Neural Style Transfer: A Tech Report

Jitesh Jain[†] 19114039
Gagan Sharma[†] 19114032
Divyansh Agarwal[†] 19115055
R Chinmay[†] 19114067
Md Junaid Mahmood[†] 19116040

Group **1**
Indian Institute of Technology Roorkee[*]

Figure 1: **Neural Style Transfer Results** We are able to reproduce the baseline and achieve competitive performance to the pseudo GTs. We trained for 1000 iterations with ResNet-18 [2] feature extractor and LBFGS optimizer for the above results. **Zoom-in for best view**.

Abstract

Neural Style Transfer is one of the earliest applications of deep learning in creative art. However, the existing technique has not changed since the first method was proposed in 2015. Since then, there have been many recent developments in deep learning and creative art, including the formulation of new feature extractors, optimizers, and losses. This tech report presents our experimental settings and findings when ablating on the feature extractor, optimizer, the number of iterations, losses, and initialization techniques. We propose to use pseudo ground truths to evaluate the performance of our methods. We find that ResNet-18, LBFGS, content initialization, and a combination of style, content, and tv loss show the best performance. Our code is publicly available at https://github.com/praeclarumjj3/NST-Tech.

[*]All authors contributed equally to the project. All authors worked under the supervision of Professor Pravendra Singh at the Department of Computer Science and Engineering, IIT Roorkee.

Final Report for Course Project under CSN-526: Machine Learning.

1 Introduction

Neural Style Transfer (NST) was first proposed by Gatys et al. in [1]. Since then, the stylizing images using filters for creative art and image editing have gained popularity among the general public. However, to the best of our knowledge, there is no existing technical study that ablates the several factors like feature-extractors, optimizers, number of iterations, initialization of the input image, and the loss terms. We work on developing an extensive study conducting ablation studies with different settings. Since there exists no real ground truth for the stylized image, we propose to use the results from the published work by Huang *et al.* [3] as pseudo ground truths for quantitatively evaluating our results.

We first briefly explain our method in Sec. 3. We then present our experimental settings and results in Sec. 4. Lastly, we present a discussion on this technical study. To the best of our knowledge, we are the first to present a technical study on Neural Style Transfer with ablations on different factors and pseudo ground truths for evaluation.

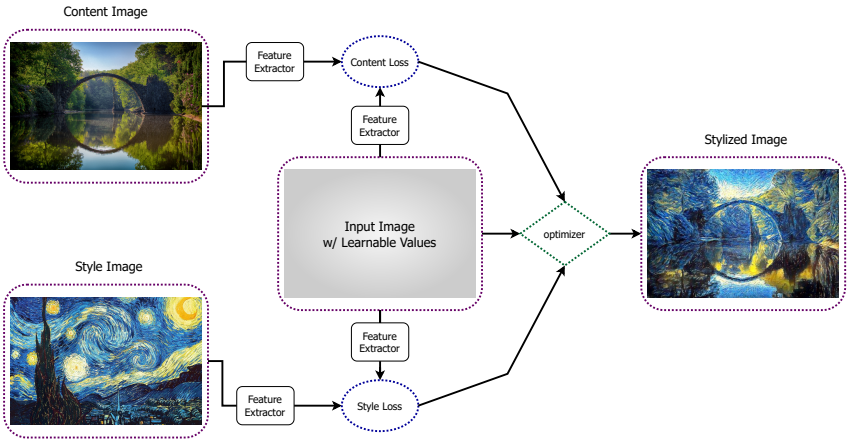

Figure 2: **Neural Style Transfer Pipeline** NST learns the *Input Image* with supervision from the *Content Image* and *Style Image*.

2 Related Works

A Neural Algorithm of Artistic Style. Gatys et al. first proposed the *Neural Style Transfer* algorithm in [1] in 2015. They formulated style transfer as an optimization problem. They proposed to first extract content and style features from corresponding images and then optimize a learnable image using the combination of features. The result is a stylized image with guidance from the content and style features as shown in Fig. 1. They didn't use any dataset but only a content and style image during training which makes this algorithm widely accepted.

