Robust Style Transfer with Transformers

Chaitanya Tatipigari
ctatipigari3@gatech.edu

Alec Kain
akain3@gatech.edu

Tyler J. Church
tchurch8@gatech.edu

Georgia Institute of Technology
Atlanta, GA

Abstract

We explore the robustness of transformer-based neural style transfer under challenging style scenarios, such as abstract art, where the distinction between content and style becomes ambiguous. Building on the StyTr^2 architecture [1], we propose two novel loss functions—$\mathcal{L}_{sep1}$ and $\mathcal{L}_{sep2}$—that explicitly enforce separability between content and style features. Our approach improves stylization consistency and reduces noise artifacts by encouraging representations to occupy distinct subspaces. We validate our method on a combination of abstract and natural image datasets and find that the $\mathcal{L}_{sep1}$ model achieves superior content preservation and faster convergence, while $\mathcal{L}_{sep2}$ demonstrates increased resilience to stylization noise. Quantitative metrics and qualitative comparisons highlight the effectiveness of our loss formulations. Our modifications maintain or surpass the original StyTr^2 performance, showing that explicitly modeling content–style separability enhances robustness in difficult style transfer tasks. All code is available at https://github.com/Chai-T29/StyleTransfer_ViT.

1. Introduction

1.1. Background

Neural style transfer generates images that combine the content of one image with the artistic style of another. Gatys et al. showed that feature activations from a pretrained CNN can define content and style losses—content from deep feature maps and style from Gram matrix statistics [3]. Johnson et al. introduced feed-forward networks for real-time stylization using perceptual loss [5], and Ulyanov et al. improved convergence and quality via instance normalization [9]. AdaIN, proposed by Huang and Belongie, enabled arbitrary style transfer by aligning channel statistics [4], while Li et al. used feature whitening and coloring transforms for universal stylization [7]. More recently, Deng et al. leveraged Vision Transformers in their StyTr^2 model to set a new state of the art [1], following the Transformer framework from Vaswani et al. [10].

Style transfer has broad applications. Designers and artists can quickly prototype visuals, while AR/VR systems can apply real-time stylization in immersive environments. In photography and film, it can automate color grading and effects. Content–style disentanglement techniques also inspire cross-modal tasks in NLP (e.g., text style transfer) and speech synthesis (e.g., voice style conversion).

1.2. Motivation

This work investigates whether the original StyTr^2 losses [1] adequately distinguish content and style in challenging scenarios—especially when style is abstract or subtle. If not, can a new loss term reinforce this separation?

Building on StyTr^2, we identified two limitations: poor robustness to abstract styles and visible noise artifacts in output. To address these, we introduce two novel loss functions—$\mathcal{L}_{sep1}$ and $\mathcal{L}_{sep2}$—that explicitly enforce content–style separability.

1.3. Data

We used two Kaggle datasets to evaluate our modified StyTr^2 model. The "Random Image Sample Dataset" (`pankajkumar2002`) offers $3,000$ content images (150×150), while "Abstract Art Images" (`greg115`) provides $8,145$ abstract style images (512×512). These datasets are publicly available, diverse, and high-quality, making them well-suited for assessing generalization and stylization fidelity.

2. Approach

2.1. Method Overview

To overcome StyTr^2's limitations [1], we introduce two separability loss variants, $\mathcal{L}_{sep1}$ (10) and $\mathcal{L}_{sep2}$ (11), designed to explicitly encourage separation between content and style features. $\mathcal{L}_{sep1}$ focuses on content distinctions, while $\mathcal{L}_{sep2}$ targets both content and style. Unlike StyTr^2,

which assumes orthogonal representations of content and style, our losses enforce this during training by pushing style and content features into distinct subspaces. This improves stylization consistency, especially for abstract styles.

2.2. Architecture Details

We use the StyTr^2 architecture [1], with changes to the loss functions. Figure 1 shows the original pipeline.

2.2.1 Image Patching and Feature Extraction

The content image I_c and style image I_s are split into $P \times P$ non-overlapping patches, yielding $N = H/P \times W/P$ patches [2]. Each patch x_p is projected to a D-dimensional embedding using learned matrices E^c and E^s:

$$z_p^c = E^c \cdot x_p^c + e_p^{CAPE}, \quad z_p^s = E^s \cdot x_p^s \tag{1}$$

e_p^{CAPE} is a content-aware positional embedding [2], computed as:

$$\begin{aligned} P_L &= F_{pos}(AvgPool_{n\times n}(E)), \\ e_p^{CAPE}(x,y) &= \sum_{k,l=0}^{s} a_{kl} P_L(x_k, y_l) \end{aligned} \tag{2}$$

with $n = 18$ as in [1], bilinear interpolation for a_{kl}, and F_{pos} as a learnable position embedding layer.

