Conditional Neural Style Transfer with Peer-Regularized Feature Transform

Jan Svoboda^{1,2}, **Asha Anoosheh**^{1}, **Christian Osendorfer**^{1}, **Jonathan Masci**^{1}
^{1}NNAISENSE, Switzerland ^{2}Università della Svizzera italiana, Switzerland
{jan.svoboda,asha.anoosheh,christian.osendorfer,jonathan.masci}@nnaisense.com

Abstract

This paper proposes a neural style transfer model to generate a stylized image conditioning only on a set of examples describing the desired style. At its core a novel graph-based Peer-Regularization layer to recompose style in latent space and to achieve better separation of style and content. In contrast to the vast majority of existing solutions, our model does not require any pre-trained networks for computing perceptual losses and can be trained end-to-end with cyclic losses operating directly in latent space — not on the stylized RGB image. The qualitative results of our graph-based architecture are competitive with respect to the current state-of-the-art, especially in the challenging zero-shot setting, and open the door to more abstract and artistic neural image generation.

1 Introduction

Neural style transfer (NST), introduced by the seminal work of Gatys [3], is an area of research that focuses on models that transform the visual appearance of an input image (or video) to match the style of a desired target image. NST has seen an exponential growth within the deep learning community and reached a wide spectrum of applications e.g. converting time-of-day [26, 8], mapping among artwork and photos [1, 26, 8], transferring facial expressions [11], transforming between animal species [26, 8], etc.

Despite their popularity and good-quality results, current NST approaches are still quite limited. The original formulation of Gatys, for instance, *et al.* requires a new optimization for each transfer, making it impractical for many real-world scenarios. Trying to resolve the problem, of notable mention is the work of Johnson *et al.* [9] and Ulyanov *et al.* [21], who also later introduced Instance Normalization [22], a popular feature-normalization scheme for style transfer. Other recent works attempt to separate style and content in feature space (latent space) to allow generalization to a style characterized by an additional input image, or set of images. The most widespread work in this family is AdaIN [7], a particular case of FiLM [17],that normalizes the features to match moments from the target image – acting as global style encoding. The current state-of-the-art allows, among other things, to control the amount of stylization applied, interpolating between different styles, and using masks to convert different regions of image into different styles [7, 18].

Beyond the study of new architectures for NST, loss functions for training these have been improved: the perceptual loss [3, 9] with a pre-trained VGG19 classifier [19] is very commonly used in this task as it, supposedly, captures high-level features of the image. However, this assumption has been recently challenged in [4]. Cycle-GAN [26] proposed a new cycle consistent loss that requires no one-to-one correspondence between the input and the target images, thus lifting the heavy burden of data annotation.

This paper addresses the NST setting where style defined by a set of input images to allow arbitrary domain transfer.

Zero-shot style transfer is achieved by by introducing a novel feature regularization layer capable of recomposing global and local style content from the input style image based on a dynamic k-NN graph in feature space. We demonstrate that this inductive bias allows better separation of style and content in latent space, instead of encoding the information in its weights, a core problem for expressive parametric models [13]. Extensive experiments, and an ablation study, successfully show that the proposed model compares favorably with state-of-the-art and that the encoded style information is used by the model and not learned in the weights.

2 Method

The core idea of our work is a region-based mechanism to exchange the style between input and target images via Peer Regularization, with the aim of obtaining disentangled style and content representations. Given that the inductive bias of the architecture is not strong enough to achieve a good level of separation from limited data, we advocate the use of metric learning to directly enforce separation of styles, which has been experimentally shown to greatly reduce the amount of style dependent information that gets encoded in the decoder.

2.1 Architecture and losses

The proposed architecture is shown in Figure 1. To prevent the generator from encoding the stylization in its weights auxiliary decoders [2] are used during training to optimize the parameters of the encoder and decoder independently. The yellow module in Figure 1 is trained as an autoencoder [15, 25, 14].while the green module is trained as a GAN [6] to generate the stylized version of the input using the encoder with fixed parameters. The optimization of both modules is interleaved together with the discriminator. Additionally, following the analysis from Martineau *et al.*. [10], the Relativistic Average GAN (RaGAN) is used as our adversarial loss formulation, which has shown to be more stable and to produce more natural-looking images than traditional losses. Let us now describe the four main building blocks of our approach in detail[1].