Arbitrary Style Transfer in Real-Time. Huang et al. [3] proposed the use of Adaptive Instance Normalization (AdaIN) to combine content from the content image and style from the style image by transferring feature statistics using an encoder-decoder structure. This work gives impressive style transfer performance.

In our report, we use the algorithm proposed in [1] for our experiments and ablation studies as described in Sec. 3. Owing to the impressive performance of [3], we use the stylized images produced using their pretrained model as pseudo-ground truths as described in Sec. 4.

3 Method

Our method is mainly inspired by [1]. We describe the baseline method briefly and then mention the changes that we plan to try.

Neural Style Transfer is an optimization problem, which takes in 3 images as input: *content image*, *style image* and a *learnable input image* as shown in Fig. 2. The goal is to learn the input image such that the final result consists of the content of the Content image and style of the Style image. We extract the content and style (*gram matrices*) features from any given image by using a deep neural network (like VGG16 or VGG19 or ResNet-18). The shallow layers in feature extractor store pixel colour information, and style information and the deeper layers store content (structure) related information. We learn the input image with supervision from a weighted sum of content loss: between the features from content and input image, and style loss: between the features from style and input image as shown in Eq. (1).

$$\mathcal{L}_{total} = \lambda_{content}\mathcal{L}_{content} + \lambda_{style}\mathcal{L}_{style} \tag{1}$$

$\lambda_{content}$ and λ_{style} are hyperparameters. We set $\lambda_{content} = 1 \times e^0$ and $\lambda_{style} = 1 \times e^6$ for all our experiments.

4 Experiments

4.1 Data and Metrics

Data. For training and evaluating our methods, we choose 5 pairs of content and style images from the web as shown in Fig. 3. We average the metric scores over the 5 pairs for recording the final scores.

Metrics. Since there exists no real ground truth for the stylized image, we use the results from the Adaptive Instance Normalization based method proposed by Huang *et al.* [3] as pseudo ground truths (Fig. 3) for quantitatively evaluating our results. We use the Root Mean Square Error (RMSE), Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) metrics for comparing our results with the pseudo ground-truths.

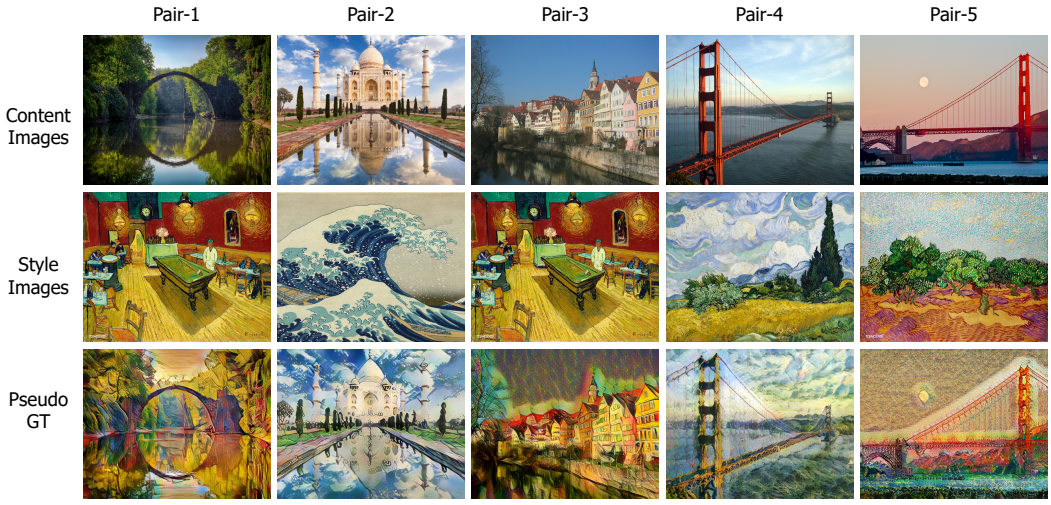

Figure 3: **Training Data Pairs.** We use 5 pairs of content and style images taken from the web for all our experiments. **Zoom-in for best view**.