2.2.2 Transformer Encoders

Just as in StyTr^2 [1] two parallel transformer encoders process content and style embeddings ($\{z_p^c\}_{p=1}^N$ and $\{z_p^s\}_{p=1}^N$) to produce embeddings ($\{h_p^c\}_{p=1}^N$ and $\{h_p^s\}_{p=1}^N$):

$$\begin{aligned} h_p^l &= \text{MSA}(\text{LN}(h_p^{l-1})) + h_p^{l-1}, \\ h_p^l &= \text{FFN}(\text{LN}(h_p^l)) + h_p^l \end{aligned} \tag{3}$$

where MSA is the multi-head scaled dot-product attention [10], LN is layer normalization, and FFN is a feedforward network. The scaled dot-product attention can be defined as:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

2.2.3 Transformer Decoder

The decoder mirrors StyTr^2 [1], using content embeddings as queries and style embeddings as memory:

$$\begin{aligned} q_p^l &= \text{MSA}(\text{LN}(q_p^{l-1})) + q_p^{l-1} \\ q_p^l &= \text{CrossAttention}(\text{LN}(q_p^l), M^s, M^s) + q_p^l \\ q_p^l &= \text{FFN}(\text{LN}(q_p^l)) + q_p^l \end{aligned} \tag{5}$$

where $M^s = \{h_1^s, h_2^s, ..., h_N^s\}$ is the style memory and $q_p^0 = h_p^c + e_p^{CAPE}$. Final outputs q_p^L are projected back to pixel space to form I_o.

2.3. Loss Functions

A core component of the loss functions is the VGG feature extractor ϕ, which allows the model to directly compare content and style of images. By extracting features at deeper layers, the model learns content representations. Conversely, shallow layers tend to contain low-level features like style.

2.3.1 Content and Style Losses

Following StyTr^2 [1], content loss compares VGG19 features of I_o and I_c across the content-focused ϕ layers C_l:

$$\mathcal{L}_c = \sum_{i \in C_l} ||\phi_i(I_o) - \phi_i(I_c)||_2^2 \tag{6}$$

Style loss compares per-channel mean and standard deviation statistics across the style-focused ϕ layers S_l:

$$\begin{aligned} \mathcal{L}_s = \sum_{i \in S_l} \Big(&\|\mu(\phi_i(I_o)) - \mu(\phi_i(I_s))\|_2^2 \\ &+ \|\sigma(\phi_i(I_o)) - \sigma(\phi_i(I_s))\|_2^2 \Big) \end{aligned} \tag{7}$$

2.3.2 Identity Losses

We employ two identity losses as defined in StyTr^2 [1] to ensure consistency in edge cases:

$$\mathcal{L}_{id1} = ||I_{cc} - I_c||_2^2 + ||I_{ss} - I_s||_2^2 \tag{8}$$

$$\mathcal{L}_{id2} = \|\phi(I_{cc}) - \phi(I_c)\|_2^2 + \|\phi(I_{ss}) - \phi(I_s)\|_2^2 \tag{9}$$

where I_{cc} is the output when the content image is fed in as both the content and style image, and I_{ss} is the output when the style image is fed in as both the content and style image. These losses ensure that both the style and content remain unchanged in I_{cc} and I_{ss}, which allows the model to distinguish between style and content.

2.3.3 Separability Losses

Our key contribution is the introduction of separability losses that explicitly enforce separation between content and style representations. We propose two variants to compare their efficacy:

Content Separability Loss ($\mathcal{L}_{sep1}$):

$$\mathcal{L}_{sep1} = \lambda_{sep} \cdot \sum_{i=1}^{N_l} R\Big(m^2 - \sum_{f=1}^{F} (\phi_i(I_o)_f - \phi_i(I_r)_f)^2\Big) \tag{10}$$

