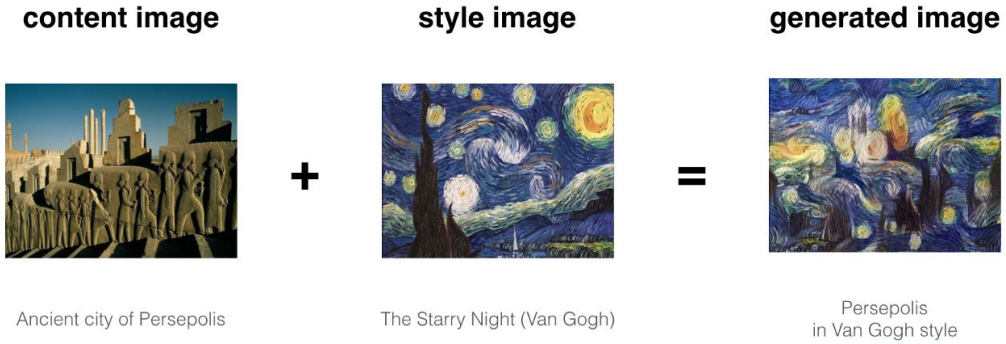
Neural Style Transfer

MEL 457 AI for Engineers

Course Coordinator:

Yash T

INTRODUCTION

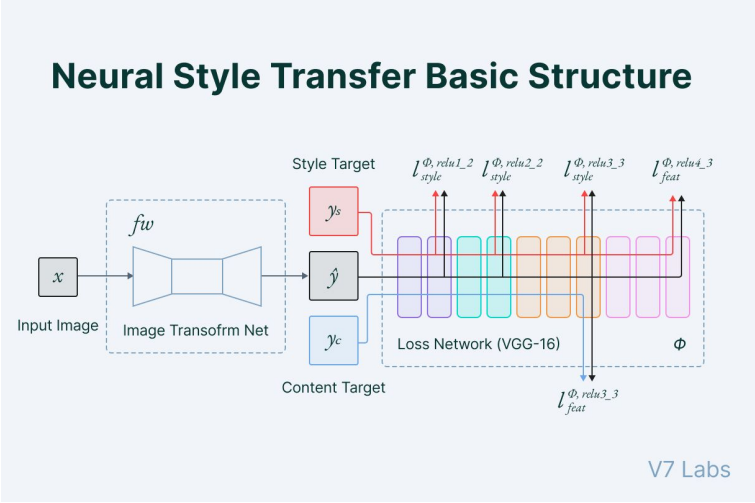


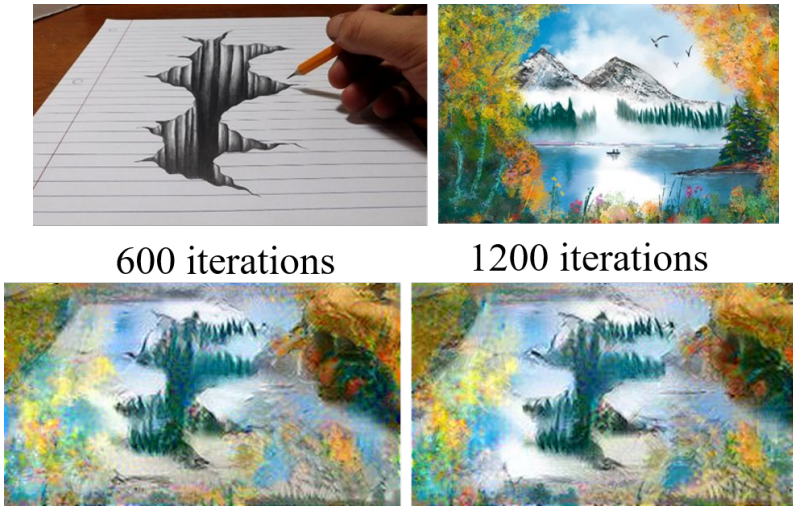
Neural Style Transfer, at the intersection of art and technology, is a revolutionary concept with broad-reaching implications. It empowers artists to blend their styles with iconic painters, offers marketing a powerful tool for brand-specific advertisements, enriches augmented reality experiences, and transforms e-commerce with virtual try-on. In gaming, it elevates immersion, while in technical fields like Computational Fluid Dynamics (CFD) and Finite Element Analysis (FEA), it enhances data visualization. This versatile technology has the potential to redefine creativity, communication, and decision-making across multiple industries.

