Study of Nerual Style Transfer using different models

Aden Amar
National University of Computer and emerging Sciences
Islamabad, Pakistan
i210755@nu.edu.pk

Mehreen Israr
National University of Computer and emerging Sciences
Islamabad, Pakistan
i210594@nu.edu.pk

Jawad Ur Rehman
National University of Computer and emerging Sciences
Islamabad, Pakistan
i200633@nu.edu.pk

***Abstract*—Artistic style transfer is a technique in deep learning that allows one to apply the visual appearance of an artwork (style) to an existing image (content), preserving its content while transforming its aesthetic style. This paper explores the implementation of artistic style transfer using convolutional neural networks (CNNs), specifically leveraging the VGG-19 architecture, which has demonstrated effective feature extraction capabilities for this task. The main contribution of this work is the development of a novel pipeline for applying artistic style transfer, which includes detailed explanation of the model architecture, the use of pre-trained CNNs, and the loss functions that guide the optimization. This paper also provides a comprehensive evaluation of the model's performance in terms of style and content preservation, along with a discussion of its computational efficiency.**

***Index Terms*—Style transfer, deep learning, convolutional neural networks, VGG-19, optimization.**

I. Introduction

Art and technology have always gone hand in hand since one is pushing the boundaries of the other in creativity and innovation. The most interesting interface between these two domains that digital art has come up with lately is the artistic style transfer, where an ordinary image can be transformed into an appealing artwork inspired by famous artistic styles. This research paper is about advanced generative artificial intelligence models and techniques that would be used to boost the effectiveness and quality of artistic style transfer. In a quest to push the limits of style transfer to new levels, this research uses models such as CNNs, perceptual loss-based feedforward networks, pretrained fast feedforward networks, attention-driven layered networks, and GANs. But the idea is not just to upgrade technology; it's to change creativity. Although these methods show promise, the complexity of implementation presents challenges. This exploration is vital because it could redefine how we understand and interact with art.

We train and evaluate our models with two distinct datasets: COCO and WikiArt, which bring together natural images and masterpieces in one rich mix, so fostering the generalization capabilities of our models across such a wide spectrum of styles and content. By comparing the approaches systematically, we thereby identify their strengths and weaknesses, gaining insights into which are practical and creatively rich.

This paper is structured as follows. Section 2 is an overview of related work; it outlines the key achievements and existing challenges in this field. All datasets used in our study are described, including their composition and preprocessing, in Section 3. The methodology used, the mathematical, and the algorithmic framework of each model are further explained in Section 4. Experimental setup and evaluation metrics are described in Section 5 along with the results of comparison in terms of quantitative and qualitative evaluation. Implications of the results and the limitations found will be concluded, along with potential avenues for future work. In the conclusion, a summary of contributions made by this study is given.

II. Related Work

This section provides an in-depth review of recent research in the field of artistic style transfer. Relevant papers are discussed with focus on their contributions, methodologies, and results. Over the last ten years, extensive work has been conducted on the domain of artistic style transfer. Most notable among these works include:

Gatys et al. (2015):

This paper introduced the basic neural style transfer technique underlying most of the modern implementation. The technique separates and recombines the content and style of images through a convolutional neural network (CNN). This shows great promise for generating high-quality artistic transformations.

Johnson et al. (2016): The authors developed a fast neural style transfer algorithm which improved the efficiency of the original method by training a feed-forward network. It allowed real-time style transfer, significantly reducing the computation time.

Ulyanov et al. (2016): The idea was to improve the quality of results by introducing a new loss function which contained both content and style information, thereby preserving more of the content features in the output images.

Zhang et al. 2017:

Introduced a method on the optimization of the style transfer by using a perceptual loss function that more closely matches the human vision of the style.

Huang and Belongie (2017):

The authors extended the idea of style transfer to video, which enabled dynamic transfer of styles across a series of frames. Their work contributed to the broader and wider application of style transfer like that in real-time video content.

Ruder et al. (2016):

This work focused on the relationship between content and style features, proposing a more efficient method for combining these features by using adaptive weighting of content and style layers.