4.2 Implementation Details

We implement our method in PyTorch [6]. We use the publicly available models on the torchvision model zoo for our feature extractors. When using VGG-16 [7] as our feature extractor, we use the features from the `relu2_2` layer for calculating the content loss and the gram matrices from [`relu1_2`, `relu2_2`, `relu3_3`, `relu4_3`] layers for calculating the style loss. In case of VGG-19 [7], we use `relu4_2` layer for content loss and [`relu1_1`, `relu2_1`, `relu3_1`, `relu4_1`, `relu4_2`, `relu5_1`]

for style loss. When using ResNet-18 [2], we use features from `layer1` for content loss and the gram matrices from [`conv1`, `layer1`, `layer2`, `layer3`] for style loss.

Unless stated otherwise, we train for 1000 iterations with the LBFGS optimizer with the Resnet-18 feature extractor.

4.3 Ablation on Feature Extractors

We experiment with three different feature extractors: VGG-16 [7], VGG-19 [7] and ResNet-18 [2]. We compare the three models on RMSE, SSIM, and PSNR values. We observe that ResNet-18 gives the best performance as shown in Tab. 1. Thus, we use ResNet-18 as our feature extractor for further experiments.

Metric	VGG-16	VGG-19	ResNet-18
RMSE ↓	0.01213	0.01231	**0.01187**
PSNR ↑	38.336	38.211	**38.528**
SSIM ↑	0.8798	0.8767	**0.8864**

Table 1: **Ablation on Feature Extractors.** ResNet-18 gives the best scores on RMSE, SSIM and PSNR.

4.4 Optimizers

We experiment with different optimizers among LBFGS, Adam [4] and AdamW [5] using the ResNet-18 extractor. In order to find the best learning rate (lr) in case of Adam and AdamW, we tune the lr hyper-parameter on Pair-2 (Fig. 3). We find lr $= 1e^{-1}$ and lr $= 1e^{-2}$ to be the best setting for Adam and AdamW, respectively. Evaluation for tuning the learning rate was also done using the PSNR, SSIM and RMSE metrics. We find LBFGS to show the best performance and hence, use LBFGS for further experiments as shown in Tab. 2.

Metric	LBFGS	Adam	AdamW
RMSE ↓	**0.01187**	0.01270	0.01238
PSNR ↑	**38.528**	38.001	38.235
SSIM ↑	**0.8864**	0.8785	0.8801

Table 2: **Ablation on Optimizers.** LBFGS gives the best scores on RMSE, SSIM and PSNR.