Applications

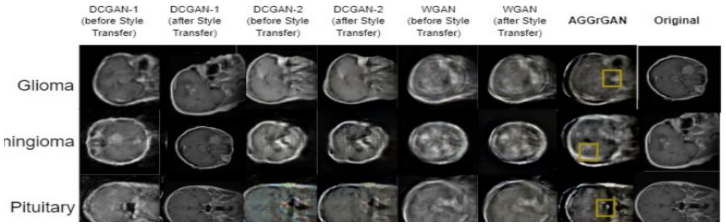
1. **Artistic Creativity:** Merging art and tech for unique, expressive creations.
2. **Movie and Video Production:** Applying styles to scenes and movies.
3. **Data Augmentation:** Enhancing ML training data diversity.
4. **Educational Engagement:** Making teaching visually captivating.
5. **Cultural Fusion & Preservation:** Modernizing and preserving cultural heritage.
6. **Virtual Reality (VR) and Augmented Reality (AR):** Immersive environments.
7. **Medical Imaging:** Enhancing medical scans and data.
8. **Interior Design:** Customizing decor and patterns.
9. **Video Game Design:** Creating unique visuals for games.

Network Architecture

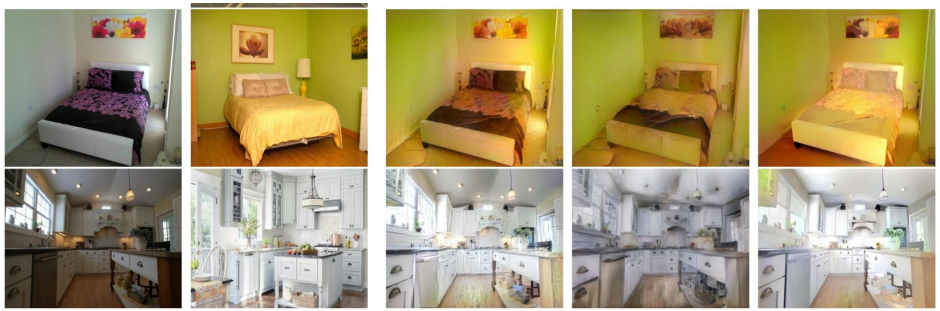




Stages and Result

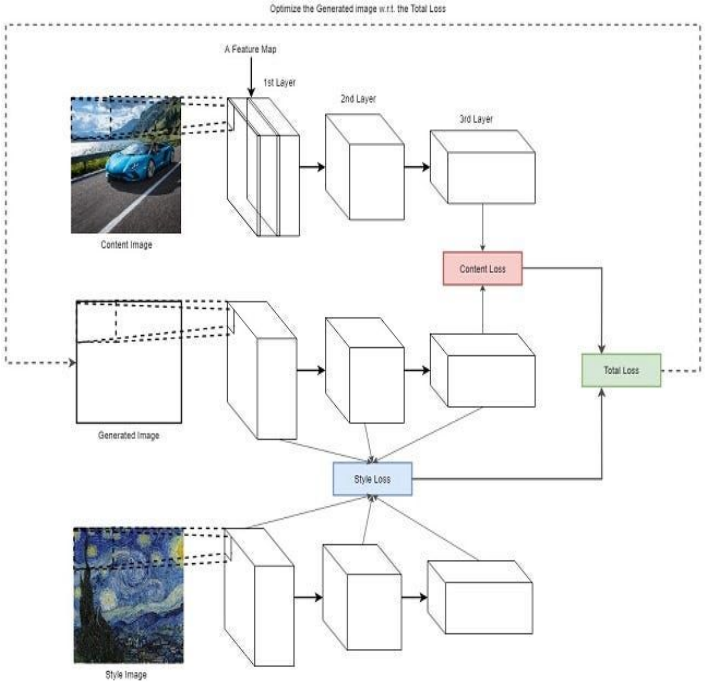


Brain tumor image generation using an aggregation of GAN models with style transfer