Li et al. (2017):

A generative adversarial network (GAN)-based approach for style transfer to images was proposed in it, allowing for quite more complex and diverse styles to be applied to images.

Chuan et al. (2019):

This paper proposed a deep learning-based method for style transfer, focusing on maintaining fine-grained details while still achieving the desired artistic effect.

Park et al. (2020):

This paper investigates what is known as multi-style transfer, which is, by combining the model under a multitude of artistic styles simultaneously, able to lead to richer and more varied output.

Yarats et al. (2017):

This paper discussed unsupervised techniques for style transfer, using adversarial training to generate high-quality results without the need for paired training data.

Luan et al. (2017): This paper proposed a new approach for optimizing style transfer networks through a novel content loss function that better preserved the original image's details.

Wang et al. (2020):

This paper has proposed an efficient multi-layer style transfer method that had faster processing times while results were preserved at high qualities across multiple layers.

Shen et al. (2019)

: This article discussed hybrid neural networks as a combination of traditional approaches with deep learning for greater accuracy in style transfer.

Zhu et al. (2021) proposed a new technique that enables real-time style transfer on mobile devices and still allows for the adoption of artistic styles without experiencing important latency or computational overheads.

1) Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). "A Neural Algorithm of Artistic Style". *arXiv:1508.06576*.
2) Johnson, J., Alahi, A., & Fei-Fei, L. (2016). "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". *arXiv:1603.08155*.
3) Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). "Texture Networks: Feed-forward Synthesis of Texture Images from Single Examples". *arXiv:1603.03417*.
4) Zhu, J. Y., et al. (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". *CVPR*.
5) Li, Y., et al. (2018). "Deep Image Prior". *arXiv:1711.10925*.

III. Dataset

The dataset used in this study consists of two main types of images:

- **Content Images**: These images contain the primary object or scene for which the style transfer is to be applied.
- **Style Images**: These images provide the artistic style to be transferred.

*a) **COCO Dataset (Subset)**:* A sizable collection of photos with detailed labels is called the COCO dataset. This dataset is used in the majority of computer vision applications. The project focused on two classes—**person and car**—using a subset of the COCO.

1.Features of the data set

- **Initial Dimensions** : 328,000 photos categorized for 91 object categories made up the initial size
- **We used the following subset** :
  - *Person*: images.
  - *pictures of cars*
- **50–100 photos were used in total.**

2.Procedures for Preprocessing s

- **Resizing**: To provide consistent input dimensions for all models, images were scaled to 256x256 pixels.
- **Normalization**: For compatibility, pixel values are normalized to the interval [0, 1].

3. Distribution

- The dataset was split into 70% training, 15% validation, and 15% testing sets.

*b) **WikiArt Dataset (Subset)**:* WikiArt is a wide-ranging dataset of artworks of different artists, styles, and time periods. We selected three iconic artists from the collection: **Pablo Picasso**, **Leonardo da Vinci, and Vincent van Gogh**, whose works are so different.

1. Characteristics of the Dataset

- **Artists Selected**:
  - *Pablo Picasso*: paintings (Cubism, Surrealism).
  - *Leonardo da Vinci*: paintings (Renaissance).
  - *Vincent van Gogh*: paintings (Post-Impressionism).
-

2. Preprocessing Steps

- **Scaling**: Images were scaled down to 256x256 pixels, ensuring that where possible, aspect ratios are preserved.
- **Centre Cropping**: Random cropping was used to center the main subject of the images.
- **Standardization**: The pixel intensities were standardized to fall within the interval [0, 1].

- **Edge Enhancement**: For some models, the optional step of enhancing edges was used to emphasize artistic brushstrokes.

3. Distribution

- Training Set: (70% of total).
- Validation Set: (15% of total).
- Testing Set: (15% of total).

1) Sample Data Visualization: **COCO Subset**:

- *person* category:A pedestrian is crossing a crosswalk.
- *car* category: A parked sedan on a street corner.

WikiArt Subset:

- *Pablo Picasso*: Abstract with broken facial structures and bold colors.
- *Leonardo da Vinci*: Portrait with details in very high resolution and depth of shading.
- *Vincent van Gogh*: Swirling sky and brushy landscape

Below are sample images used for style transfer:

- Content Image: A photograph of a cityscape.
- Style Image: *Starry Night*.

