-tuned to a lesser extent to adapt to your specific task.

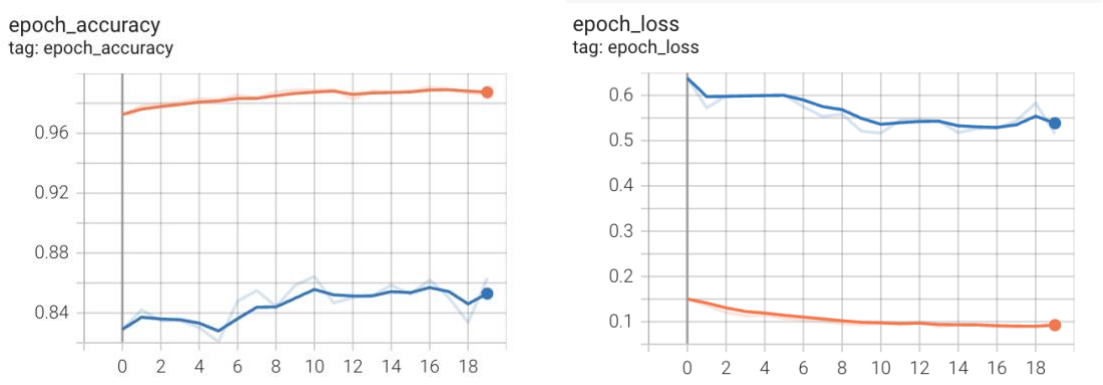


Fig12. Train/Validation Accuracy/Loss via Epoch

3.3.3 Experimental Results

The train accuracy and validation accuracy before and after tuning are listed in the table2, and the confusion matrix is shown in fig13.

Train Accuracy	Validation Accuracy
0.9937	0.8571

Table2. Train & Validation Accuracy of the Transfer Learning Model Based on ResNet50

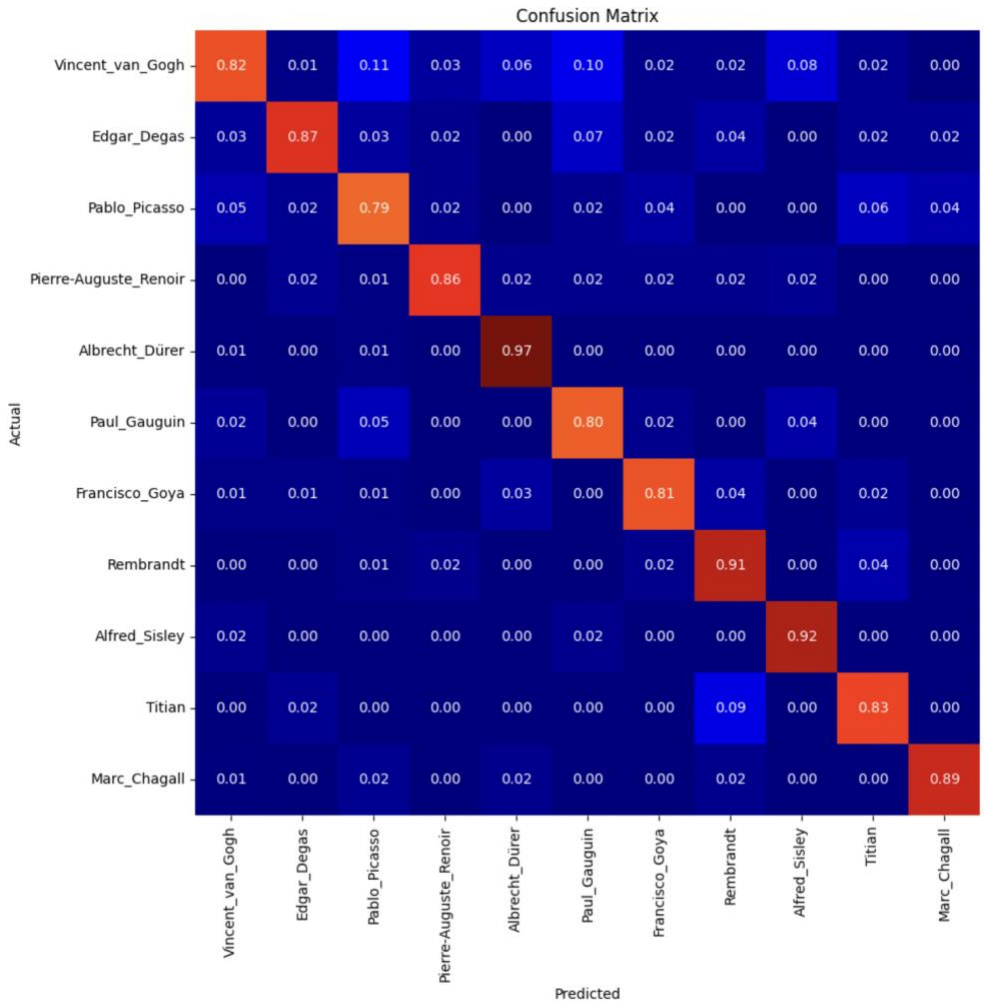


Fig13. Confusion Matrix (Result of Transfer Learning Based on ResNet50)

- The model achieves a very high training accuracy of approximately 99.37%. This indicates that the model has learned to fit the training data very well.
- The validation accuracy is around 85.71%. It is lower than the training accuracy, which is expected, as the model might be slightly overfitting to the training data.
- A confusion matrix provides a more detailed performance analysis by showing how well the model classifies different classes. The transfer learning approach exhibits pretty high accuracy in recognizing artists, as evident from the diagonal elements of the confusion matrix.

