Farzi Van Gogh

Task description

- **Making a Renderer**- Generate a painted version of an input image, given a specific painting style (oil painting, water-color).
- **Neural Style Transfer**- render an input image in the style of a given style image.

Approach

- We built on the paper 'Stylized Neural Painting' by Zhengxia Zou and Co.
- The paper's approach has two parts training a renderer and searching for stroke parameters.

Renderer

- The renderer has two parts, the shading network and the rasterization network
- The shading network is used to generate the stroke colormap.
- The rasterization network is used for generating the stoke silhouette.

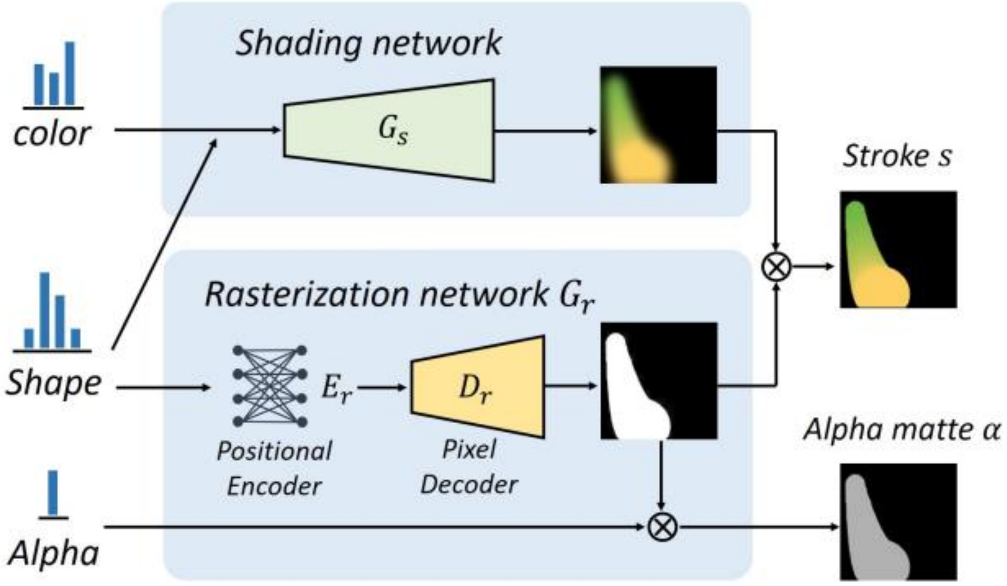
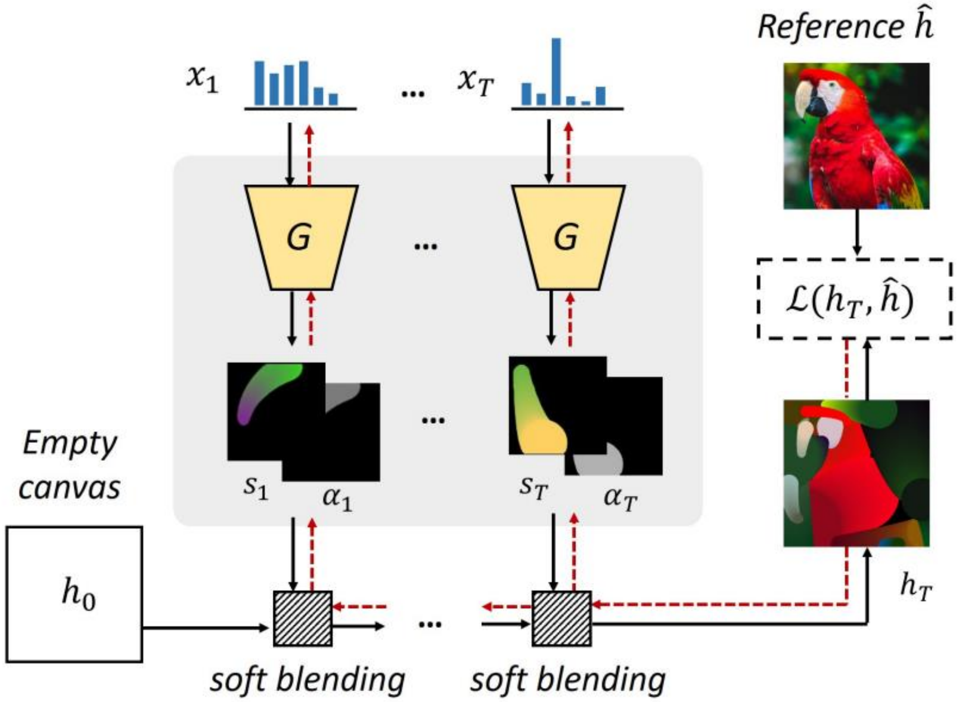


