Comparing Optimization Methods for Deep Learning at the Example of Artistic Style Transfer

Alexander Geng[1], Ali Moghiseh[2], Katja Schladitz[2], and Claudia Redenbach[1]

[1] University of Kaiserslautern, Gottlieb-Daimler-Straße 47, 67663 Kaiserslautern
[2] Fraunhofer Institute for Industrial Mathematics ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern

Abstract Artistic style transfer is an application of deep learning using convolutional neural networks (CNN). It combines the content of one image with the style of another one using so-called perceptual loss functions. More precisely, the training of the network consists in choosing the weights such that the perceptual loss is minimized. Here, we study the impact of the choice of the optimization method on the final transformation result. Training an artistic style transfer network with several optimization methods commonly used in deep learning, we obtain significantly differing models. In a default parameter setting, we show that Adam, AdaMax, Adam_AMSGrad, Nadam, and RMSProp yield better results than AdaDelta, AdaGrad or RProp, both measured by the perceptual loss function and by visual perception. The results of the last three methods strongly depend on the chosen parameters. With a suitable selection, AdaGrad and AdaDelta can achieve results similar to the versions of Adam or RMSProp.

Keywords Convolutional neural network, perceptual loss, stochastic gradient descent

DOI: 10.58895/ksp/1000124383-27 erschienen in:

Forum Bildverarbeitung 2020

DOI: 10.5445/KSP/1000124383 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000124383/

1 Introduction

In order to achieve artistic style transfer as first described in [1,2] in real time as desirable for live demonstrations, we train a feed forward network for style images and transfer the styles to content images as specified in [3]. The training of such a network is essentially an optimization problem. The weights of the network are chosen such that the prediction is as close as possible to the training data. We compare eight methods available in PyTorch [4]: AdaGrad [5], AdaDelta [6], RProp [7], RMSProp [8] and the four variants of Adam (Adam, AdaMax, Adam_AMSGrad [9], Nadam [10]).

2 Method

An overview of the system is visualized in Figure 2.1. It consists of two components: an *image transformation network* f_W on the left and a *loss network* ϕ on the right side, which is used to define several *loss functions*. The mapping $\hat{y} = f_W(x)$ transforms the input image x into the output image $\hat{y}$, where W are the weights of the image transformation network. We consider loss functions $\ell_{feat}(\hat{y}, y_1)$ and $\ell_{style}(\hat{y}, y_2)$ which measure the content and style differences between the transformed image $\hat{y}$ and the *target images* y_1 (content target) and y_2 (style target), respectively. In our case, the content target image y_1 is the same as the input image x. Training of the image transformation network consists in minimizing a weighted combination of the loss functions by using a suitable optimization method. We get an optimal value W^* with

$$W^* = \arg\min_W \left[\lambda_c \ell_{feat}(f_W(x), y_1) + \lambda_s \ell_{style}(f_W(x), y_2) \right], \quad (2.1)$$

where λ_c and λ_s are non-negative weight factors.

2.1 Image transformation network

The image transformation network is a deep convolutional neural network consisting of several convolutional layers with varying stride. All convolutional layers are followed by spatial batch normalization and rectified linear units (ReLU) cutting off negative parts.

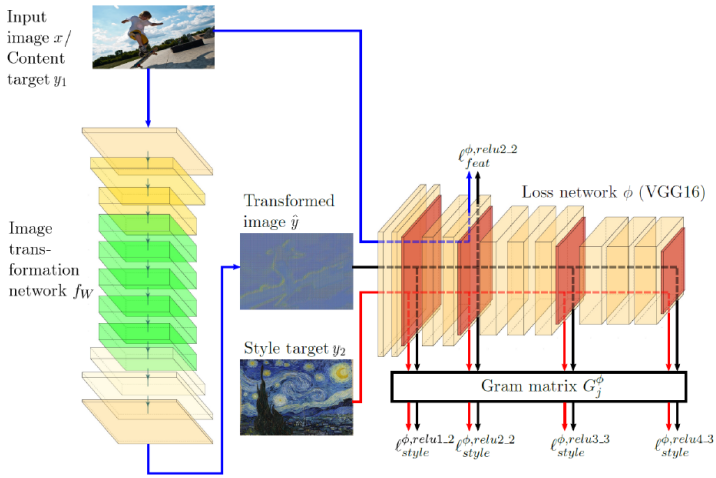


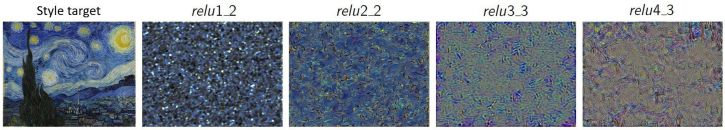
Figure 2.1: System overview. We train an image transformation network to transform input images x into output images $\hat{y}$. A loss network ϕ pre-trained for image classification is used to define loss functions. We measure the differences in content and style between the target images and the transformed image. The loss network is not changed during training.