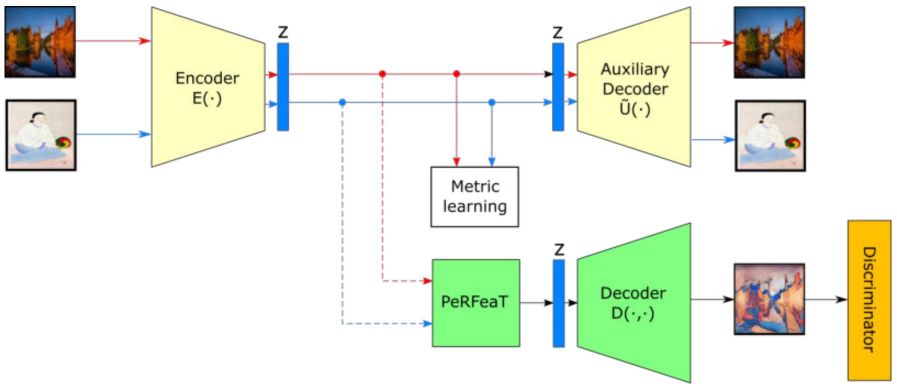


Figure 1: The proposed architecture with two decoder modules. The yellow decoder is the auxiliary generator, while the main generator is depicted in green. Dashed lines indicate lack of gradient propagation.

Encoder. The encoder used to produce latent representation of all input images is composed of several strided-convolutional layers for downsampling followed by multiple ResNet blocks. The latent code z is composed by two parts: the content part, $(z)_C$, which holds information about the image content (e.g. objects, position, scale, etc.), and the style part, $(z)_S$, which encodes the style that the content is presented in (e.g. level of detail, shapes, etc.). The style component of the latent code $(z)_S$ is further split equally into $(z)_S = [(z)_S^{loc}, (z)_S^{glob}]$. Here, $(z)_S^{loc}$ encodes local style information per pixel of each feature map, while $(z)_S^{glob}$ undergoes further downsampling via a small sub-network to generate a single value per feature map to encode for global style information.

[1]The full architecture details are found in the supplementary material.

Auxiliary generator. The Auxiliary generator reconstructs an image from its latent representation and is used only to train the encoder module. It is composed of several Resnet blocks followed by fractionally-strided convolutional layers to generate the image. The loss is composed of a metric learning loss $L_{z_{style}}$, enforcing clustering of the style part of the latent representations, a latent cycle loss $\tilde{L}_{z_{cycle}}$, and a classical reconstruction loss $\tilde{L}_{idt}$. It is defined as follows:

$$L_{\tilde{G}} = L_{z_{style}} + \tilde{L}_{z_{cycle}} + \lambda \tilde{L}_{idt} \tag{1}$$

Main generator. This network replicates the architecture of the auxiliary generator, and uses the output of the Peer Regularized Feature Transform module (see Section 2.2). During training of the main generator the encoder is kept fixed, and the generator is optimized using loss that consists of a latent cycle loss $L_{z_{cycle}}$, image reconstruction loss L_{idt} and GAN generator loss L_{gen}.

$$L_G = L_{gen} + L_{z_{cycle}} + \lambda L_{idt} \tag{2}$$

Discriminator. The discriminator is a convolutional network receiving two images concatenated over the channel dimension and producing an $N \times N$ map of predictions. The first image is the one being discriminated, whereas the second serves as conditioning for the style class. The output prediction is ideally 1 if the two inputs come from the same style class and 0 otherwise.

2.2 Peer Regularized Feature Transform (PeRFeaT)

The PeRFeaT module draws inspiration from PeerNets [20] and Graph Attention Layer (GAT) [24] and performs style transfer in latent space taking advantage of the separation of content and style information. It receives $z_i = [(z_i)_C, (z_i)_S]$ and $z_t = [(z_t)_C, (z_t)_S]$ as an input and computes the k-Nearest-Neighbors (k-NN) between $(z_i)_C$ and $(z_t)_C$ using the Euclidean distance to induce a graph of peers.

Attention coefficients over the graph nodes are computed and used to recompose the style portion of $(z_{out})_S$ as convex combination of its nearest neighbors representations. The content portion of the latent code remains instead unchanged, resulting in: $z_{out} = [(z_i)_C, (z_{out})_S]$.