Image generation

- For making the final rendered image, we start with a set of stoke parameters.
- We generate an image by feeding these parameters into the trained renderer.
- We now optimize these parameters using gradient descent.



Loss function

$$\mathcal{L} = \beta_{\ell_1}\mathcal{L}_{\ell_1} + \beta_{ot}\mathcal{L}_{ot} + \beta_{sty}\mathcal{L}_{sty},$$

- For rendering the loss used is the sum of pixel l1 loss and a translation loss which the paper names as sinkhorn loss.
- For style transfer we add a term of style loss to the above losses.
- The style loss is calculated using the Grammian of the rendered image and the style image after passing through some layers of a pretrained VGG16 network.

Major Contributions

- Modifying loss function using Laplacian
- Using a lightweight renderer architecture
- Extending the implementation for videos

Laplacian

- Inspired by Laplacian-Steered Neural Style Transfer by Shaohua Li and Co.
- Incorporated Laplacian loss within neural style transfer to accentuate and sharpen edge features for enhanced visual definition.
- Utilized the convolution process with a specified matrix 'D' to approximate the Laplacian of an image.

$$D = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Laplacian Loss

- Given two images x and y, we adapted the loss function by incorporating the contribution of L_{lap}- a metric used to assess the dissimilarity between the Laplacians of images x and y