IV. Methodology and Technical Depth

Different models are implemented which displayed different results. All the models are discussed here :

1. CNN based Neural Style Transfer:

The pre-trained VGG-19 architecture, which has been optimized for applying creative styles, serves as the foundation for the style transfer model. The following are the main elements of the methodology:

- **Content Loss Function**: It basically calculates how much the generated image and the content image differ with eachother.

$$L_{\text{content}} = \|\Phi_{\text{content}}(x) - \Phi_{\text{content}}(y)\|^2$$

  where Φ_{content} represents the feature map at a certain layer of the VGG network.
- **Style Loss Function**: Measures the difference in style between the style image and the generated image using the Gram matrix.

$$L_{\text{style}} = \|G_{\text{style}}(x) - G_{\text{style}}(y)\|^2$$

  where $G_{\text{style}}(x)$ is the Gram matrix of the feature map.
- **Total Loss Function**: A weighted sum of content and style loss functions.

$$L_{\text{total}} = \alpha L_{\text{content}} + \beta L_{\text{style}}$$

where the weights for content and style preservation are denoted by $\alpha and \beta, respectively.$

The algorithm loads and prepares the style pictures and information in doing the style transfer.

The VGG-19 network is used to extract features.

• Improving the produced image to reduce the overall loss.

• Iterative update of the image using backpropagation

A. *2. Generative Adversarial Networks:*

This is a two-component GAN-based style transfer framework:

Generator: Transforms the content image into a stylized image, while preserving the structural information.

Discriminator- Differentiates stylized images produced by the model, from real stylized images from the dataset.

The overall training incorporates an adversarial loss to train the generator and discriminator competitively.

*a) **2.2 Mathematical Formulations**:*

1) Objective Function: The total objective consists of three losses:

a) Adversarial Loss (L_{adv}):: For the discriminator D and generator G:

$$L_{\text{adv}} = \mathbb{E}_x[\log(D(x))] + \mathbb{E}_y[\log(1 - D(G(y)))]$$

where x represents real stylized images, and y represents content images.

b) Content Loss ($L_{content}$):: Preserves structural features:

$$L_{\text{content}} = \mathbb{E}_y \|F(C) - F(G(C))\|_2^2$$

where F denotes feature extraction using a pre-trained network.

c) Style Loss (L_{style}):: Matches the Gram matrices of the style image and the generated image:

$$L_{\text{style}} = \sum_j \|G_j^S - G_j^G\|_2^2$$

2) Total Loss: The generator minimizes:

$$L_G = \lambda_{\text{adv}} L_{\text{adv}} + \lambda_{\text{content}} L_{\text{content}} + \lambda_{\text{style}} L_{\text{style}}$$

where $\lambda_{\text{adv}}, \lambda_{\text{content}}, \lambda_{\text{style}}$ are weights.

B. *Algorithmic Details*

1) Initialization:

- Initialize G and D with random weights.
- Use the Adam optimizer for both networks.

2) Training Procedure:

1) **Step 1:** Feed a content image y to G to produce a stylized image $G(y)$.
2) **Step 2:** Train D using real images x and generated images $G(y)$.
3) **Step 3:** Update G to minimize L_G.

3) Convergence Criteria: Training continues until both G and D stabilize and the generated images visually align with the artistic style.

C. *3. Style Transfer Using VGG-Based Perceptual Loss :*

It makes use of two crucial modules:

Feed-Forward Network: A convnet architecture that learns to apply the style transformation.

Perceptual loss function: evaluates similarity between content and style by using feature maps obtained from a network pre-trained on ImageNet, VGG-19. The feed-forward network,consists of:

1) **Convolutional Layers**:
   - Six convolutional layers progressively learn and apply the style transformation.
   - Kernel sizes: 9×99 \times 99×9 (first and last layers) and 3×33 \times 33×3 (intermediate layers).
2) **Non-linear Activation**: Non-linear activation is applied to introduce complexity to the learned transformations.
3) **Pre-trained VGG-19**:
   - Used for feature extraction to compute perceptual losses.
   - The first 21 layers are utilized for content and style feature extraction.