Considering the outstanding performance of the transfer learning model based on ResNet 50, here we also show its prediction result of 4 random images: (fig14)

Fig14. Prediction Result on Random Images

3.4 Model Evaluation and Comparison

Here we list a table(table3) summarizing the train and validation accuracy of the Custom CNN
after hyper-parameter tuning and fine-tuned transfer learning model based on ResNet50:

	Train Accuracy	Validation Accuracy
Custom CNN after hyper-parameter tuning	0.6583	0.5761
Transfer Learning Based on ResNet50	0.9937	0.8571

Table3. Model Comparison

- **From accuracy table above:** the transfer learning approach based on ResNet50 demonstrated remarkable performance. The model achieved an impressive training accuracy of 99.37% and a validation accuracy of 85.71%. Leveraging the pre-trained ResNet50 architecture on a larger and more diverse dataset allowed the model to extract powerful and meaningful features, resulting in a highly accurate artist recognition system.
- **From the confusion matrix listed in 4.2.4 and 4.3.3:** The transfer learning approach exhibits higher accuracy in recognizing artists, as evident from the diagonal elements of the confusion matrix. The majority of predictions fall on the correct class, resulting in fewer misclassifications compared to the custom CNN.
- **From the training efficiency:** The transfer learning approach significantly reduces the training time and computational resources required, as it starts with pre-trained weights, enabling faster convergence and better optimization.

3.5 Strengths and Limitations

	Pros	Cons
Custom CNN	• **Flexibility:** The custom CNN allows full control over the architecture, providing flexibility to tailor it to the specific task and dataset. • **Interpretability:** Being a manually designed model, it is easier to interpret and understand the learned representations	• **Architecture Complexity:** Designing an optimal CNN architecture requires expert knowledge and extensive experimentation, which can be time-consuming and challenging. • **Limited Generalization:** The custom CNN may struggle to generalize well to diverse artistic styles or when the training data is limited.
Transfer Learning Model Based on ResNet50	• **High Accuracy:** Leveraging knowledge from a large dataset enables the model to capture intricate features and achieve superior accuracy. • **Generalization**: ResNet50 is highly suitable for tasks with diverse and complex data, such as artist recognition. • **Faster Training**: Fine-tuning a pre-trained model significantly reduces the training time and computational resources required compared to training from scratch.	• **Limited Interpretability:** The complexity of the pre-trained ResNet50 model may make it less interpretable, as the learned representations are not explicitly defined. • **Domain Shift:** The pre-trained model may have been trained on images from different domains, potentially causing a domain shift that could affect performance.

Table4. Strengths and Limitations of Different Models

In general, yhe custom CNN offers flexibility and interpretability but may lack the generalization power needed for artist recognition due to manual design and limited data. On the other hand, the transfer learning approach based on ResNet50 excels in accuracy and generalization by leveraging pre-trained features, making it an excellent choice for this task. Despite the limited interpretability, transfer learning with ResNet50 proves to be highly efficient, achieving superior results with reduced training time, which makes it a more practical and powerful solution for artist recognition.

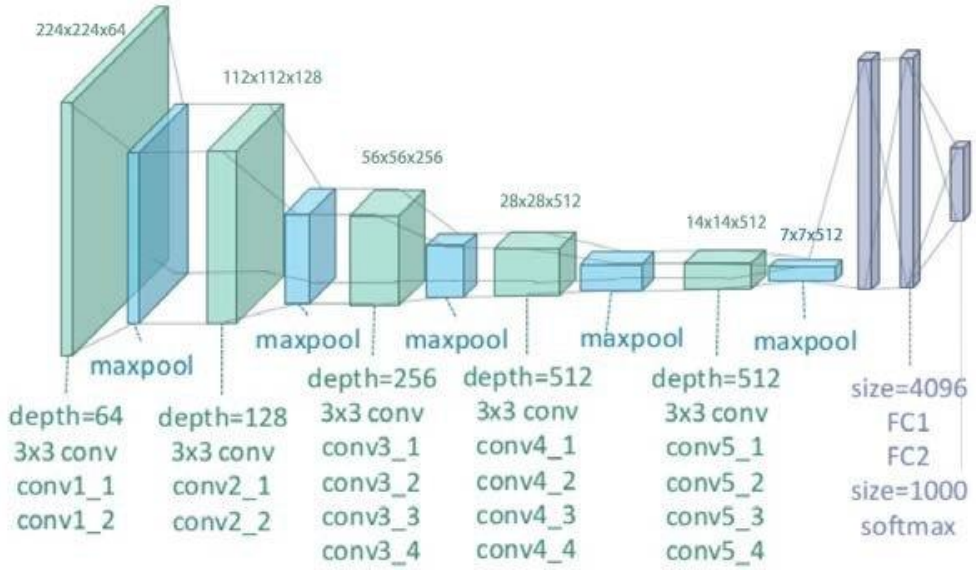
4. Task 2: Style Transfer

4.1 Model 1: Transfer Learning - Based on VGG19

In this part, we use the self portrait of Van Gogh, a color oil painting, as the content image and Francisco Goya's black and white study for Capriccio, plate 33, as the style image.

4.1.1 VGG19 and Style Transfer

VGG19 is a deep convolutional neural network with 19 layers (16 convolutional layers, 3 fully connected layers). This depth allows the network to capture intricate patterns and details in images.Unlike some other networks that mix various types of layers, VGG19 uses a very consistent architecture. It mainly uses 3x3 convolutional filters (which are the smallest size to capture the notion of left/right, up/down, center) and 2x2 pooling layers. The VGG19 network can extract features from content and style images at various levels of abstraction. This is crucial for style transfer as the intricate details of the style image and the broader constructs of the content image need to be effectively captured. In style transfer, the intermediate layers of the VGG19 model are often used to capture the style of an image. These layers effectively capture textures and patterns, which are essential for the style of an image. The deeper layers of the VGG19 model, which capture high-level content features, are used to retain the content of the original image during style transfer. VGG19's pre-trained weights allow us to leverage the features it has already learned, which aids in the convergence and effectiveness of the style transfer process.