Only the output layer uses a scaled hyperbolic tangent function instead to ensure that the output image has pixels with values in the range $[0, 255]$. These encoders and decoders are connected by residual blocks.

2.2 Perceptual loss function

We apply a perceptual loss function, derived from a pre-trained network. This loss network ϕ consists of the first four blocks of the VGG-16 network [11] pre-trained on the ImageNet dataset [12]. This dataset contains a total of over 14 million human annotated images developed for computer vision research. These are organized into around 22K sub-categories, which can be considered as sub-trees of 27 higher-level categories such as animals, plants or people. The loss network is used to define a feature reconstruction loss and a style reconstruction loss, that measure differences in content and style of

(a) Images $\hat{y}$ that minimize the feature reconstruction loss $\ell^{\phi,j}_{feat}(\hat{y}, y_1)$. An image of the Microsoft COCO dataset [13] is taken as content target y_1.



(b) Images $\hat{y}$ that minimize the style reconstruction loss $\ell^{\phi,j}_{style}(\hat{y}, y_2)$. Vincent van Gogh's painting *The Starry Night* [14] is taken as style target y_2.

Figure 2.2: Reconstruction from different layers of the pretrained VGG-16 loss network ϕ. Input image is a white noise image.

the transformed image and the content and style targets, respectively. Finally, the combination of these two losses is minimized.

The feature reconstruction loss is defined as the squared, normalized Euclidean distance (mean squared error) between feature representations

$$\ell^{\phi,j}_{feat}(\hat{y}, y_1) = \frac{1}{H_j W_j C_j} \|\phi_j(\hat{y}) - \phi_j(y_1)\|_2^2, \tag{2.2}$$

where $\phi_j(x)$ is a feature map of layer j with shape $H_j \times W_j \times C_j$. In this case, H_j and W_j represent the height and width, respectively, and C_j the number of channels of the feature map. The reconstruction of images from the first layers of the loss network provides images that are perceptually similar to the target image, but that do not necessarily fit exactly, see Figure 27.2(a). We use the feature map at layer $j =$ relu2_2 of the loss network to calculate the feature reconstruction loss in Equation (2.2).

In addition to the content of the target image, the style of another image has to be met. However, the difference of two images in style is not as simple to represent as the in difference in content. Copying the feature reconstruction loss would result in comparing the content

of the style image with the output image $\hat{y}$, which is not our aim. In order to extract the style representation of the style image, only, we use the Gram matrix $G_j^\phi(x)$ to find the correlation of the channels (features) of a feature map. This approach is based on the assumption that the style of the image is defined through the co-occurrence of particular features. As in the loss function above, let $\phi_j(x)$ be the outcome of the network ϕ at layer j for the input x, which is a $H_j \times W_j \times C_j$ feature map. Then, the *Gram matrix* $G_j^\phi(x)$, which has a size of $C_j \times C_j$, is defined by

$$G_j^\phi(x)_{c,c'} = \frac{1}{H_j W_j C_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}, \tag{2.3}$$

where $c, c' \in [1, \ldots, C_j]$. Thus, we get the style reconstruction loss via

$$\ell_{style}^{\phi,j}(\hat{y}, y_2) = \|G_j^\phi(\hat{y}) - G_j^\phi(y_2)\|_F^2, \tag{2.4}$$

i. e., the squared Frobenius norm of the difference of the Gram matrices of output and target image.