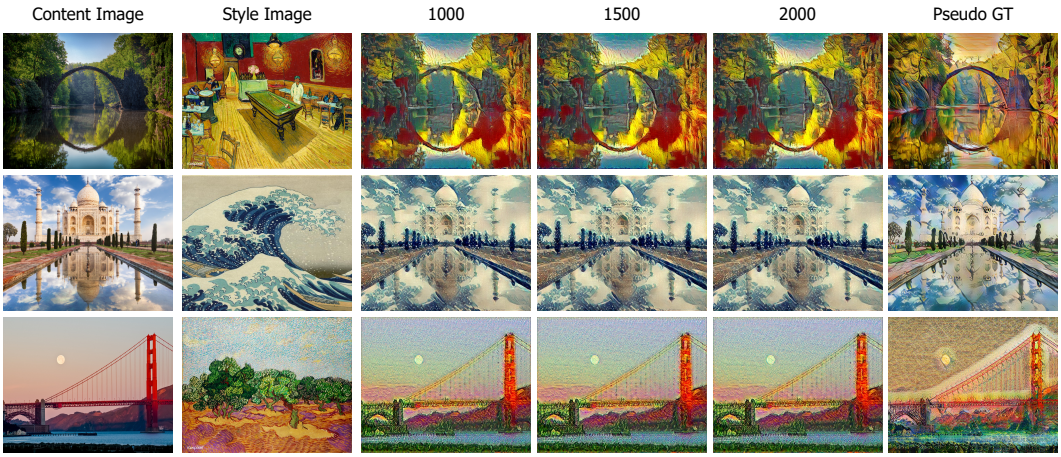

Figure 4: **Qualitative analysis on Number of Iterations** We find small visual differences when training for longer duration. Quantitatively the 1000 iterations setting gives the best performance. **Zoom-in for best view**.

4.5 Number of Iterations

We also experiment with the number of iterations for the ResNet-18 feature extractor and LBFGS optimizer. We selected three numbers 1000, 1500, 2000 and used the same methodology in order to compare and evaluate the three models using different numbers of iterations. We find that the 1000 iterations gives the best performance as shown in Tab. 3 and Fig. 4. Thus, we set the number of iterations to 1000 for further experiments.

Metric	**1000**	**1500**	**2000**
RMSE ↓	**0.01187**	0.01191	0.01191
PSNR ↑	**38.528**	38.504	38.501
SSIM ↑	**0.8864**	0.8859	0.8860

Table 3: **Ablation on Number of Iterations.** 1000 iterations give the best scores on RMSE, SSIM and PSNR.

4.6 Initialization Settings

We experiment with three initialization settings for the learnable input image: content, style or a random noise image. We used the ResNet-18 feature extractor, LBFGS optimizer and 1000 iterations for this experiment. We tune the values $\lambda_{content}$ and $\lambda_{content}$ on Pair-2 for the different settings. We find ($\lambda_{content} = 1 \times e^{0}$, $\lambda_{style} = 1 \times e^{6}$), ($\lambda_{content} = 1 \times e^{3}$, $\lambda_{style} = 1 \times e^{0}$) and ($\lambda_{content} = 1 \times e^{5}$, $\lambda_{style} = 1 \times e^{0}$) give best results for initialization with content, style and random image respectively. The mean results for all 5 pairs show that initialization of the input image with content image gives the best performance as shown in Tab. 4.

Metric	**Content**	**Style**	**Random**
RMSE ↓	**0.01187**	0.01290	0.01298
PSNR ↑	**38.528**	37.865	37.813
SSIM ↑	**0.8864**	0.8713	0.8719

Table 4: **Ablation on Different method of initialization.** Initialization with Content image gives the best scores on RMSE, SSIM and PSNR.

We also find that the initialization setting is pair dependent, as shown in Tab. 5. For Pairs-1,3,4, content initialization gives the best performance on all the metrics. However, style initialization works best for Pair-2, and for Pair-5, it is unclear which works best. However, overall, content initialization gives the best performance, as shown in Tab. 4.

Pair↓	**Content**			**Style**			**Random**		
Metric→	RMSE	PSNR	SSIM	RMSE	PSNR	SSIM	RMSE	PSNR	SSIM
Pair-1	**0.01187**	**38.5038**	**0.8694**	0.01507	36.4378	0.8102	0.01494	36.5101	0.8229
Pair-2	0.01227	38.2207	0.8930	**0.01189**	**38.4891**	**0.9059**	0.01206	38.3723	0.9006
Pair-3	**0.01195**	**38.4488**	**0.8692**	0.01458	36.7221	0.8463	0.01487	36.5487	0.8357
Pair-4	**0.01038**	**39.6741**	**0.9181**	0.01041	39.6495	0.9214	0.01043	39.6313	0.9187
Pair-5	0.01288	37.7964	**0.8822**	**0.01255**	38.0263	0.8726	0.01258	**38.0043**	0.8813

Table 5: **Pair-Wise Ablation on Different method of initialization.** The initialization setting is pair-dependent.