4.1.2 Model Selection and Configuration

We employed the pre-trained VGG19 model. We ignore the fully connected layer (set by include_top=False), because we solely need to utilize the intermediate convolutional layers for extracting content and style features. All layers of the VGG19 were set to be

non-trainable (using model.trainable=False), as there was no need for further training of this model; our primary objective was feature extraction.
For content representation, we selected the 'block3_conv4' layer of VGG19, given its capacity to capture high-level features of an image. While, for style representation, we chose five layers spanning from shallow to deeper layers, enabling the capture of stylistic features across different levels.
This style transfer training undertaken is simple, leveraging just a single image as the content source and another as the style source. We exclusively relied on VGG19's weights for image generation, refraining from fine-tuning the top layers of VGG19 using an entire collection of a specific artist's works.

4.1.3 Hyperparameter Selection

In our approach, we employed Random Search for hyperparameter optimization, aiming to find the optimal set of parameters that would produce the most visually pleasing style-transferred images. Random Search is known for its effectiveness in high-dimensional spaces, offering a better chance to find a global optimum compared to exhaustive Grid Search.
For our experiment, the hyperparameters under consideration included the learning rate, style weight, content weight, and weights for different convolutional layers within the VGG19 network, which are essential in controlling the balance between style and content in the generated images. After each iteration, the total cost of the style transfer, which combined both style and content costs, was evaluated. At the end of the Random Search process, we selected the hyperparameter combination that resulted in the lowest total cost, which is *'learning rate': 2.0213317396466923, 'style wight': 0.03645219149405942, 'content wight': 0.03581117505484557, 'style layer wights': [1.26322337193265, 3.5051343379434887, 0.05154450093504583, 0.184438134398096, 0.03642465476026685]*. Subsequently, we utilized this optimal set of parameters to generate our final style-transferred images.

4.1.4 Experimental Results

From the cost converge curve fig15, we can see that in the initial phase, the model's cost value declines rapidly, indicating that the model learned a significant amount of information at the beginning. This is because the pre-trained model has already learned many general features and only needs to refine and adapt to specific tasks. Subsequently, after epoch 40, the decline slows down and stabilizes, implying that the model is beginning to converge. During this phase, the model is still fine-tuning some features or addressing more challenging patterns, but the learning effect is already quite good.

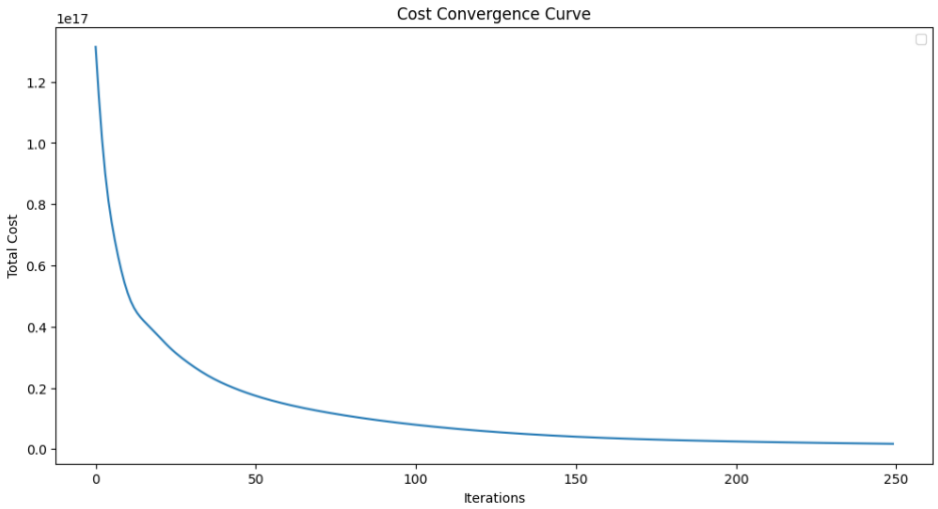


Fig15. Cost Convergence Curve of VGG19 Transfer Learning

We can see that fig16 through the transfer learning of VGG19, the source image and the target style have been merged, and the effect of style transfer is obvious. The characteristics of the target style such as color are clearly reflected in the generated image, and the original color of Van Gogh self-portrait is transformed into almost black and white. The main content structure and character boundaries of the source images remain clear after style transfer, which indicates that the algorithm successfully balances style and content. The transitions between style and content are natural, with no glaring breaks or dissonances.

The deficiencies are: Van Gogh's beard is still brown; the lines of the generated picture imitate the lines of the clothes of the characters in the style pictures, but the category of watercolor painting is not learned from the whole, and the strokes of the generated picture are still similar to oil paintings.

In general, this style transfer performs well in the use of color, but the learning of lines and strokes still needs to be strengthened.

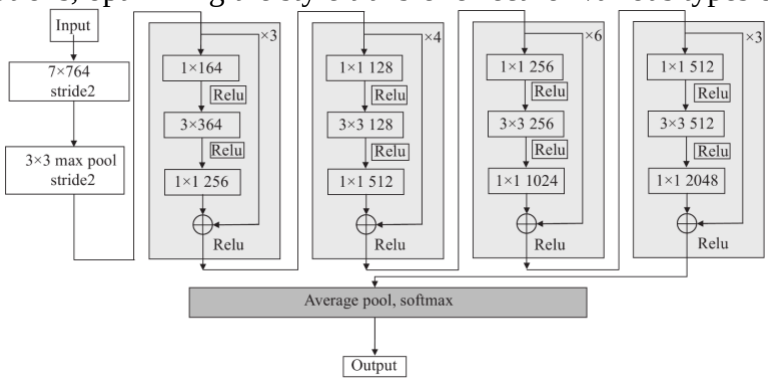
Fig16. Style Transfer Result of VGG19

4.2 Model 2: Transfer Learning - Based on ResNet50

In this part, we also use the same content image and style image as the above model, in order to do comparison.