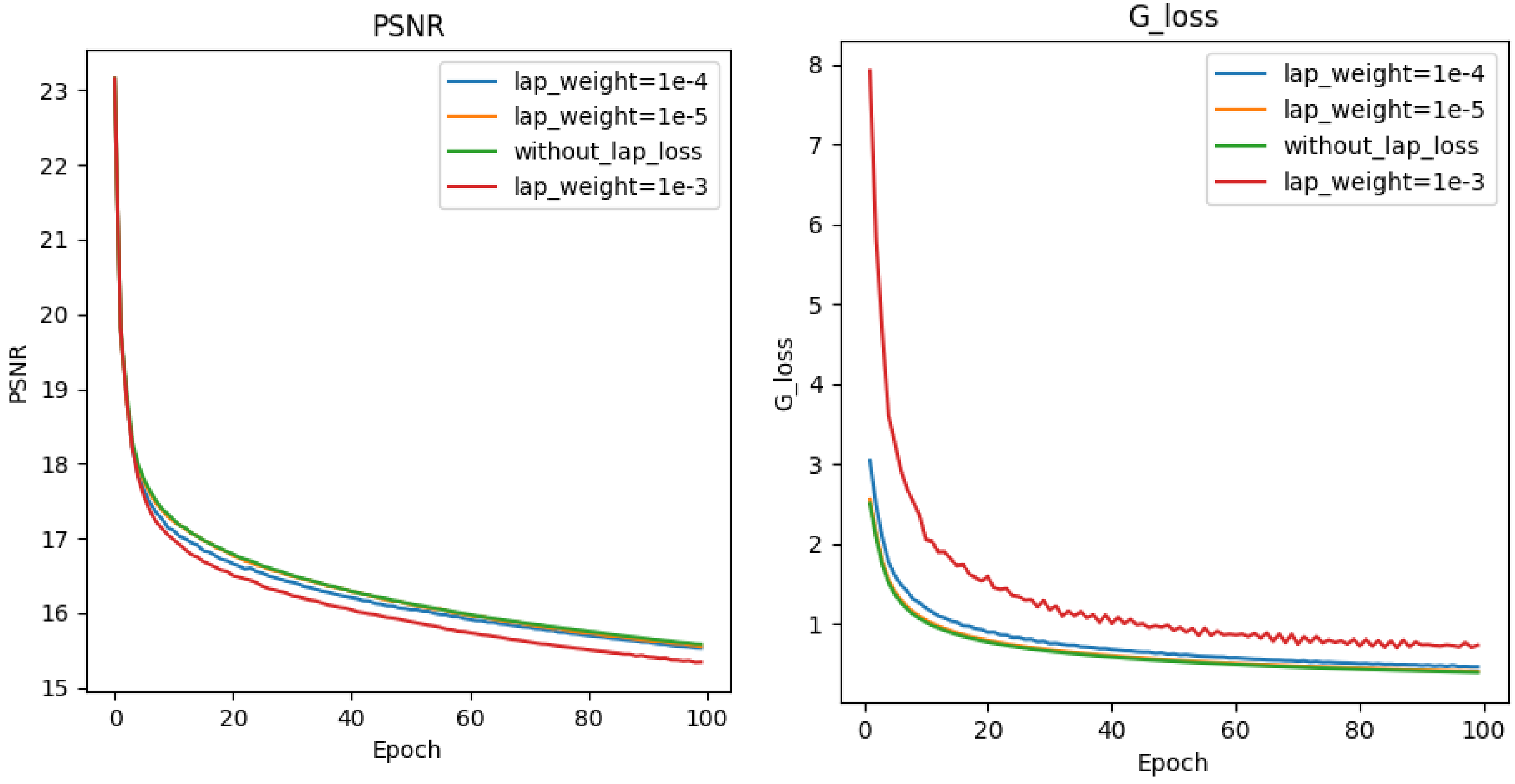
$$\mathcal{L}_{\text{lap}} = \sum_{ij} (D(\boldsymbol{x}_c) - D(\hat{\boldsymbol{x}}))^2_{ij}.$$

- This adjustment involved strategically introducing L_{lap} after specific layers in the VGG network, and taking their sum to get a modified loss function