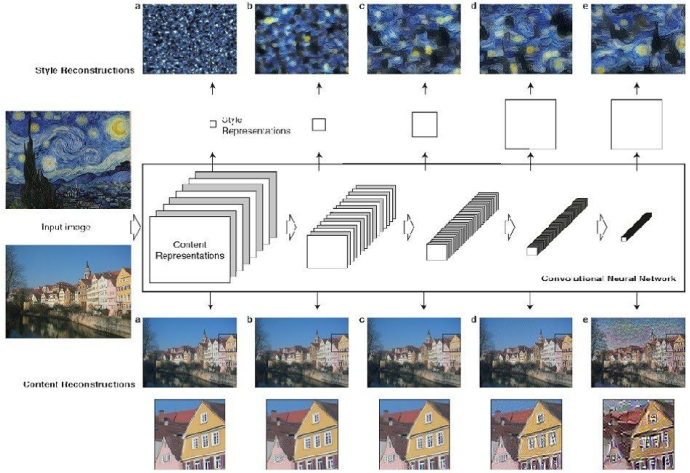
How Neural Style Transfer Works?

1. NST transforms a content image with the style of another image.
2. It uses pre-trained CNNs for feature extraction.
3. Content loss measures content preservation.
4. Style loss quantifies style resemblance.
5. Total loss combines both losses.
6. Optimization iteratively adjusts the image to minimize the total loss.

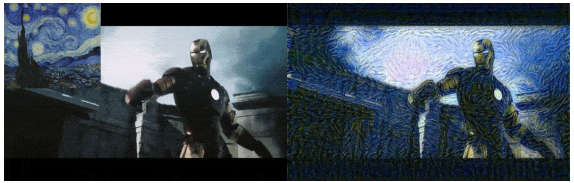


Network : VGG-19

- VGG-19 is a deep convolutional neural network commonly used in Neural Style Transfer (NST).
- It excels at feature extraction due to its 19-layer architecture, making it suitable for capturing both content and style information from images.
- Its hierarchical layers help preserve fine details and textures in generated images, contributing to the quality of stylized results in NST applications.



Loss Function are used to ensures similarity between higher-level activations of the content image and the generated image.



The content cost function ensures that the content of the content image is faithfully captured in the generated image. It uses higher-level neural network layers to calculate the mean squared error (MSE) between activations of the content and generated images, ensuring that the content remains recognizable in the generated artwork.

$$L_{content} = \frac{1}{2} \sum_{i,j} (A^l_{ij}(g) - A^l_{ij}(c))^2$$

$$L_{style} = \sum_l w^l L^l_{style} \text{ where,}$$

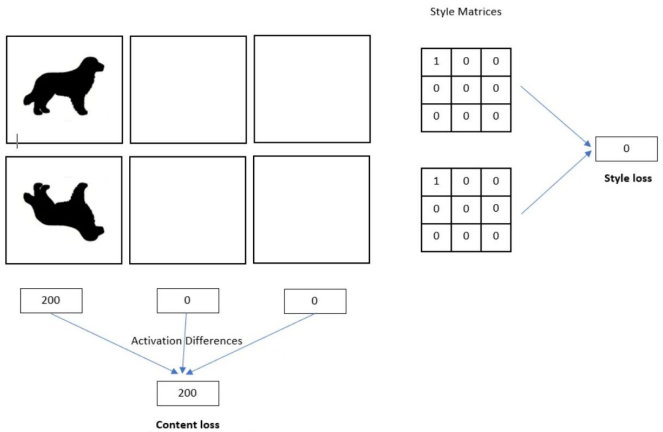
$$L^l_{style} = \frac{1}{M^l} \sum_{ij} (G^l_{ij}(s) - G^l_{ij}g)^2 \text{ where,}$$

$$G^l_{ij}(I) = \sum_k A^l_{ik}(I) A^l_{jk}(I).$$

The style loss is essential for transferring the artistic style of the style image to the generated image. It calculates the difference in feature map correlations between the style and generated images across multiple neural network layers. By minimizing this difference while considering layer weights and hyperparameters, the style loss function achieves the desired artistic style transfer, resulting in visually appealing and unique compositions.