4.2.1 ResNet50 and Style Transfer

ResNet50 stands for a 50-layer deep "Residual Network". The core idea behind ResNet is the introduction of "skip connections" or "shortcut connections" that allow activations to bypass one or more layers. With increasing network depth, most networks face the problem of vanishing gradients, where the gradients of the loss function diminish to zero, making the network hard to train. ResNet alleviates this issue with its residual blocks, ensuring the network can be trained deeper without sacrificing performance.
ResNet50's depth means that it extracts features at various levels of abstraction. For style transfer, this is advantageous as we can use different layers to represent content and style, with deeper layers capturing content and shallower layers capturing style. The residual learning mechanism ensures that the features extracted are robust and capture intricate patterns in images, making the content and style representations more distinct and expressive. Given the pre-trained weights on datasets like ImageNet, ResNet50 understands a wide variety of image content. This general knowledge of images can be leveraged for more effective content and style separations in style transfer tasks. With its many layers, users can experiment with different combinations of layers for content and style representations, optimizing the style transfer effect for various types of images.



4.2.2 Model Selection and Configuration

We employed the pre-trained ResNet50 model. We also ignore the fully connected layer. We initialize the model with weights pre-trained on the ImageNet dataset (set by weights='imagenet'). All layers of the ResNet50 were also set to be non-trainable.
Utilizing ResNet50 for style transfer bears certain resemblances to employing VGG19. However, the architecture of ResNet50 is decidedly more intricate, leveraging residual connections to train deeper networks. As a consequence, the selection of appropriate layers for representing content and style becomes even more paramount.
It is noteworthy that 'conv3_block4_out' serves a dual role as both content and style layers. Within ResNet, 'conv3_block4_out' occupies an intermediate position, granting ample abstraction while still retaining some spatial information about the image. This makes it suitable as a content layer, which predominantly focuses on high-level and

abstract features of the image, such as objects and their arrangement, typically drawn from deeper layers of the network. On the other hand, style layers are more concerned with varying textures and color distributions rather than the precise layout. Typically, in a bid to capture style details across different scales, layers ranging from shallow to deep are chosen as style layers. Hence, intermediate layers, including 'conv3_block4_out', capture both moderate and fine-grained style details. In conclusion, a single layer can furnish valuable insights about both content and style simultaneously. By incorporating 'conv3_block4_out' into the style layers, we can extract moderate style attributes from this layer; and when treated as a content layer, it conveys meaningful information about the image's content.

4.2.3 Hyperparameter Selection

We also used Random Search for this model, and used the same value choices as the former model. This time the best combination is *'learning rate': 1.507645417472133, 'style wight': 0.060063177258714934, 'content wight': 0.021136481251005562, 'style layer wights': [0.08052899504643046, 0.3727774574590765, 3.752271001977682, 9.886461318685159, 3.4500491708372483]*, which are significantly different from the previous VGG19 model.

4.2.4 Experimental Results

From the cost converge curve fig17, we can see that the general shape of the curve is similar to that of the previous model, but the convergence is slower, which is around the 80th epoch.

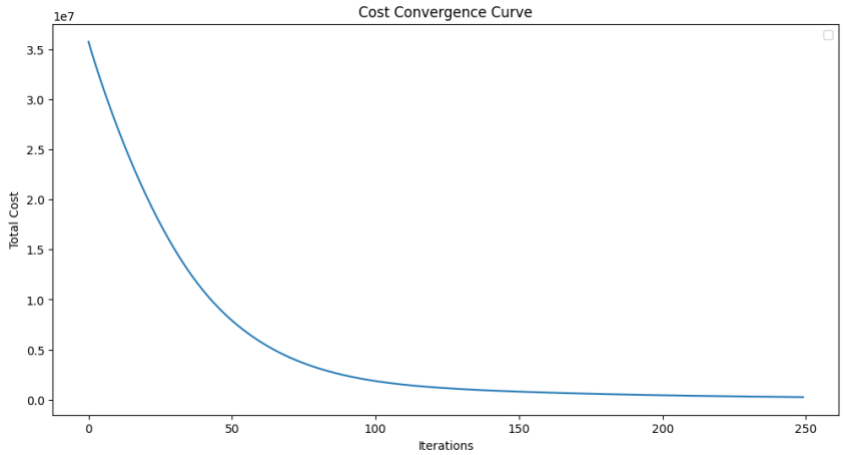


Fig17. Cost Convergence Curve of ResNet50 Transfer Learning

Transfer Learning of ResNet50 can also do a satisfying job of Style Transfer given the content image and style image. While Van Gogh's beard is still brown, just like the former VGG19 model, the lines of the face and the clothes of the generated picture are more like those in the style image.

In general, this style transfer performs well in both the use of color and the learning of lines and strokes (fig18).

Fig18. Style Transfer Result of ResNet50

4.3 Model 3: CycleGAN

To delve deeper into artistic transformations, CycleGAN was employed. It was trained using a diverse dataset comprising natural landscapes and architectural imagery. The overarching ambition was to metamorphose these images to resonate with Claude Monet's iconic painting style.

4.3.1 CycleGAN and Style Transfer

CycleGAN is a Generative Adversarial Network (GAN) that uses two generators and two discriminators. We call one generator $G_{A \to B}$. It converts images from the A domain to the B domain. The other generator is called $G_{B \to A}$, and converts images from B to A. Each generator has a corresponding discriminator, which attempts to tell apart its synthesized images from real ones. CycleGAN is a model that aims to solve the image-to-image translation problem. The goal of the image-to-image translation problem is to learn the mapping between an input image and an output image using a training set of aligned image pairs.
Traditional GANs require paired training data, meaning if we want to translate between two domains (e.g., horse and zebra), we would need many pairs of corresponding images in both domains. CycleGAN, on the other hand, eliminates the need for paired data and only requires that you have a collection of images from each domain. One of the key innovations in CycleGAN is the cycle consistency loss. If we transform an image from domain A to domain B and then back to domain A, the cycle consistency loss ensures that the final image is close to the original. This mechanism allows the model to learn mappings in both directions (A to B and B to A) simultaneously.