Reconstruction from higher layers of the loss network transfers larger scale structure from the target image (see Figure 27.2(b)). We use this fact to reconstruct style from a set of layers J instead of a single layer j and define $\ell_{style}^{\phi,J}(\hat{y}, y)$ as the sum of losses for each layer $j \in J$. We combine the four layers *relu*1_2, *relu*2_2, *relu*3_3, and *relu*4_3 of the VGG-16 loss network ϕ for the style reconstruction loss, using all available information. Hence, we set $J = \{relu1\_2, relu2\_2, relu3\_3, relu4\_3\}$ and get the following optimization task

$$\hat{y} = \arg\min_y \left[\lambda_c \ell_{feat}^{\phi,relu2\_2}(y, y_1) + \lambda_s \ell_{style}^{\phi,J}(y, y_2) \right]. \tag{2.5}$$

Here, $\lambda_c > 0$ is a content and $\lambda_s > 0$ a style weight factor. These weights have to be adjusted carefully by trial-and-error, in our case $\lambda_c = 10^5$ and $\lambda_s = 10^{10}$. To solve Equation (2.5), we use several optimization methods.

2.3 Optimization methods

We focus on comparing extensions of the stochastic gradient descent method (SGD) [15]. In SGD, the step size or learning rate in each iteration is initially selected and kept fix. The first improvement to SGD is AdaGrad, which adjusts the learning rate dynamically based on all gradients observed before. AdaDelta restricts the number of accumulated past gradients to a fixed number, instead of accumulating all past gradients. It has been developed to avoid the radical decay of learning rates observed in AdaGrad. In RProp, the idea of only using the sign of the gradient is combined with the idea of adapting the step size individually for each weight. However, the particular gradient is not available. This is improved by the use of the moving average in RMSProp, which has been developed independently of AdaGrad. It also keeps the estimates of the squared gradients, but uses a moving average instead of continually accumulating them. Finally, Adam and its variants are very popular in style transfer. Adam is similar to RMSProp and AdaDelta, but uses an exponentially decaying average of the past gradients. Compared to Adam, AdaMax scales the gradients inversely proportional to the L^∞ norm instead of the L^2 norm of the past gradients. Adam_AMSGrad maintains the maximum of all exponentially decaying averages of the gradients until the present time step and uses this maximum in place of the actual one. Nadam is a combination of Adam and Nesterov's momentum method [16].

We train on the Microsoft COCO dataset of the year 2017 [13]. It contains a total of over 123K images with annotations belonging to 80 object categories. In each step we update the weights of the image transformation network.

3 Outcome

We train the model for two epochs with default learning rate $\eta = 0.001$ and batch size $bs = 4$ recommended in [17]. During the training process, the optimal weights of the image transformation network for a style image are determined. For prediction, we pass a content image through this network and get the results for the eight considered optimization methods as shown in Figure 3.1. We take

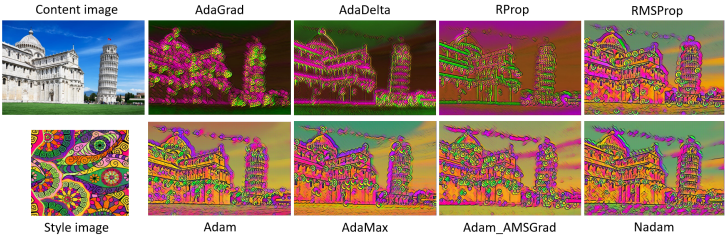
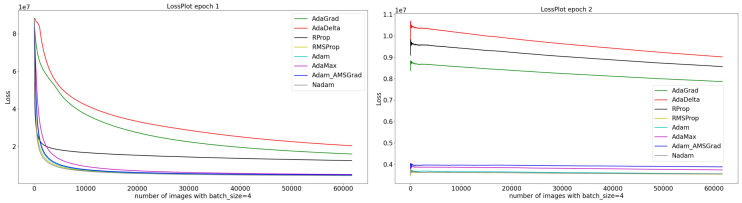


Figure 3.1: Content image (1 000 × 668), style image (800 × 800) and visualization of stylized content image with various optimizers (each 1 000 × 668).



(a) Epoch 1. (b) Epoch 2.

Figure 3.2: Loss plots for two epochs using the eight optimization methods.

an image of the Leaning Tower of Pisa [18] as content image and a colorful pattern [19] as style image. Visually, the differences are quite strong. AdaDelta, AdaGrad or RProp result in rather dark images whose content is not as well visible as in the results of the Adam versions or RMSProp. The loss plots for the eight methods over the two training epochs also clearly show the differences, see Figure 3.2. To investigate the dependence of the solution on the hyperparameters of the optimization method, we vary the learning rate $\eta \in \{0.1, 0.01, 0.001, 0.0001\}$ and batch size $bs \in \{1, 2, 4, 8\}$. The best setting and the corresponding loss values and training times are shown in Table 1. The adjusted parameters result in more similar images except for AdaGrad and RProp. The loss value for the RProp result differs by a factor of almost two from the loss values obtained by the other methods. This is also reflected in the updated results in Figure 3.3. The training times yield a similar picture: All methods

Table 1: Summary of the selected parameters, the loss values, and training times for two epochs training using the eight optimization methods.