Gram matrix and Intuition behind the Style Loss

Gram Matrix Captures Style: The Gram matrix's significance lies in encapsulating feature distribution. Minimizing style loss effectively matches feature distribution between images, facilitating artistic style transfer in Neural Style Transfer (NST).The style loss simplifies to computing style matrices for the generated and style images, measuring the root mean square difference. This involves using Gram matrices, which capture feature distribution, aligning style characteristics between images.



Computation

Content image 1

Content image 2

+

Style image 1

Style image 2

Code

FUNCTION TO CONVERT AN IMAGE TO A TENSOR

```
def imageToTensor(image):
    size = 400
    image = Image.open(image)

    im_size = size

    in_transform = transforms.Compose([
        transforms.Resize((im_size, im_size)),
        transforms.ToTensor(),
    ])

    image = in_transform(image).unsqueeze(0)

    return image
```

FUNCTION FOR GRAM MATRIX

```
def gram_matrix(tensor):
    _, n_c, n_h, n_w = tensor.size()
    tensor = tensor.view(n_c, n_h * n_w)
    g = torch.mm(tensor, tensor.t())
    return g
```

FUNCTION TO EXTRACT FEATURES

```
def get_features(image, model):
    layers = {
            '0' : 'conv1_1',
            '5' : 'conv2_1',
            '10' : 'conv3_1',
            '19' : 'conv4_1',
            '21' : 'conv4_2',
            '28' : 'conv5_1'
     }

    features = {}
    x = image
    for name, layer in enumerate(model):
        x = layer(x)
        if str(name) in layers:
            features[layers[str(name)]] = x

    return features
```

Optimize

```
optimizer = torch.optim.Adam([G], lr=0.02)
loss = torch.nn.MSELoss()
```

LOADING THE IMAGES AND FEATURES

```
# image path : /content/content.jpeg
# /content/style.jpeg
content = imageToTensor("/content/content2.jpeg").to("cuda:0")
style = imageToTensor("/content/style2.jpeg").to("cuda:0")

# display images
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
ax1.imshow(tensorToImage(content))
ax2.imshow(tensorToImage(style))

# Extract Features
content_features = get_features(content, vgg)
style_features = get_features(style, vgg)
```