4.3.2 Model Architecture

a. Generator (get_generator function)

The primary goal of the generator is to produce a new output image from an input image. It mainly consists of the following three parts:

- Encoder
  - Uses convolutional layers to progressively reduce the image's dimensions.
  - Incorporates instance normalization and ReLU activation functions.
- Transformer Block
  - Comprises multiple residual blocks, each containing two convolutional layers.
  - The output of this block is added to its input, forming a residual connection.
- Decoder
  - Uses transposed convolutional layers to progressively increase the image's dimensions.
  - Incorporates instance normalization and ReLU activation functions.
  - Contains skip connections with the encoder, aiding in recovering image details.

b. Discriminator (get_discriminator function)

The task of the discriminator is to determine whether the input image is "real" or "generated". Its structure is as follows:

- Builds its structure multiple times using the encoder function.
- Each encoder block employs LeakyReLU as its activation function.
- The final layer is a convolutional layer, outputting a value indicating whether the image is real or generated.

c. CycleGAN Model (CycleGan class)

CycleGAN integrates two generators and two discriminators, aiming to perform image-to-image translation without paired data. Specifically:

- Monet Generator: Converts photos into Monet-style paintings.
- Photo Generator: Transforms Monet-style paintings into photos.
- Monet Discriminator: Differentiates between real Monet paintings and generated ones.
- Photo Discriminator: Differentiates between real photos and generated ones.

During training, the model strives to fulfill the following objectives:

- Adversarial Objective: The generator aims to produce fake images, while the discriminator tries to distinguish between real and fake images.
- Cycle Consistency Objective: An image transformed from one domain to another and then back to the original domain should resemble the original image.
- Identity Objective: When passing an image from a certain domain to its corresponding generator, it should closely match the original image.

4.3.3 Loss Function Definition

a. Adversarial Loss

For both the generator and discriminator, the Mean Squared Error (MSE) is used as their adversarial loss.

- Discriminator Loss (discriminator_loss function):

    - real_loss: Calculates the MSE between the real images and an array entirely of 1s. This indicates our desire for the discriminator to recognize real images as "real".
    - generated_loss: Calculates the MSE between the generated images and an array entirely of 0s. This indicates our desire for the discriminator to recognize generated images as "fake".
    - total_disc_loss: This is the average of the two aforementioned losses.
- Generator Loss (generator_loss function):
    - Calculates the MSE between the generated images and an array entirely of 1s. This indicates our desire for the generated images to be recognized by the discriminator as "real".

b. Cycle Consistency Loss

The Cycle Consistency Loss (calc_cycle_loss function) measures the difference between the original image and the image after two transformations (from one domain to another and then back to the original domain).

- Uses the L1 norm (i.e., the sum of absolute values) to calculate the difference between the original image and the image after bidirectional transformation.
- Multiplied by a hyperparameter, Lambda, to control the weight of this loss in the overall loss.

c. Identity Loss

The Identity Loss (identity_loss function) compares the difference between the image and its generator. For instance, if we pass a photo to the "photo-to-photo" generator, it should be very similar to the original photo.

- Uses the L1 norm to calculate the difference between the original image and the image generated by its corresponding generator.
- Multiplied by half of a hyperparameter, Lambda, to control the weight of this loss in the overall loss.

In summary, the combination of these loss functions ensures that CycleGAN can perform image-to-image translation without paired training data. The adversarial loss ensures that the generated images are realistic, while the cycle consistency loss and identity loss ensure that the key information of the original image is retained during the transformation process.

4.3.4 Experimental Results

We used TensorBoard to show the loss converge curve of each discriminator and generator. As we can see, fig19 both discriminators converge well, while the loss of both generators is unstable, which maybe because the architecture of generators is much more complex than that of the discriminators.

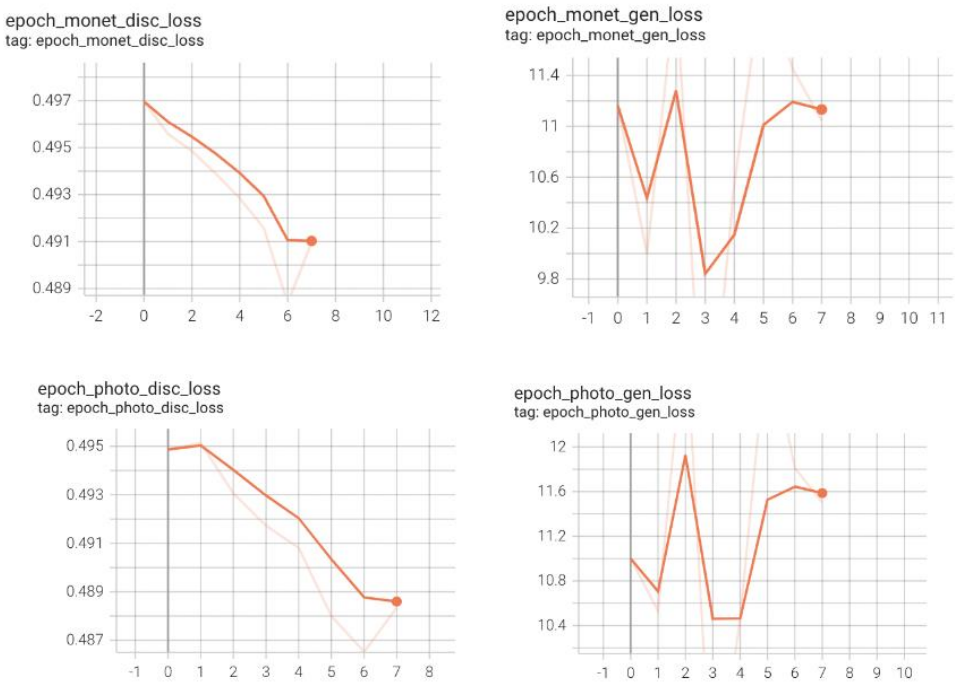


Fig19. loss converge curve of each discriminator and generator

Because of the computing limit of Google Colab, although we set the training epoch as 120, the training stopped at the 8th epoch.