Without Laplacian

With Laplacian

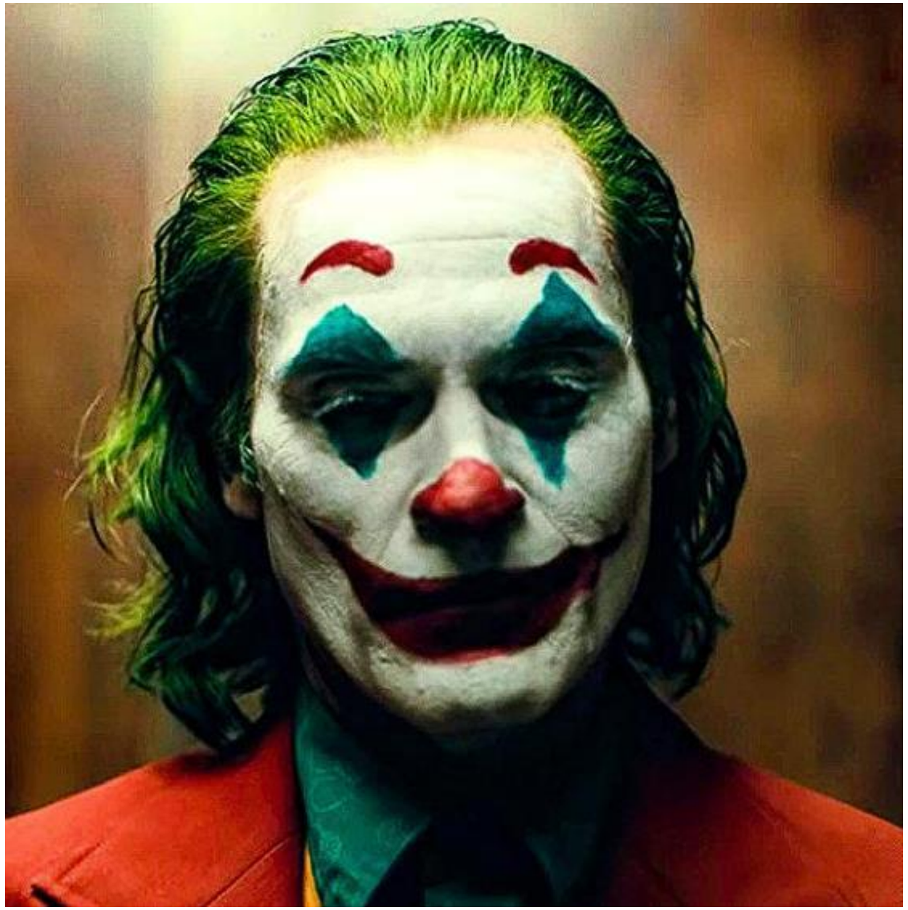
Farzi Van Gogh



Comparing effect of Laplacian Loss

Lightweight Renderer

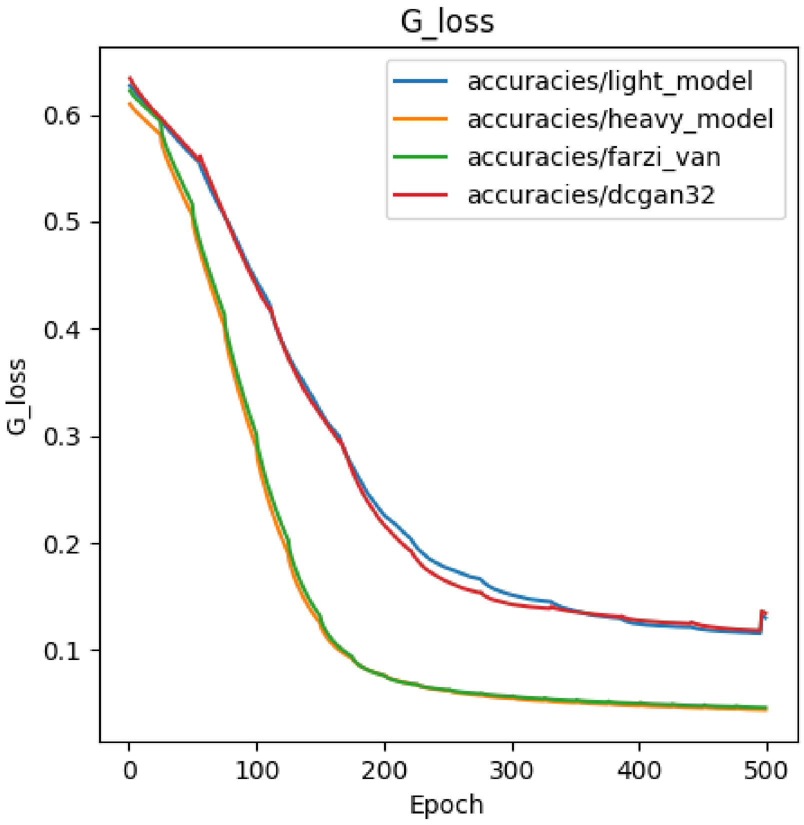
- We designed both the shader and a rasterization network for a new renderer.
- We significantly brought down the number of parameters from 18.1 million to 5.4 million, roughly 3 times.
- We observe an approx. Speedup of 5% in the rendering of images