Output

Original

Generated

Result

Content image 1

Content image 2

+

Style image 1

Style image 2

Generated 1

Generated 2

Conclusion

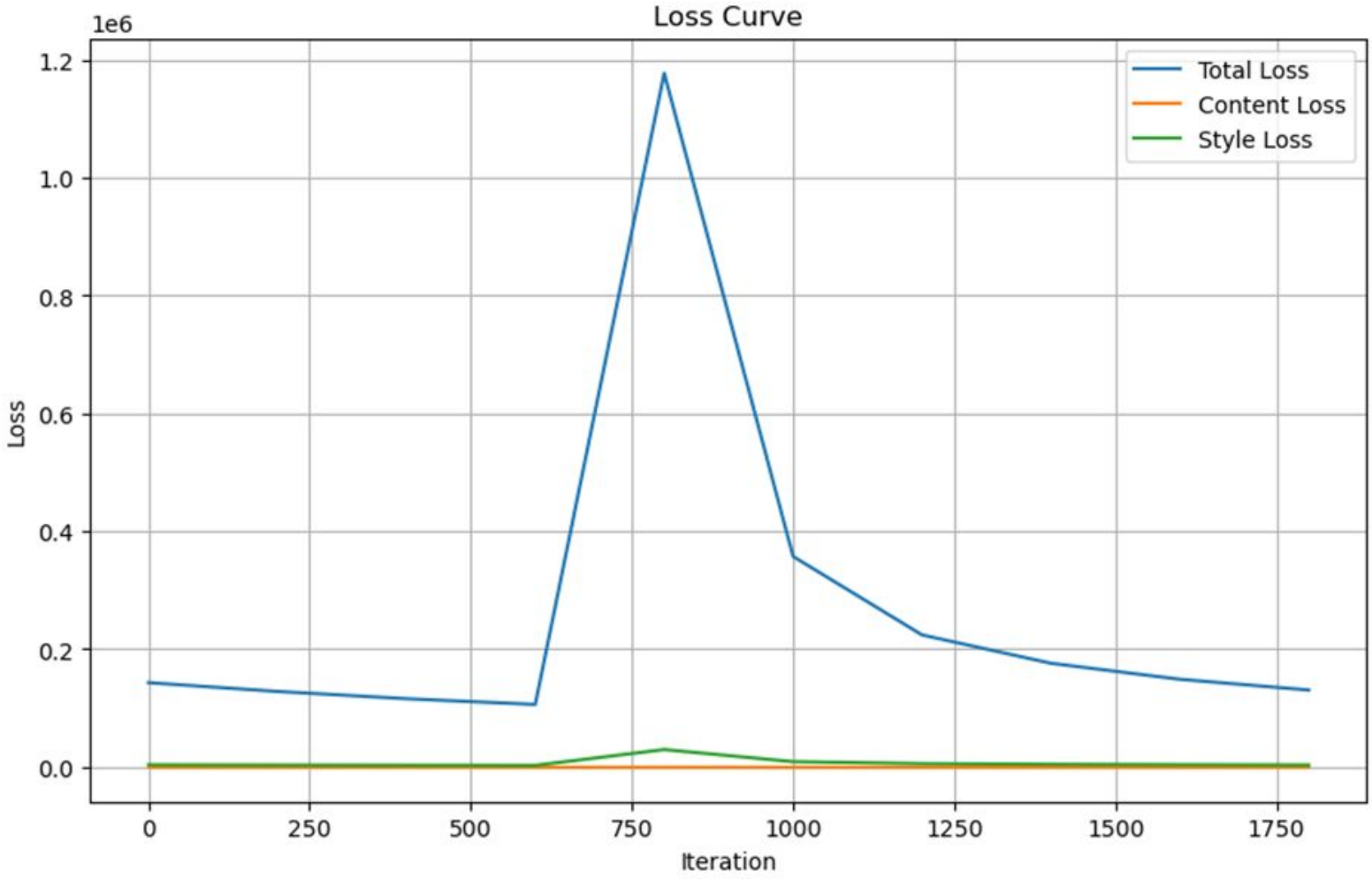
Neural style transfer is a creative tool that combines the content from one image and the style from another to produce unique artwork. This process involves understanding the necessity of NST and grasping the method's architecture. With TensorFlow, you defined the NST network, including critical functions for variables, VGG output computation, loss calculations, and optimization. You delved into the content and style losses, comprehending how they contribute to the final loss, balancing content preservation and style transfer. By running the model, you witnessed the generation of captivating artwork through this innovative fusion of content and style.