The generated images fig20 are able to retain the contours and lightness of the original images, but they have not effectively captured the style of the style images. The cycled image should ideally resemble the original image, as it represents the result of the image undergoing two transformations: first from the original domain to the target domain, and then back from the target domain to the original domain. However, in this experiment, there is still a significant difference between the cycled image and the original image. This indicates that our cycle consistency loss is relatively high, and our model is losing a significant amount of information from the original image during the transformation process.

Overall, the results of style transfer using CycleGAN are not very satisfactory. We need more computational power to support training for additional epochs.

Fig 20. CycleGAN style transfer result

4.4 Model Evaluation and Comparison

Unlike Classification, style transfer is an unsupervised learning, so indicators such as accuracy cannot be used to evaluate the effect of style transfer. We mainly evaluate the two models from the two aspects of the convergence speed of the loss and the naked eye. First, for the two transfer learning tasks, because the model architectures are different, we can't directly compare the absolute value of the loss. But it is obvious that the ResNet Model converges faster than the VGG Model. For the CycleGAN, both discriminators converge well, while the loss of both generators is unstable.
Second, we can see from the generated image results that the ResNet Model performs best, while the CycleGAN performs worst. The VGG Model performs well in the use of color, but the learning of lines and strokes still needs to be strengthened. The ResNet Model performs well in both the use of color and the learning of lines and strokes. The results of style transfer using CycleGAN are not very satisfactory because the model even can't learn enough of the style.
Here are our evaluation table for the models we used (table5)

	Total Style	Color	Lines & Strokes	Total Performance
VGG19	Good	Good	Middle	Good
ResNet50	Best	Good	Best	Best
CycleGAN	Worst	Worst	Good	Worst

Table 5: Model generation output evaluation

4.5 Strengths and Limitations

Style transfer is one of the more complex problems in deep learning. Using a simple Customized Model, the transfer results are often unsatisfactory, so we need to utilize more sophisticated models. The three models we used in our experiments each have their own strengths and weaknesses.

	Pros	Cons
VGG19 Transfer Learning	• **Proven Architecture:** VGG19 has been validated on large image datasets and exhibits good performance. • **Style and Content Separation:** The intermediate layers of VGG19 are used to separate style and content, making style transfer more intuitive.	• **Computational Resources:** Requires relatively more computational resources, especially for high-resolution images. • **Not Real-time:** For every pair of style and content images, a separate optimization process is needed.
ResNet50 Transfer Learning	• **Depth:** Due to its depth, ResNet50 can capture complex style features. • **Residual Connections:** Improves the training process, allowing for deeper network architectures.	• **Computational Demands:** Similar to VGG19, ResNet50 also requires a significant amount of computational resources. • **Potential Overfitting:** Might overfit on small datasets.
CycleGAN	• **Unpaired Training:** No need for paired style and content images, making data collection easier. • **Bidirectional Transformation:** Can transform from one domain to another and then transform back.	• **Training Complexity:** The training process for CycleGAN is more complex and unstable • **Computational Resources:** Requires a large amount of computational resources due to its need of training separately on the entire set of style images and content images.

Table 6: Strength and Limitations of Different Style Transfer Models

5. Model Operations

5.1 Model Deployment

5.1.1 For Classification Model

We build a web interface with Streamlit for users to upload test images and plot the corresponding artists. To be more specific, we deploy the trained models into the web application and provide a feature so users can dynamically select one of the multiple

models to make a prediction. Here is the example of our interface and the classification example (fig21).