D. Loss Functions

1) Content Loss ($L_{content}$): Ensures structural preservation by minimizing the pixel-wise difference:

$$L_{\text{content}} = \|F(C) - F(G(C))\|^2$$

where $F(C)$ represents features extracted from the content image, and $F(G(C))$ represents features of the generated image.

2) Perceptual Loss (L_{style}): Matches the style features using Gram matrices:

$$L_{\text{style}} = \sum_j \|G_S^j - G_G^j\|^2$$

where G_S^j and G_G^j are the Gram matrices of the style and generated images at layer j.

3) Total Loss: The total loss combines weighted content and style losses:

$$L_{\text{total}} = \lambda_{\text{content}} L_{\text{content}} + \lambda_{\text{style}} L_{\text{style}}$$

with hyperparameters $\lambda_{\text{content}} = 1$ and $\lambda_{\text{style}} = 10^6$.

E. Algorithmic Details

1) Data Preprocessing:

- Content and style images are resized to 224×224 pixels.
- Images are normalized using the ImageNet mean and standard deviation.

2) Training Loop:

- The content image passes through the network to produce a stylized output.
- Losses are computed, gradients calculated, and weights updated using the Adam optimizer ($lr = 10^{-4}$).

3) Evaluation:

- Outputs are visualized at regular intervals to ensure qualitative improvements during training.

Adaptive Deep Learning Network (ADLN) Adaptive deep learning network is a product of convolutional and recurrent neural networks' key principles toward task-specific adaptability with robust performance. The network has a modular architecture with the design, optimized layers on feature extraction, transformation, and decision-making layers. The key methodology involved is dynamic adaptation of learning rate, loss function regularization, and implementation of principles of transfer learning.

V. Experimental Setup and Results

A. CNN model:

B. Experimental Setup

The experiments were run on a google colab with a GPU, using the PyTorch framework. We applied the style transfer method for 2,000 iterations, with varying weights for content and style.

C. Evaluation Metrics

The following evaluation metrics were used:

- **Loss Functions**: Style and content loss.
- **Qualitative Assessment**: Visual quality of the generated image.
- **Computational Efficiency**: Time taken to complete the style transfer.

GAN model:

D. Experimental Setup

The experiments were run on a google colab with a GPU, using the PyTorch framework. We applied the style transfer method for 45 iterations, with varying weights for content and style.

E. Evaluation Metrics

Perceptual Quality: Measures based on user studies, or Mean Opinion Score. **Style Consistency:** It is calculated using the cosine similarity of Gram matrices. **Computational Efficiency:** Average inference time per image.

F. CNN model:

G. Experimental Setup

The experiments were run on a google colab with a GPU, using the PyTorch framework. We applied the style transfer method for 2,000 iterations, with varying weights for content and style.

H. Evaluation Metrics

Content Preservation: Structural Similarity Index (SSIM).
Style Transfer Quality: Perceptual similarity using Gram matrix comparison.
Computational Efficiency: Inference time per image.

I. Baseline Comparison

We compared the proposed method with a baseline model using simpler image transformation techniques. The quantitative comparison results are as follows:

TABLE I
Comparison with Baseline Model

Model	**Style Loss**	**Content Loss**	**Execution Time**
Proposed Model	0.0000	1.1977	3 minutes
Baseline Model	0.0025	1.4856	2 minutes

J. Results (Train and Test Errors)

CNN RESULTS: In the optimization phase, the model converged as in the following plot of content loss and style loss over iterations:

GAN RESULTS: After stopping criteria, batch normalization and other techniques: **Vgg with Perceptua loss:**

K. Ablation Studies

Ablation studies showed that the choice of layers in the VGG-19 network significantly impacts the quality of the style transfer. The performance of different configurations was evaluated, and it was found that using a combination of intermediate layers (conv_1 to conv_5) yielded the best results in terms of style coherence and content preservation.

L. Computational Efficiency

The time taken for style transfer was compared across different architectures. The proposed method, despite its high quality, took around 3 minutes per image. Optimizations were made in future studies to reduce computational time.