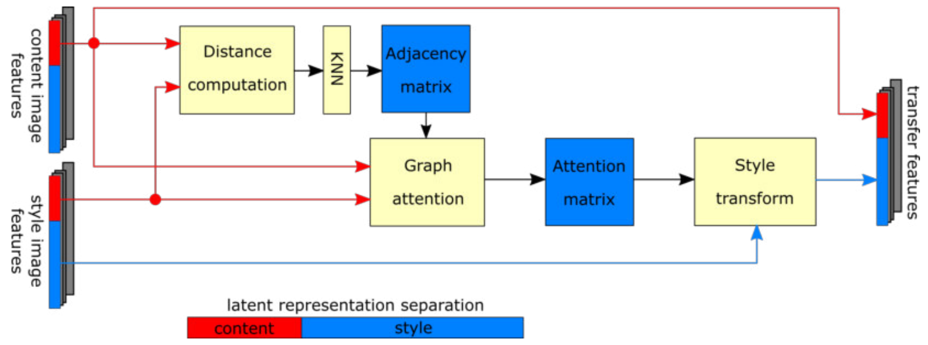
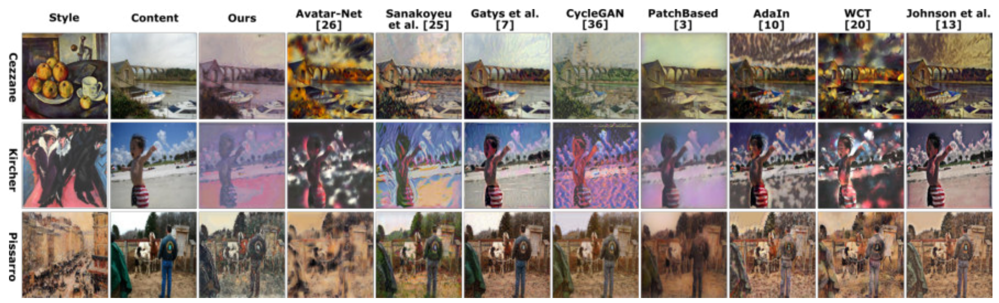


Figure 2: The content part of the latent representation is used to induce a graph structure on the style latent space, which is then used to recombine the style part of the content image's latent representation from the style image's latent representation.

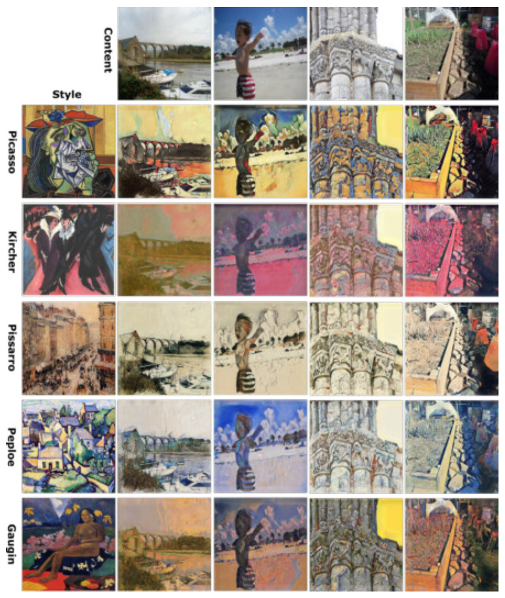
Given a pixel $(\mathbf{z_m})_\mathbf{C}$ of feature map m, its k-NN graph in the space of d-dimensional feature maps of all pixels of all peer feature maps n_k is considered. The new value of the style part $(z)_S$ for the pixel is expressed as:

$$(\tilde{\mathbf{z}}_p^m)_S = \sum_{k=1}^{K} \alpha_{mn_kpq_k} (\mathbf{z}_{q_k}^{n_k})_S, \quad \alpha_{mn_kpq_k} = \frac{\mathrm{LeakyReLU}(\exp(a((\mathbf{z}_p^m)_C, (\mathbf{z}_{q_k}^{n_k})_C)))}{\sum_{k'=1}^{K} \mathrm{LeakyReLU}(\exp(a((\mathbf{z}_p^m)_C, (\mathbf{z}_{q_{k'}}^{n_{k'}})_C)))} \tag{3}$$