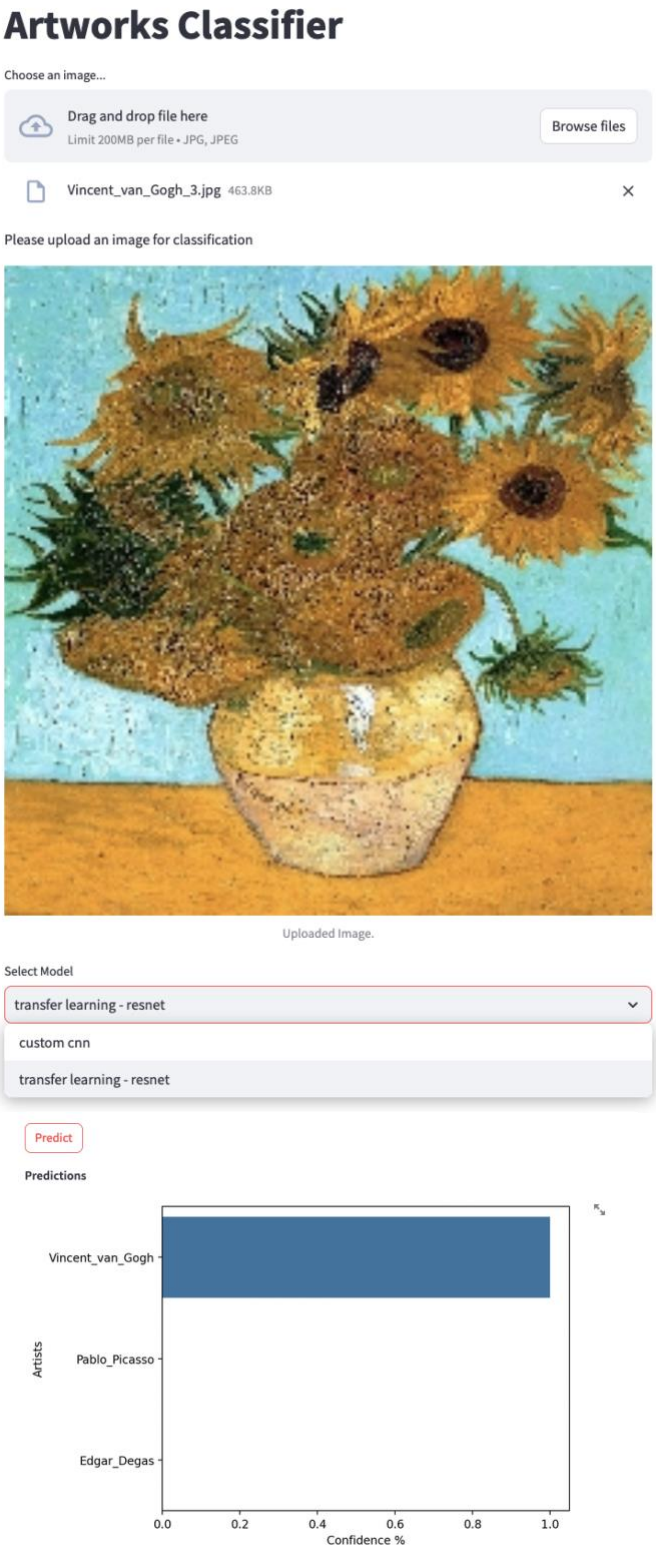


Fig 21. Web Interface for Classification

5.1.2 For Style Transfer Model

For the style transfer model based on VGG19/ResNet, we tried to build a web interface (fig22) where the user should be able to upload the base image and style image, while the interface will output the transferred image.

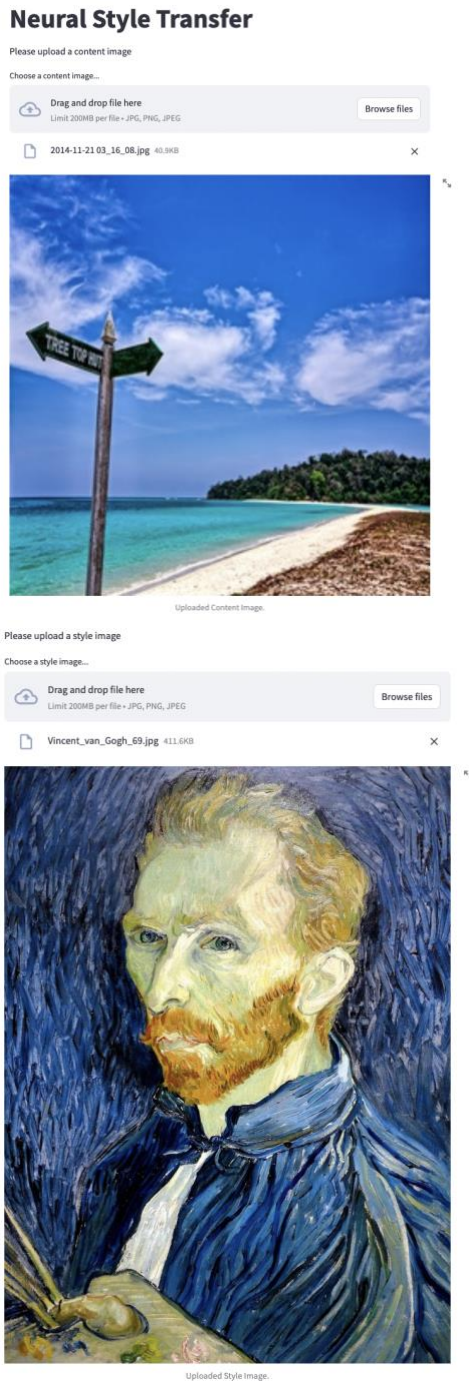
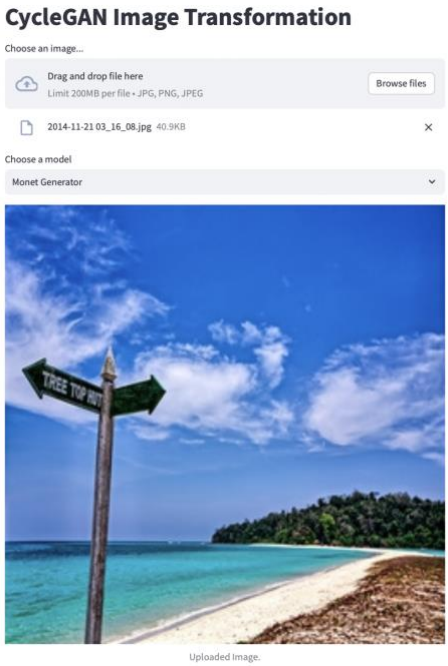


Fig 22. Web Interface for Neural Style Transfer

However, this process takes considerably long in practice since our Streamlit app is running on a CPU-only environment and the result has a relatively large loss as well. Here we show the interface. And some Possible improvements can be found in the 7.3 Appendix part.

For the style transfer model using CycleGAN, we built a web interface with Streamlit for users to upload a photo and generate the image in Monet's painting style (matching with the model training process). And here is the example (fig23) of our interface and the generated example. It can be seen that currently, the generated image has a relatively large loss, and further attempts should be made in order to optimize the result. The possible methods include hyper-parameter tuning, loss function redefining, or even trying a new architecture for the generator or discriminator.



