Neural Style Transfer

CSN-382 Machine Learning

26 APRIL 2023

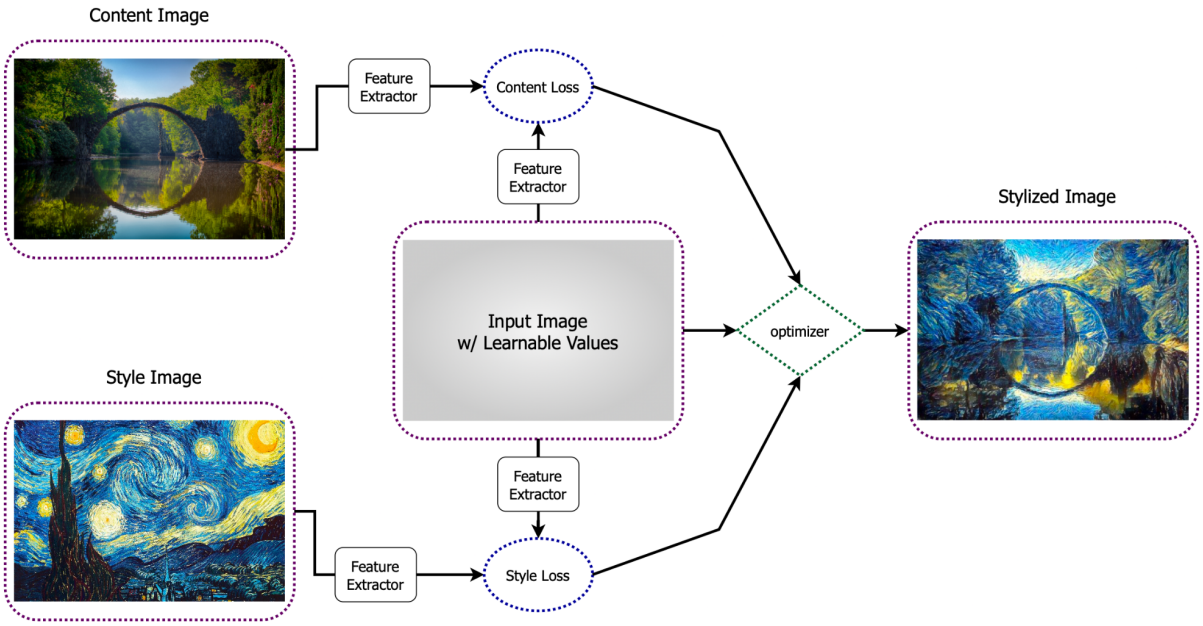
BY
Aditya Singh Bisht
20117013

Abstract

One of the early uses of deep learning in artistic creation is Neural Style Transfer. But since the initial approach was put forth in 2015, the current procedure has remained unchanged. Since then, a lot has changed in the fields of deep learning and artistic creation, including the introduction of new feature extractors, optimizers, and losses. This technical paper details the experimental parameters I used and the results I obtained while varying the feature extractor, optimizer, iteration count, losses, and initialization methods. In order to assess the effectiveness of our methods, I suggest using fictitious ground truths. I discover that the optimum performance is provided by ResNet-18, LBFGS, content initialization, and a combination of style, content, and TV loss.

Introduction

Gatys et al. first proposed neural style transfer (NST) in [1]. Since then, the public has grown more and more accustomed to employing filters to stylize photographs for creative art and image editing. To the best of my knowledge, however, there isn't any technical research in place that eliminates a number of variables including feature-extractors, optimizers, iterations, initialization of the input image, and loss terms. I am preparing a comprehensive study and conducting ablation experiments in many scenarios. Since there is no actual ground truth for the stylised image, I suggest using the findings from Huang et al.'s published work [3] as fictitious ground truths to quantitatively assess my findings.



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.

Method

Mymethod is mainly inspired by [1]. I describe the baseline method briefly and then mention the changes that I 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).

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

λcontent and λstyle are hyperparameters. We set λcontent = 1 × e0 and λstyle = 1 × e6 for all our experiments.

Experiments

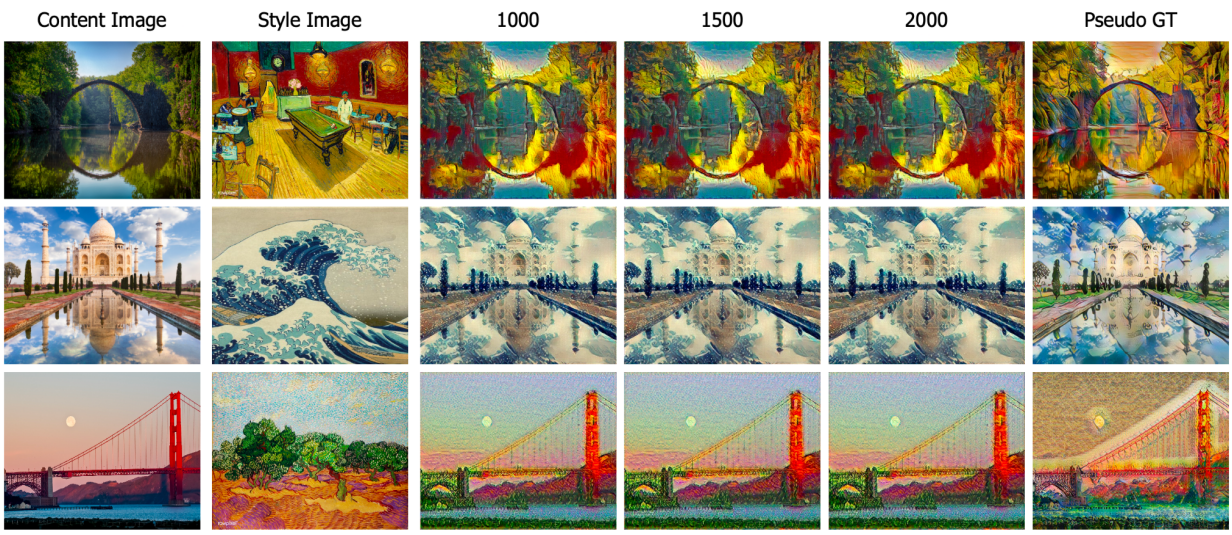
Implementation Details

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.

Optimizers

I 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, I tune the lr hyper-parameter on Pair-2 (Fig. 3). I 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. I find LBFGS to show the best performance and hence, use LBFGS for further experiments as shown.

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



Number of Iterations

I also experimented with the number of iterations for the ResNet-18 feature extractor and LBFGS optimizer. I 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. I find that the 1000 iterations gives the best performance as shown in the above figure. Thus, I 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

Conclusion and Discussion

I presented an extensive experimental analysis considering various Neural Style Transfer task factors. I also proposed to use pseudo ground truth to evaluate and compare different experimental settings. I adopt the widely used ResNet-18 to the NST task and perform better than the original baseline. Additionally, I conduct ablations on various critical performance factors like optimizers and loss terms.

Acknowledgment

I thank Dr. R. Balasubramanian at the Department of Computer Science and Engineering, IIT Roorkee for providing us with an opportunity to work on the project. This report is a part of my submission for the course project under CSN-382: 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-performing 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