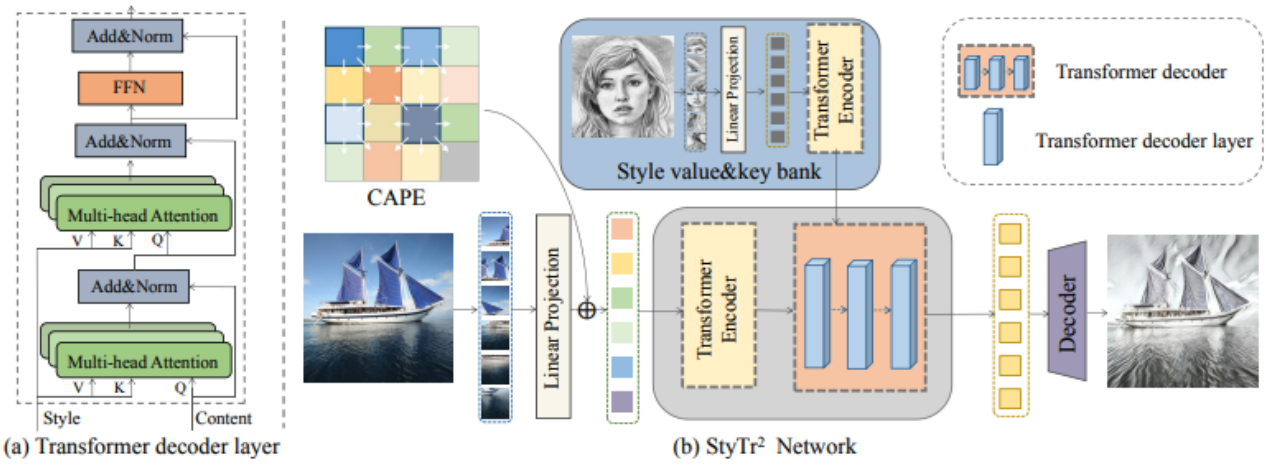

Figure 1. Overview of the StyleTr^2 architecture, as shown in [1]. The figure highlights the transformer encoders for content and style, along with the transformer decoder that processes the style memory with content queries.

Full Separability Loss ($\mathcal{L}_{sep2}$):

$$\mathcal{L}_{sep2} = \lambda_{sep} \cdot \sum_{i=1}^{N_l} \Big(R\Big(m^2 - \sum_{f=1}^{F} (\phi_i(I_o)_f - \phi_i(I_r)_f)^2 \Big) + R\Big(m^2 - \sum_{f=1}^{F} \Big((\mu(I_o)_f - \mu(I_r)_f)^2 - (\sigma(I_o)_f - \sigma(I_r)_f)^2 \Big) \Big) \Big) \tag{11}$$

where R is the ReLU activation function, m is the margin parameter, F is the length of the flattened feature dimension, ϕ_i is the VGG19 feature extractor at the i-th layer, μ and σ represent the mean and standard deviation operations, and I_r is the output of the network when the style and content images are switched in the forward pass.

The intuition behind these separability losses is to enforce a minimum distance (margin m) between the feature representations of I_o and I_r image. If the distance is less than m, the loss penalizes the model, pushing it to increase the separation. If the distance exceeds m, the ReLU function zeros out the loss term.

2.3.4 Total Loss

The total loss is a weighted combination of all the individual loss terms:

$$\mathcal{L}_{total} = \lambda_c \mathcal{L}_c + \lambda_s \mathcal{L}_s + \lambda_{id1} \mathcal{L}_{id1} + \lambda_{id2} \mathcal{L}_{id2} + \lambda_{sep} \mathcal{L}_{sep} \tag{12}$$

where λ_c, λ_s, λ_{id1}, λ_{id2}, and λ_{sep} are hyperparameters that control the relative importance of each loss term. In our implementation, we typically use $\lambda_c = 10$, $\lambda_s = 7$, $\lambda_{id1} = 70$, $\lambda_{id2} = 1$, and $\lambda_{sep} = 10$ for $\mathcal{L}_{sep1}$ or $\lambda_{sep} = 1$ for $\mathcal{L}_{sep2}$.

For the total loss, the separability loss $\mathcal{L}_{sep}$ refers to either $\mathcal{L}_{sep1}$ or $\mathcal{L}_{sep2}$ depending on the experiment, as we aim to compare the efficacy of each of these loss functions.

2.4. Technical Challenges

We anticipated several technical challenges in our approach:

Loss Function Balancing. We expected that integrating new loss terms would require careful balancing of the loss weights (λ values) to ensure stable training. This proved even more challenging than anticipated, particularly for the $\mathcal{L}_{sep2}$ loss which required significantly lower weights than other loss components to avoid convergence issues.