4.7 TV Loss

We also experiment by adding Total Variation [8] (TV) Loss to the loss term, which originally consisted of content loss and style loss. We set the TV weight, $\lambda_{tv} = 1 \times e^{0}$ (tuned on Pair-2), and used ResNet-18 feature extractor, LBFGS optimizer, 1000 number of iterations and initialized the learnable input image with the content image. We find that including the TV Loss term in the total loss equation gives better performance, as shown in Table Tab. 6. Surprisingly the results with the

Metric	without TV-Loss	with TV-Loss
RMSE ↓	0.01187	**0.01140**
PSNR ↑	38.528	**38.879**
SSIM ↑	0.8864	**0.8946**

Table 6: **Ablation on Different Loss Combinations.** Including TV-loss with $\lambda_{tv} = 1 \times e^0$ gives the best scores on RMSE, PSNR and SSIM.

TV-loss term are worse, as shown in Fig. 5. However, the metric scores are better for TV loss settings, which raises questions about the evaluation metrics. We might need better metrics for evaluating the performance of style transfer.

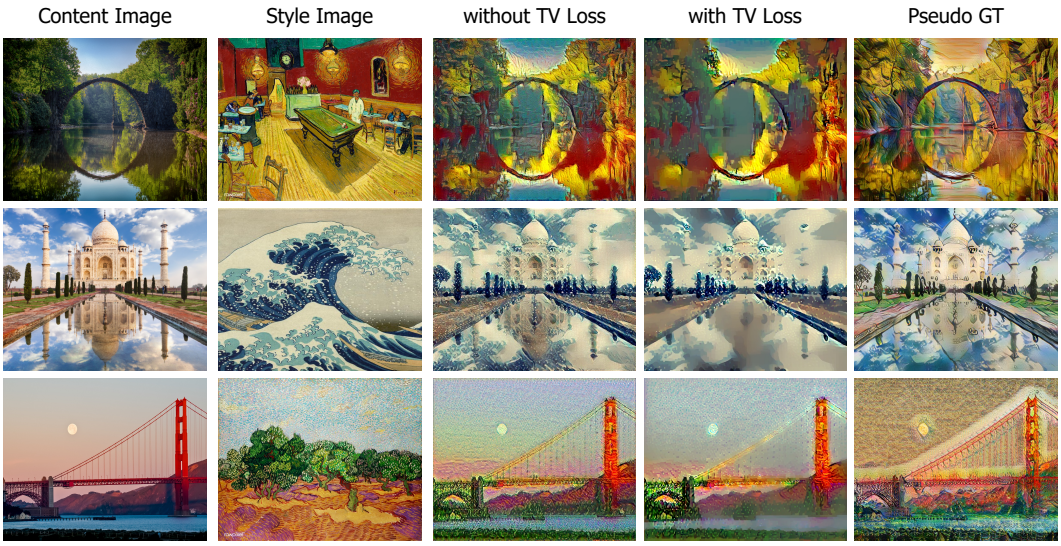

Figure 5: **Qualitative analysis on using tv-loss.** We find the visual quality worse when using tv loss with unclear boundaries. **Zoom-in for best view**.

5 Conclusion and Discussion

We presented an extensive experimental analysis considering various Neural Style Transfer task factors. To the best of our knowledge, we are the first to present an extensive technical study on Neural Style Transfer. We also proposed to use pseudo ground truth to evaluate and compare different experimental settings. We adopt the widely used ResNet-18 to the NST task and perform better than the original baseline. Additionally, we conduct ablations on various critical performance factors like optimizers and loss terms. We hope our study will help the community gain a deeper practical understanding of neural style transfer.

6 Acknowledgment

We thank Professor Pravendra Singh at the Department of Computer Science and Engineering, IIT Roorkee for providing us with an opportunity to work on the project. This technical report is a part of our submission for the course project under CSN-526: Machine Learning.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv*, 2015. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016*, 2016. 1, 4

[3] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 3

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 4

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 3

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3, 4

[8] Wikipedia. Total variation. 5