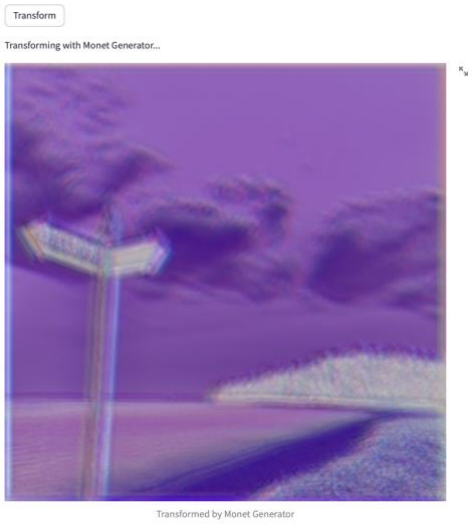


Fig 23. Web Interface for Monet Style Transfer Based on Cycle GAN

5.2 Model Maintenance

5.2.1 For Classification Model

- **Performance Monitoring:** Monitor the model's performance metrics, including accuracy and confusion matrices, to detect any drop in performance. If the model's accuracy degrades, retrain with the most recent data.
- **Hyperparameter Tuning:** Continue hyperparameter tuning to optimize the model's performance and adapt it to the changing data distribution. Use techniques like cross-validation and grid/random search for hyperparameter selection.
- **User Feedback and Monitoring:** Encourage user feedback to identify any potential misclassifications or inaccuracies in artist recognition. Regularly monitor user feedback to address issues promptly.

5.2.2 For Style Transfer Model

- **Performance Monitoring:** Monitor the model's performance metrics, such as visual quality, style fidelity, and content preservation, to detect any drop in performance. If the stylized images show degradation in quality, initiate retraining with the most recent data.
- **Hyperparameter Tuning:** Continue hyperparameter tuning to optimize the style transfer algorithm and achieve better stylization results. Experiment with different combinations of hyperparameters to find the best settings.
- **User Feedback and Monitoring:** Encourage user feedback to identify any potential issues or shortcomings in the stylization process. Regularly monitor user feedback to address issues promptly.

5.2.3 General

- **Data Monitoring:** Continuously monitor the incoming data to identify any changes or shifts in the distribution of artworks. Regularly check for new artists, updates in artistic styles, or potential biases in the data.
- **Data Collection and Retraining:** Periodically collect new data and add it to the training dataset. Regularly retrain the model using the updated dataset to ensure it remains up-to-date with the latest artistic styles and trends.
- **Model Versioning:** Maintain different versions of the model to ensure backward compatibility and easy rollback if required. Implement versioning to keep track of model updates and improvements.
- **A/B Testing:** Implement A/B testing for new model versions before deploying them to the production environment. Compare the performance of the new version against the existing model to ensure it meets or exceeds the desired accuracy/stylization quality.
- **Continuous Integration and Deployment (CI/CD):** Automate the deployment process using CI/CD pipelines to ensure seamless and frequent updates to the model. This reduces deployment downtime and allows for quick parameter updates.
- **Error Handling and Logging**: Implement robust error handling and logging mechanisms to detect and resolve any issues that may arise during model deployment or inference.

6. Conclusion

6.1 Findings

1) **Image Classification:** Our empirical results indicated a clear distinction in the performance of the two models employed. The transfer learning model rooted in the ResNet architecture outperformed the custom CNN model after hyperparameter tuning. This ResNet-based model achieved an accuracy rate of 86%, a commendable score in the context of our dataset.
2) **Style Transfer:** Two primary architectures, VGG19 and ResNet50, were employed for the style transfer task. Our observations highlighted that both models were adept at transferring styles successfully. A notable milestone was the convergence of both models after a span of 50 epochs, indicating the stability and effectiveness of our design and training procedures.
3) **CycleGAN for Monet-style Transformations:** Venturing into a more complex style transfer paradigm, our CycleGAN model is aimed at transforming general photos to mimic Claude Monet's painting style. The outcomes, however, were not entirely satisfactory. Potential avenues for improvement encompass hyperparameter tuning, refining the loss functions, or even pivoting to an entirely new architecture for either the generator or discriminator components.
4) **Web Interface Deployment:**

- The classification interface proved effective and efficient, delivering favorable results.

- The style transfer interface, although functional, encountered performance issues. Due to reliance on a CPU on our local setup, the computation was notably protracted. Additionally, the interface exhibited a relatively higher loss. Subsequent endeavors should focus on refining this interface, possibly exploring hardware-accelerated solutions or model optimizations to ameliorate these issues.

6.2 Next Steps

1) Continue to conduct hyperparameter tuning and explore more model architecture for both the classification and style transfer task.
2) Optimize the CycleGAN model for Monet-style Transformations, including hyper-parameter tuning, loss function redefining, or even trying a new architecture for the generator or discriminator.
3) Refine style-transfer interface, possibly exploring hardware-accelerated solutions or model optimizations to ameliorate loss issues in generated images.

6.3 Reference

[1] Main dataset: https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time
[2] Extra data for style transfer: https://www.kaggle.com/datasets/ayaderaghul/gan-getting-started-2
[3] Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A Neural Algorithm of Artistic Style. arXiv preprint.
[4] Luan, F., Paris, S., Shechtman, E., & Bala, K. (2017). Deep Photo Style Transfer.
[5] Dumoulin, V., Shlens, J., & Kudlur, M. (2017). A Learned Representation For Artistic Style. International Conference on Learning Representations (ICLR). Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
[6] Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. arXiv preprint.
[7] Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. Proceedings of the IEEE International Conference on Computer Vision (ICCV).
[8] Choi, Y., Choi, M., Kim, M., Ha, J. W., Kim, S., & Choo, J. (2017). Toward Multimodal Image-to-Image Translation. Advances in Neural Information Processing Systems (NIPS).