Margin Parameter Sensitivity. While we knew the margin parameter m in our separability losses would be important, we underestimated its impact. Our experiments revealed extreme sensitivity to this parameter, with small variations in m producing dramatically different stylization results. Finding optimal values required extensive grid searches and careful qualitative assessment.

Abstract Style Handling. We anticipated difficulties in handling highly abstract style images but encountered more substantial challenges than expected. Many abstract styles in our dataset contained extreme color distributions and unconventional patterns that initially led to training instability and poor generalization.

Convergence Speed. We encountered unexpected differences in convergence speeds between models. The $\mathcal{L}_{sep1}$ model converged significantly faster than both the baseline

and $\mathcal{L}_{sep2}$ models, which was a surprising but welcome discovery.

Our first implementation attempts focused on implementing the separability losses effectively. After thorough experimentation, we discovered that the balance between content preservation and style transfer was highly sensitive to both the margin parameter and the weight applied to the separability loss terms.

Another early challenge was finding the right architecture for feature extraction. We initially experimented with modifying the VGG19 layers used for feature extraction, but found that the layer selections from the original StyTr^2 paper were indeed optimal for our task.

2.5. Implementation Details

Our implementation builds upon the official StyTr^2 codebase [1], with several key modifications:

1. We restructured the codebase using PyTorch Lightning to improve reproducibility and enable efficient multi-GPU training when available.

2. We implemented our novel separability loss functions ($\mathcal{L}_{sep1}$ and $\mathcal{L}_{sep2}$) as modular components easily integrated into the existing loss calculation pipeline.

3. We modified the feature extraction process to use the standard PyTorch pretrained VGG19 model [8] rather than the custom-trained version in the original implementation, which improved reproducibility and reduced training overhead. All VGG layers are frozen during training.

4. We implemented an extensive hyperparameter tuning framework to systematically explore the impact of different loss weights, margin values, and learning rate schedules.

5. We enhanced the data loading pipeline to handle our diverse datasets of content and abstract style images more efficiently.

The core transformer architecture remains similar to the original StyTr^2 implementation, but we simplified certain components to focus computational resources on our loss function experiments. Our code and trained models will be made publicly available upon publication to support reproducibility and future research. All of our experiments were run in Google Colab, but can be easily replicated on any setup with sufficient RAM capacity and a hardware accelerator. The models were trained on a single A100 GPU, but our implementation supports multi-GPU training and provides support for other hardware accelerators with the help of PyTorch Lightning.

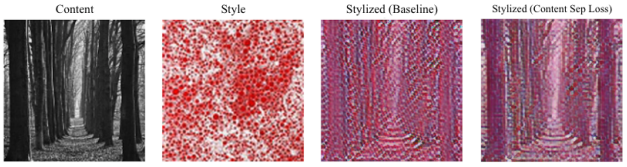

Figure 2. Comparison of baseline results and a stylization experiment with $\mathcal{L}_{sep}$. Both experiments yielded highly similar $\mathcal{L}_c$ and $\mathcal{L}_s$ but visibly different stylized images.

3. Experiments and Results

3.1. Experimentation Overview

Style transfer is an unsupervised learning task, which makes performance monitoring inherently subjective. As a result, the simplest and most convenient form of performance assessment is a combination of both quantitative and qualitative measures. Much like the original StyTr^2 paper [1], we measure $\mathcal{L}_c$ and $\mathcal{L}_s$ for quantitative evaluation, and visually compare the stylized images for a qualitative evaluation.

Since the style dataset we used is extremely abstract in nature, both qualitative and quantitative evaluations are crucial in assessing model performance. With the new loss function $\mathcal{L}_{sep}$, our model aims to separate both the style and content of I_o and I_r by margin m. This drives the model to distinctly separate style and content for both I_c and I_s. This distinction may yield an optimal I_o, with unexpected changes in $\mathcal{L}_c$ and $\mathcal{L}_s$. Quantitatively, the results may appear to be the same, but qualitatively we observe subtle differences that each model parameter can make. Figure 2 demonstrates an example that yielded similar results quantitatively, but different results qualitatively.

3.2. Hyperparameters

As a baseline comparison, we recreated the StyTr^2 paper [1] with the exact same hyperparameters and training procedure. The optimal hyperparameter settings from our baseline model were used as a starting point of experimentation for both of our custom $\mathcal{L}_{sep}$ implementations.