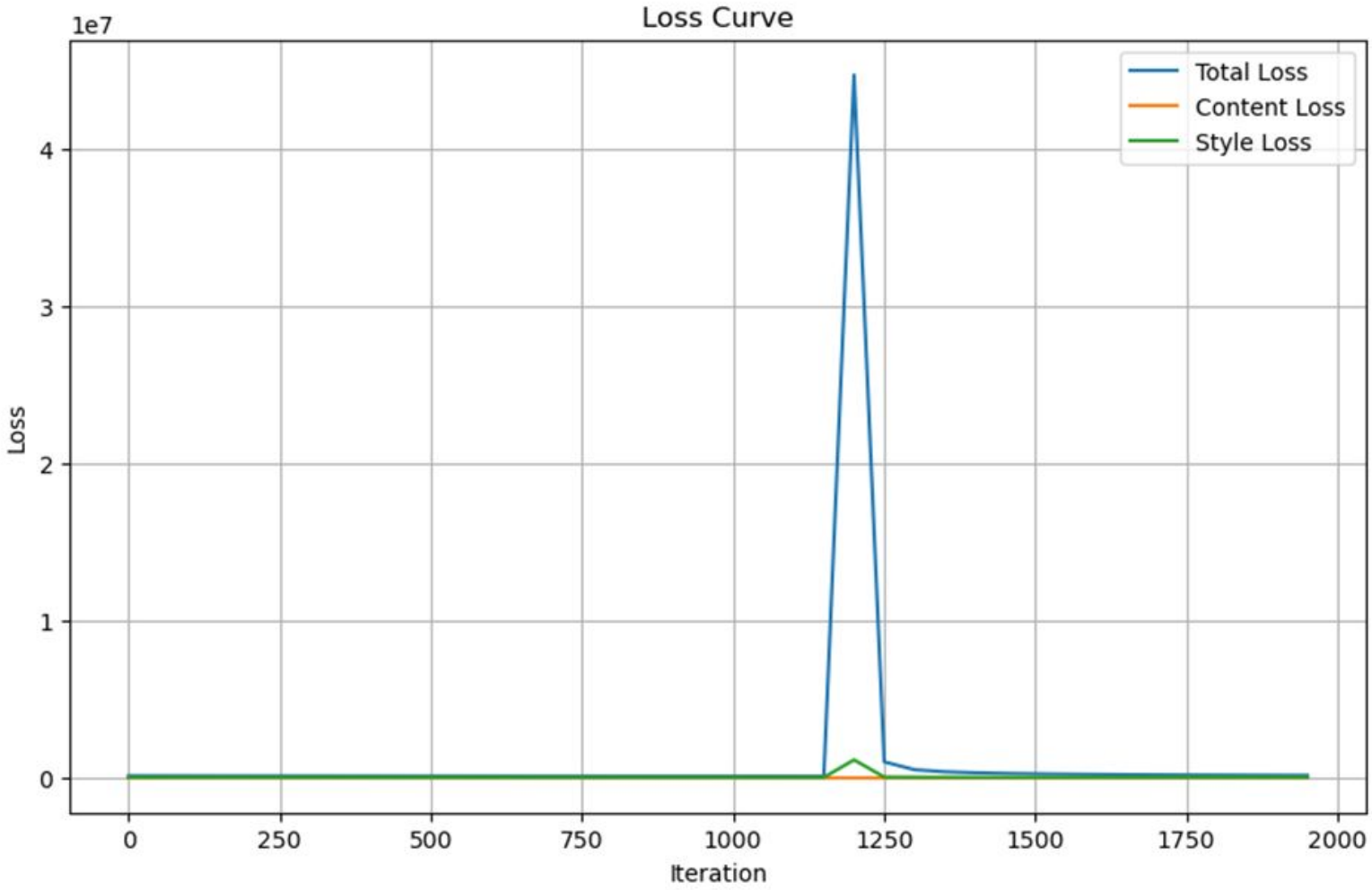
Dataset

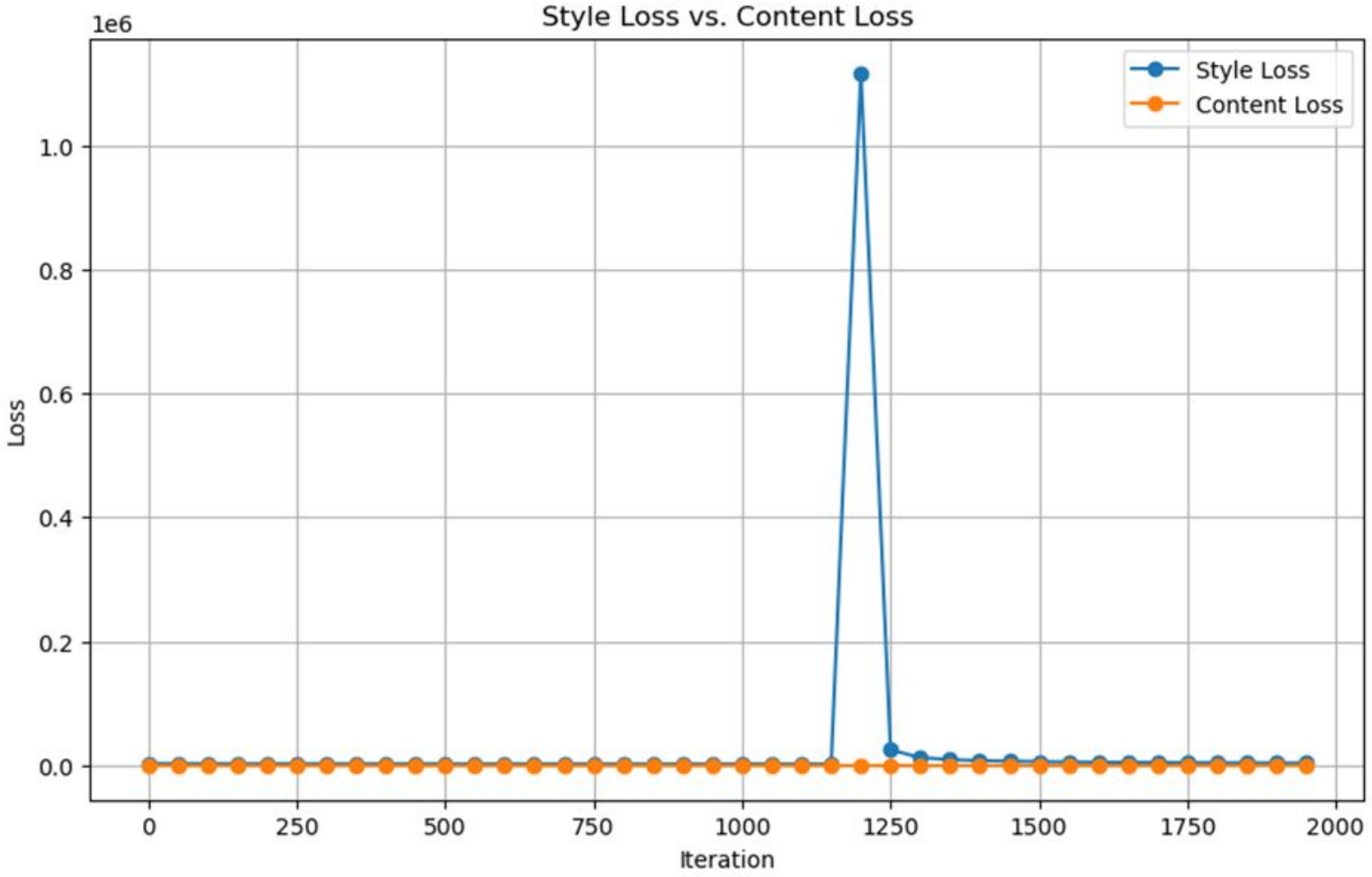
- **Neural Style Transfer (NST) typically doesn't require a specific dataset for training in the traditional sense, as it leverages pre-trained Convolutional Neural Networks (CNNs) for feature extraction.**
- **However, during the optimization process of style transfer, you'll need a content image and a style image to create the stylized output. These images serve as the input to the NST algorithm, and the content and style features are extracted from them.**
- **Thus datasets like WikiArt, COCO, Cityscapes etc. can be used.**
    - WikiArt Dataset
    - COCO Dataset

Literature References

1. Neural Style Transfer: A Review | IEEE Journals & Magazine
2. Structure-Preserving Neural Style Transfer | IEEE Journals & Magazine
3. [1801.01589] Neural Style Transfer for Audio Spectograms
4. Stereoscopic Neural Style Transfer
5. Fast Neural Style Transfer for Motion Data | IEEE Journals & Magazine
6. Neural style transfer: a paradigm shift for image-based artistic rendering?
7. Semantic Segmentation of Agricultural Images Based on Style Transfer Using Conditional and Unconditional Generative Adversarial Networks
8. https://dl.acm.org/doi/pdf/10.1145/3550454.3555517


Loss Curve
1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Loss
0
250
500
750
1000
1250
1500
1750
Iteration
Total Loss
Content Loss
Style Loss


Loss Curve
1e7
Total Loss
Content Loss
Style Loss
Loss
Iteration
0
1
2
3
4
0
250
500
750
1000
1250
1500
1750
2000


Style Loss vs. Content Loss
1e6
Style Loss
Content Loss
1.0
0.8
0.6
0.4
0.2
0.0
Loss
0
250
500
750
1000
1250
1500
1750
2000
Iteration