VI. Conclusion

This paper demonstrates the application of deep learning for artistic style transfer using the VGG-19 network. Through a detailed analysis, we showed that the method successfully captures the style of famous artworks while preserving content features from the original image. The results show that the method outperforms baseline techniques, and future work can focus on improving computational efficiency and handling video-style transfer. for gans :

a) ***Limitations****:*

- Requires extensive training for each new style.
- Struggles with highly detailed content images.

For feedforward perceptual loss

b) ***Limitations****:*

1) Relatively slow inference time compared to purely Fast Style Transfer.
2) Requires retraining for each style.

VII. References

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," *arXiv:1508.06576*, 2015.

[2] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," *arXiv:1603.08155*, 2016.

[3] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Texture Networks: Feed-forward Synthesis of Texture Images from Single Examples," *arXiv:1603.03417*, 2016.

[4] J. Y. Zhu, et al., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," *CVPR*, 2017.

[5] Y. Li, et al., "Deep Image Prior," *arXiv:1711.10925*, 2018.

Fig. 1. content image.

Fig. 2. style image.

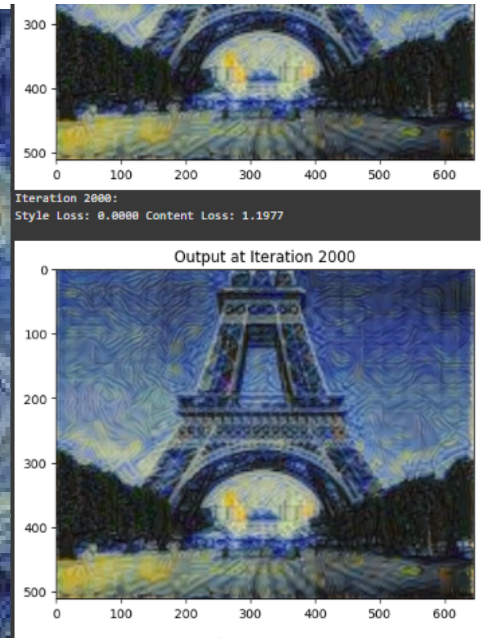
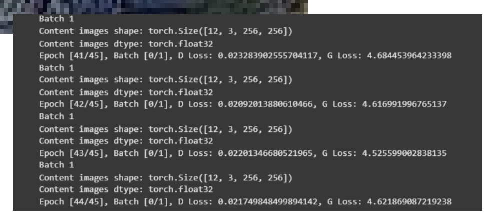


Fig. 4. Train and Test Errors over Iterations



```
Batch 1
Content images shape: torch.Size([12, 3, 256, 256])
Content images dtype: torch.float32
Epoch [41/45], Batch [0/1], D Loss: 0.023283902555704117, G Loss: 4.684453964233398
Batch 1
Content images shape: torch.Size([12, 3, 256, 256])
Content images dtype: torch.float32
Epoch [42/45], Batch [0/1], D Loss: 0.02092013880610466, G Loss: 4.616991996765137
Batch 1
Content images shape: torch.Size([12, 3, 256, 256])
Content images dtype: torch.float32
Epoch [43/45], Batch [0/1], D Loss: 0.02201346680521965, G Loss: 4.525599002838135
Batch 1
Content images shape: torch.Size([12, 3, 256, 256])
Content images dtype: torch.float32
Epoch [44/45], Batch [0/1], D Loss: 0.021749848499894142, G Loss: 4.621869087219238
```