3.2.1 Data

The models were trained on a set of 3000 content images, each paired with 10 randomly selected style images, yielding a total dataset of $30,000$ images. The batch size was set to 8 as done in the original paper, as this yielded the most stable convergence. The batches are fed in as a dictionary with a key for content and style image tensors.

3.2.2 VGG19

To extract features for the loss function, N_l number of hidden states were selected to extract hidden states h_l, where N_l are the number of selected layers and $h_l = \phi(I)$. In our experiments, N_l was set to 6, each h_l was extracted after a ReLU activation, and the selection of layers l_i was tuned for optimality. The intuition is that deeper layers in the VGG19 model are capable of extracting complex shapes, which are more representative of the content. Conversely, shallow layers are able to capture textures and low-level features, which are more representative of style. Optimizing the depth of feature extraction from the VGG has a direct relationship to the style and content retention of the model. For our experiments, we chose $\{l_1, ..., l_6\}$ to be $\{8, 15, 20, 26, 31, 35\}$ as this was shown to balance $\mathcal{L}_c$ and $\mathcal{L}_s$.

3.2.3 Loss

Base Loss. The base loss terms $\mathcal{L}_c$, $\mathcal{L}_s$, $\mathcal{L}_{id1}$, and $\mathcal{L}_{id2}$ were set to the optimal values from the original StyTr^2 experiments (10, 7, 70, 1), to enable a controlled experiment.

Separability Loss. In our experiments, we observed that in our $\mathcal{L}_{sep1}$ model that that increasing m beyond a threshold yielded no significant change. So, for our $\mathcal{L}_{sep1}$ experiments, we set $\lambda_{sep} = 10$ and $m = 2$. With these settings, $\mathcal{L}_{total}$ converged to an optimal solution much quicker than the baseline StyTr^2 architecture and even showed slightly improved results.

When tuning for $\mathcal{L}_{sep2}$, we found that there was a high sensitivity of this model to m because of the μ and σ terms. When we lower $m = 2$, the loss signal quickly disappears, with $m = 3$, the signal remains, though less frequent as the model learns to distinguish I_o's and I_r's respective styles and contents. However, increasing m to 4 and higher yields extreme results, as shown in Figure 3. Even at $m = 3$, λ_{sep} has to be tuned down to 1 to ensure the model achieves stable convergence. One interesting observation is that both $\mathcal{L}_c$ and $\mathcal{L}_s$ remain within optimal ranges despite erratic stylization at higher margins, showcasing the need for careful qualitative assessment as well.

L1 vs. L2 Loss Comparison. One of the key aspects we wanted to explore in our research was whether using L1 loss instead of L2 loss would improve the preservation of sharp edges and details in stylized images. Based on prior work suggesting that L1 loss might better preserve sharp edges [13], we conducted experiments comparing L1 and L2 formulations for each component of our total loss function.

Contrary to our initial hypothesis, we found that L1 loss formulations actually produced less coherent stylizations in most cases. The L2 loss provided smoother gradient flows during optimization, leading to more stable training and visually pleasing results. This finding aligns with recent research suggesting that while L1 loss can theoretically preserve sharper edges, the practical benefits may be counteracted by optimization difficulties in deep neural networks, especially in the presence of multiple competing loss terms.

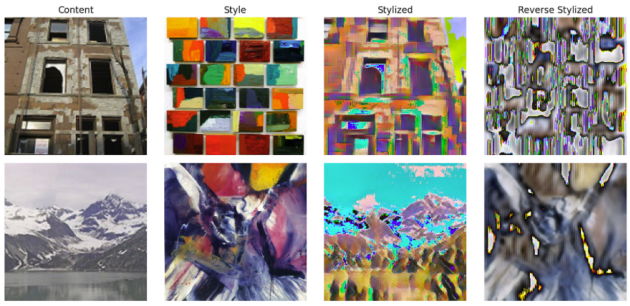

Figure 3. Two examples of high margin ($m = 4$) stylized images with $\mathcal{L}_{sep2}$. The top row shows an example of proper stylization with high margin, while the bottom row shows poor stylization from the same run. The I_r images showcase erratic behavior in both examples.