Optimization method	η	bs	Loss value	Training time (in hours)
AdaGrad	0.01	1	$4.2293 \cdot 10^6$	5.6
AdaDelta	0.1	1	$3.3681 \cdot 10^6$	6.0
RProp	0.0001	1	$6.0191 \cdot 10^6$	8.6
RMSProp	0.001	1	$3.3217 \cdot 10^6$	5.9
Adam	0.001	1	$3.5556 \cdot 10^6$	5.8
AdaMax	0.001	1	$3.4540 \cdot 10^6$	5.8
Adam_AMSGrad	0.001	2	$3.5892 \cdot 10^6$	5.9
Nadam	0.001	2	$3.4687 \cdot 10^6$	6.0

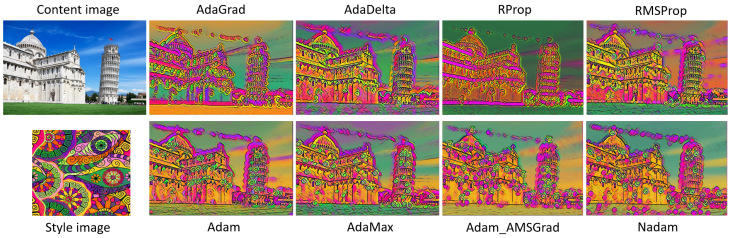


Figure 3.3: Content image (1000×668), style image (800×800) and visualization of stylized content image with different optimizer and adjusted parameters (each 1000×668).

take approximately 6 hours or less, while the training with RProp requires 8.6 hours.

Differences of the optimization methods show with respect to parameter selection, too. The Adam versions or RMSProp lead to similar loss values and stylized images, even if the selected learning rates and batch sizes are not optimal, whereas for AdaGrad, AdaDelta, and RProp the loss values depend strongly on the chosen parameters and can exceed the optimal ones by far. Comparing Figures 3.1 and 3.3 also clearly shows these differences.

4 Summary

The choice of the optimization method can be decisive for the result of the artistic style transfer. It is advisable to use one of the Adam versions or RMSProp which are more robust with respect to parameter choice than the other methods considered here. Even though we have used loss functions for measuring the quality of style reproduction, the evaluation of "style" is rather subjective and, hence, hard to measure accurately. Thus, in the next step, we plan to investigate the effect of the optimization method on a segmentation problem where differences can be quantified more explicitly.

5 Acknowledgements

This research was supported by the Fraunhofer FLAGSHIP PROJECT ML4P.

References

1. L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *Journal of Vision*, vol. 16, no. 12, p. 326, 2016.
2. ——, "Image style transfer using convolutional neural networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, 2016.
3. J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," *Lecture Notes in Computer Science*, vol. 9906, pp. 694–711, 2016.
4. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *ArXiv*, vol. abs/1912.01703, 2019.
5. J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011.
6. M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *ArXiv*, vol. abs/1212.5701, 2012.

7. M. A. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," *IEEE International Conference on Neural Networks*, pp. 586–591 vol.1, 1993.
8. T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
9. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.
10. T. Dozat, "Incorporating Nesterov Momentum into Adam," *ICLR Workshop*, 2016.
11. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.
12. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *ArXiv*, vol. abs/1409.0575, 2014.
13. T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," *Lecture Notes in Computer Science*, 2014.
14. V. van Gogh, "The Starry Night," Museum of Modern Art, New York, 1889. [Online]. Available: https://www.vangoghgallery.com/catalog/Painting/508/Starry-Night.html
15. H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.
16. A. Botev, G. Lever, and D. Barber, "Nesterov's accelerated gradient and momentum as approximations to regularised update descent," *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1899–1903, 2017.
17. P. Saini, "StyleTransferApp," GitHub repository, 2019. [Online]. Available: https://github.com/puneet29/StyleTransferApp
18. P. la Quiete, "Torre pendente di Pisa," https://poderelaquiete.it/, [accessed: September 20, 2020].
19. Colourbox, "Nahtlose bunte Muster, Stock-Vektor," https://www.colourbox.de/vektor/nahtlose-bunte-muster-vektor-7143090, [accessed: September 20, 2020].