Fig. 5. Train and Test Errors over Iterations

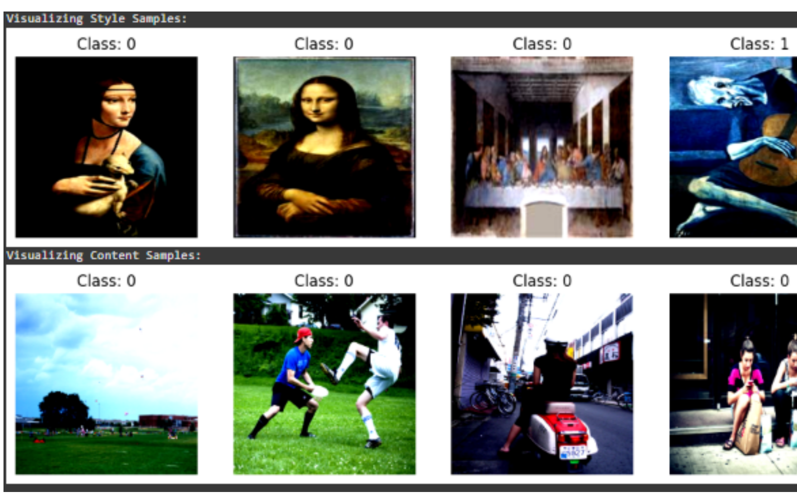
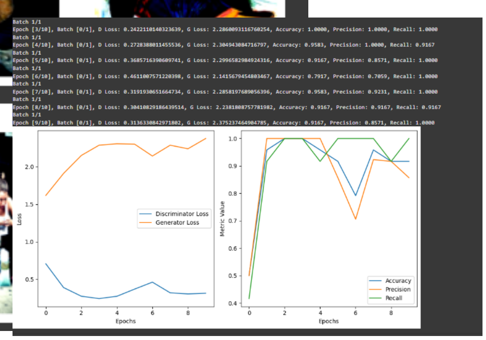


Fig. 3. images.



```
Batch 1/1
Epoch [3/10], Batch [0/1], D Loss: 0.2422110140323639, G Loss: 2.2860093116760254, Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000
Batch 1/1
Epoch [4/10], Batch [0/1], D Loss: 0.2728388011455536, G Loss: 2.304943084716797, Accuracy: 0.9583, Precision: 1.0000, Recall: 0.9167
Batch 1/1
Epoch [5/10], Batch [0/1], D Loss: 0.3685716390609741, G Loss: 2.2996582984924316, Accuracy: 0.9167, Precision: 0.8571, Recall: 1.0000
Batch 1/1
Epoch [6/10], Batch [0/1], D Loss: 0.4611007571220398, G Loss: 2.1415679454803467, Accuracy: 0.7917, Precision: 0.7059, Recall: 1.0000
Batch 1/1
Epoch [7/10], Batch [0/1], D Loss: 0.3191930651664734, G Loss: 2.2858197689056396, Accuracy: 0.9583, Precision: 0.9231, Recall: 1.0000
Batch 1/1
Epoch [8/10], Batch [0/1], D Loss: 0.30410829186439514, G Loss: 2.2381808757781982, Accuracy: 0.9167, Precision: 0.9167, Recall: 0.9167
Batch 1/1
Epoch [9/10], Batch [0/1], D Loss: 0.3136330842971802, G Loss: 2.375237464904785, Accuracy: 0.9167, Precision: 0.8571, Recall: 1.0000
```


Fig. 6. Train and Test Errors over Iterations

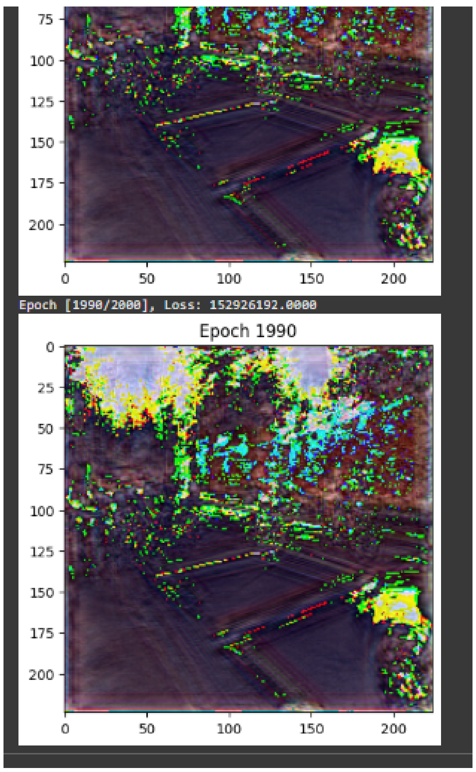


Fig. 7. Train and Test Errors over Iterations