3.2.4 Training

We use the Adam optimizer [6] with the standard β_1 and β_2 configuration $(0.9, 0.999)$. Just as in the original implementation, the learning rate is set to 0.0005 with an initial warm-up and subsequent decay phase [12]. Equation (13) demonstrates how the learning rate is updated by a multiplier η at every step s. In the decay phase, a decay term γ was tuned to 0.05.

$$\eta(s) = \begin{cases} 0.1 \cdot \left(1 + 3 \times 10^{-4} \cdot s\right), & \text{if } s < 10^4 \\ \frac{1}{1+\gamma \cdot (s - 10^4)}, & \text{if } s \geq 10^4 \end{cases} \tag{13}$$

We also attempted a simple learning rate decay that updates on a plateau [11], but the convergence was slow and suboptimal.

3.3. Results

3.3.1 Quantitative Evaluation

As shown in Table 1, we found that the model with $\mathcal{L}_{sep1}$ had the best overall results, balancing both $\mathcal{L}_c$ and $\mathcal{L}_s$, while having the lowest $\mathcal{L}_c$ of the three models. The original StyTr^2 had the lowest $\mathcal{L}_s$, but at the cost of a higher $\mathcal{L}_c$. Finally, the $\mathcal{L}_{sep2}$ model performed the worst quantitatively, relative to the other two models. However, all three models had very similar losses, with the $\mathcal{L}_{sep2}$ model having almost identical loss terms to the original StyTr^2.

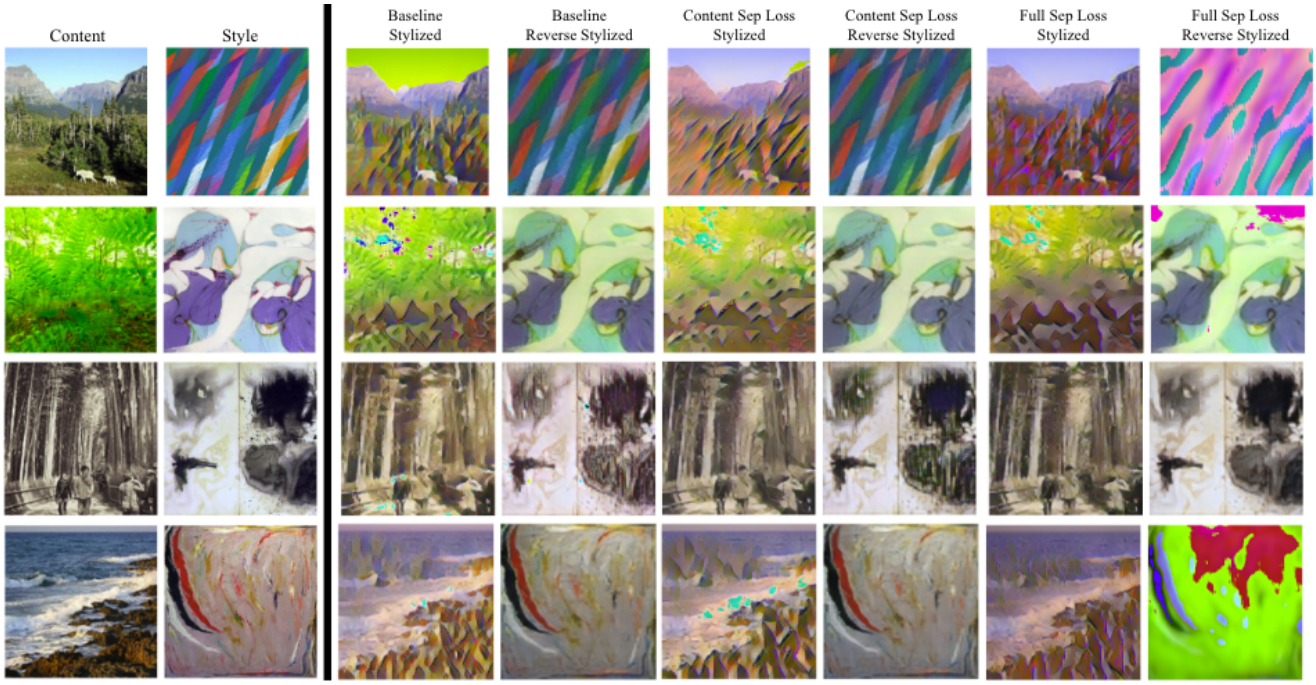

Figure 4. Four examples of generated images using the baseline, $\mathcal{L}_{sep1}$ (content sep loss), and $\mathcal{L}_{sep2}$ (full sep loss) models.