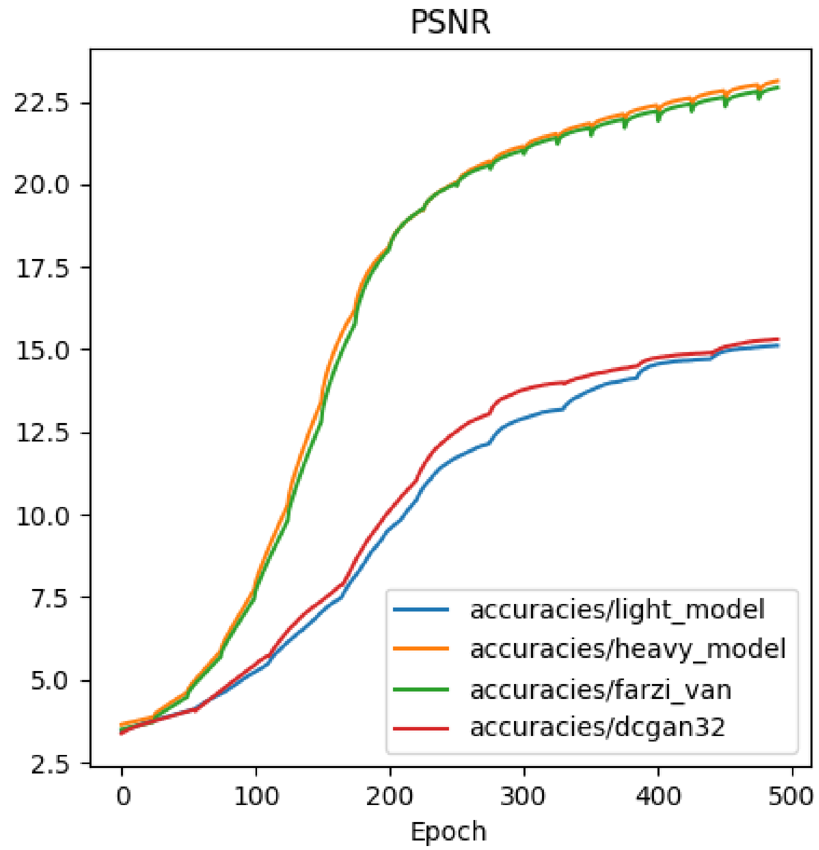


Our Render Network

Water-color Rendered Image

Style Transferred Image

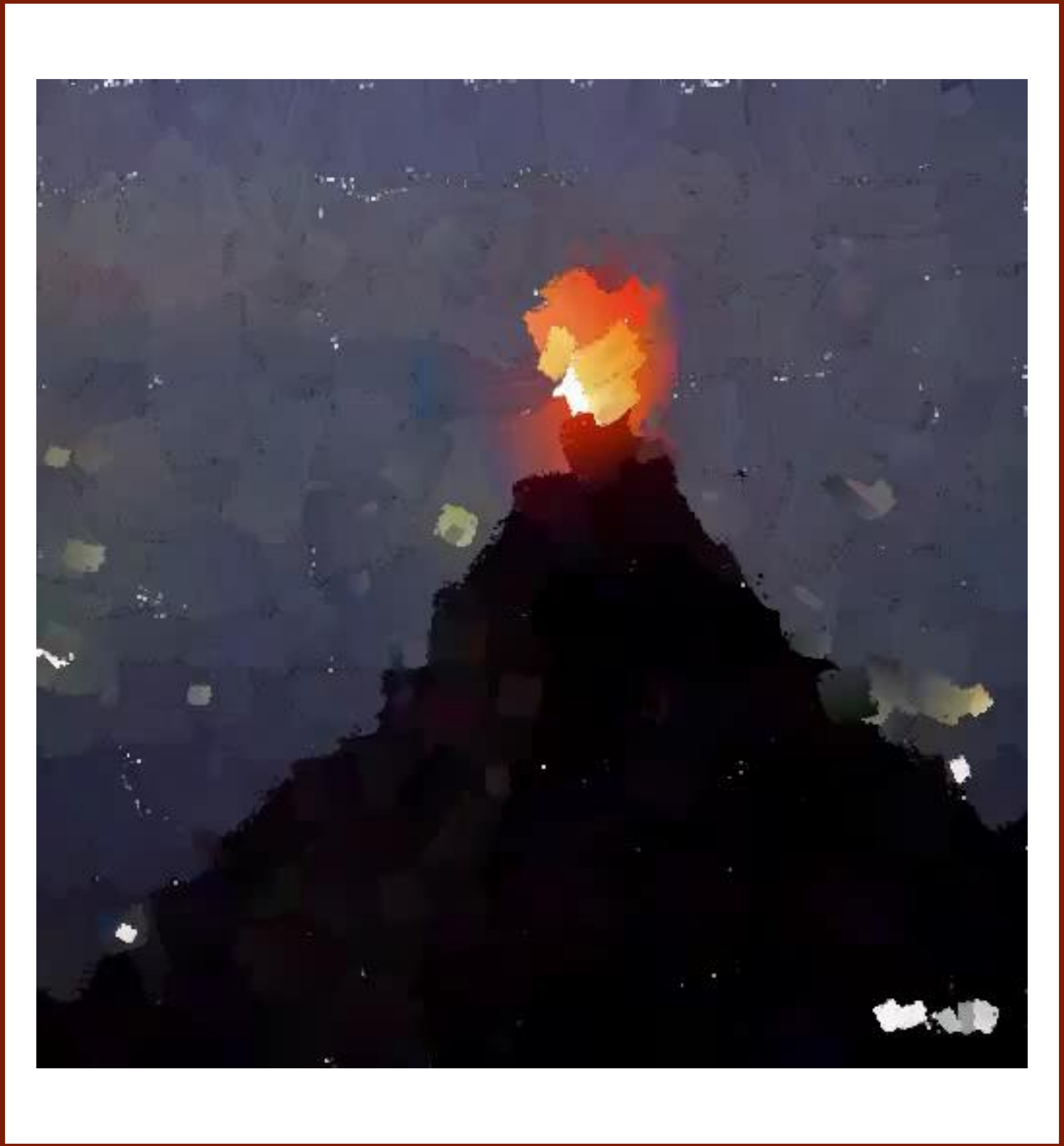




Comparison of Models

Video Style Transfer

- We extended this style transfer framework to videos.
- We added another interface that takes a video and renders it into a particular input style



Other contributions

We sped up the existing code by adding vector operations in place of loops wherever possible, resulting in reduction of running time, a 7% speedup

We tried using other networks like VGG19 for calculating the style loss, we observed similar results across different architectures.

Changes made in code

- Source Code [Stylized Neural Painting]
- In loss.py added code for laplacian loss in VGGStyleLoss class.
- Created the classes Shader, Rasterizer and Light-Net for building our renderer model.
- Added VideoPainter class for implementing rendering transfer over Videos.
- Created demo_video.py as an interface for video rendering.

Work Division

- Ideation/Searching for related papers – Akshat, Soham
- Analysing Base paper model and reproducing – Akshat, Ayush
- Introduction of Laplacian – Ayush, Soham
- Application to videos – Sankalan, Soham
- New model with fewer parameters – Ayush, Sankalan
- Trying new architecture like VGG19 in base model – Akshat, Sankalan
- Code Optimization like vectorization – Andreas, Sankalan
- Writing of scripts for data generation – Akshat, Andreas
- Preparation of presentation/report - Andreas, Soham
- Writing scripts for plotting - Andreas, Ayush

- All team members have contributions in all the above topics but people specializing in the corresponding points have been mentioned