Table 1. Experimentation results. Best results are in bold.

	StyTr^2	$\mathcal{L}_{sep1}$ Model	$\mathcal{L}_{sep2}$ Model
$\mathcal{L}_c$	2.0125	**1.9909**	2.0253
$\mathcal{L}_s$	**1.9512**	1.9630	1.9555

3.3.2 Qualitative Evaluation

Unlike the loss terms, there are clear separations in the stylization characteristics for each of the three models, as shown in Figure 4. At first glance, the original StyTr^2 visualizations have much better content preservation, despite the other two models having comparable, if not better, $\mathcal{L}_c$. On closer inspection, the stylization process for the original StyTr^2 is also prone to a lot of noise, whether it be colorizing skies a different color, or adding spurious pixel artifacts throughout the image. However, the $\mathcal{L}_{sep2}$ model seems to be robust to a lot of the noise in the stylization process. With $m = 2$ and $\lambda_{sep} = 10$, the model is able to make large gradient jumps in the first few steps which marks the difference between resilience and corruption in a lot of the stylized images. However, $\mathcal{L}_{sep2}$ also warrants further investigation as it has proven to yield highly competitive results to the original StyTr^2 model with minimal impact to $\mathcal{L}_{total}$.

4. Conclusion

In this research, we proposed two forms of separability losses, $\mathcal{L}_{sep1}$ and $\mathcal{L}_{sep2}$. From a holistic perspective, our $\mathcal{L}_{sep1}$ model had the most balanced results, having both resilience to noise, but also excelling at content preservation. From a robustness perspective, our $\mathcal{L}_{sep1}$ model is highly resilient to noise in the stylization process. By directly comparing the content and style of both input images, these new loss functions allow the model to make more definitive assumptions about the stylization process. Although our proposed separability losses hinder content preservation, they significantly improve resilience to noise, especially in the context of difficult style images, like abstract art. This showcases the need to clearly define how style and content are represented in the loss functions. The current best models that we have developed are at least on-par with the original StyTr^2 model, but with further tuning, they have the potential of outperforming StyTr^2 in complex tasks.

5. Work Division

Chaitanya Tatipigari led the project and proposed the concept of separability losses, implementing both $\mathcal{L}sep1$ and $\mathcal{L}sep2$ alongside the PyTorch Lightning architecture and VGG19 feature extraction. Alec Kain collected datasets, conceptualized separability loss mechanism, implemented CAPE, encoders, and patching, and explored L1 loss as an alternative to MSE. Tyler Church developed the transformer and CNN decoders, supported hyperparameter tuning, and helped finalize the data modules and report.

Individual team member contributions have been provided and can be found in Table 2.

Table 2. Contributions of team members.

Student Name	Contributed Aspects	Details
Chaitanya Tatipigari	PyTorch Lightning Setup/Model Architecture; Separability Loss Formulation; VGG feature extraction; Team Lead	Addressed the need for more robust Style Transfer by creating and developing both separability losses. Designed and developed the PyTorch Lightning environment, model setup/architecture, and training pipelines. Developed a feature extraction mechanism using a pretrained VGG19 model. Led the team through the experimentation and design process and made sure all PyTorch modules were fully integrated into a unified architecture.
Alec Kain	Dataset Gathering; Conceptualized Separability Loss Mechanism, Transformer Encoders, CAPE, Patching	Supplied the team with content and style data. Implemented CAPE, Encoders, and Patching. Hypothesized Separability Loss mechanism and that changing original paper's MSE Losses (L_2) to L_1 Losses would result in greater model performance.
Tyler Church	Hyperparameter Testing; Transformer/CNN Decoders; Data Modules; Report Setup	Developed the Transformer Decoder and Upsampling CNN Decoder and organized the experimentation process. Assisted with hyperparameter testing and organization of final report.

References

[1] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Xingjia Pan, Lei Wang, and Changsheng Xu. Stytr^2: Image style transfer with transformers, 2022. 1, 2, 3, 4

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 2

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015. 1

[4] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017. 1

[5] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. 1

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 5

[7] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms, 2017. 1

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 4

[9] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017. 1

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 1, 2

[11] Yanzhao Wu and Ling Liu. Selecting and composing learning rate policies for deep neural networks, 2022. 5

[12] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture, 2020. 5

[13] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017. 5