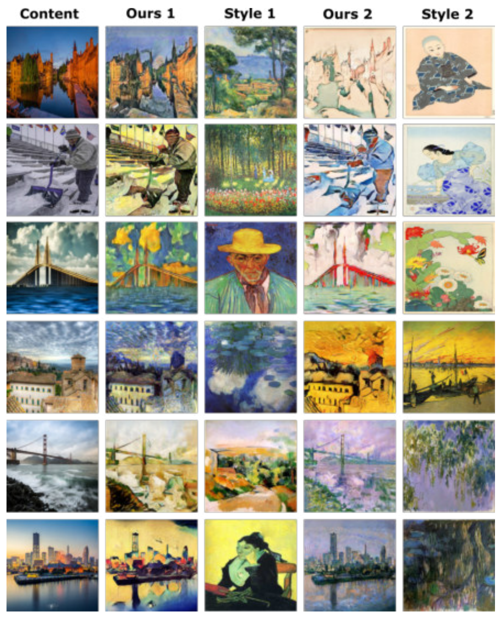
where $a(\cdot,\cdot)$ denotes a fully connected layer mapping from $2d$-dimensional input to scalar output, and $\alpha_{mn_kpq_k}$ are attention scores measuring the importance of the q_kth pixel of feature map n to the output pth pixel $\tilde{\mathbf{x}}_p^m$ of feature map m. The resulting style component of the input feature map $\tilde{\mathbf{X}}^m$ is the weighted average, pixel-wise, of its peer pixel features defined over the style input image.



(a) Comparison wrt. state-of-the-art.



(b) Zero-shot style transfer.



(c) Styles seen during training.

Figure 3: Comparison wrt. current state-of-the-art methods (not all methods are zero-shot as ours). Qualitative evaluation of our method for previously unseen styles (bottom left) and for styles seen during training (bottom right).

3 Experimental setup and Results

The proposed approach is compared against the state-of-the-art on extensive qualitative evaluations.[2] Only qualitative comparisons are provided as no standard quantitative evaluations are defined for NST algorithms.

The dataset of [26], composed of a collection of photographs and four different painter collections is used for training the model. In particular, the datasets named *monet2photo*, *cezzane2photo*, *vangogh2photo* and *ukiyoe2photo* are combined into a single dataset named *painter2photo*, consisting of 6280 real photos and 2560 paintings in total. The training loss is defined as $L = L_D + L_G + L_{\tilde{G}}$ where D is the discriminator, G the main generator, and $\tilde{G}$ is the auxiliary generator (see Section 2.1). ADAM [12] is used as optimizer with learning rate set to 2×10^{-4} and batch size of 1, training is performed for total of 200 epochs. In each epoch, all the real photos are visited, which results in 6280 iterations per epoch. The identity loss and the metric learning weighting are set for all experiments to, respectively, 25 and 4. Images are cropped and resized to 256×256 pixels. Note that during testing, our method can operate on images of arbitrary size. Qualitative results, including comparison of zero-shot style transfer w.r.t. some of the state-of-the-art is shown in Figure 3.

[2]More results and ablation studies supporting our design choices are shown in the supplementary material.

References

[1] Asha Anoosheh, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. Combogan: Unrestrained scalability for image domain translation. *CoRR*, abs/1712.06909, 2017.

[2] Jeffrey De Fauw, Sander Dieleman, and Karen Simonyan. Hierarchical autoregressive image models with auxiliary decoders. *CoRR*, abs/1903.04933, 2019.

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[4] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.

[5] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *NIPS*. 2014.

[7] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR*, abs/1703.06868, 2017.

[8] Xun Huang, Ming-Yu Liu, Serge J. Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *CoRR*, abs/1804.04732, 2018.

[9] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[10] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. *CoRR*, abs/1807.00734, 2018.

[11] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] Julius Kunze, Louis Kirsch, Hippolyt Ritter, and David Barber. Noisy information bottlenecks for generalization, 2019.

[14] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. *CoRR*, 2016.

[15] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. *Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction*, pages 52–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[16] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[17] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. *CoRR*, abs/1709.07871, 2017.

[18] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *CVPR*, 2018.

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

[20] Jan Svoboda, Jonathan Masci, Federico Monti, Michael M. Bronstein, and Leonidas J. Guibas. Peernets: Exploiting peer wisdom against adversarial attacks. *CoRR*, abs/1806.00088, 2018.

[21] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *CoRR*, abs/1603.03417, 2016.

[22] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *CoRR*, abs/1701.02096, 2017.

[23] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. 2008.

[24] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Alejandro Romero, Pietro Lió, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2018.

[25] Junbo Jake Zhao, Michaël Mathieu, Ross Goroshin, and Yann LeCun. Stacked what-where auto-encoders. *CoRR*, 2015.

[26] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

A Network architecture

This section describes our model in detail. We describe the generator and discriminator in separate sections below.

A.1 Generator

Detailed scheme of the architecture is depicted in Figure 4. Each of the convolutional layers (in yellow) is followed by Instance Normalization (IN) [22] and ReLU nonlinearity [16]. The PeRFeaT module uses Peer Regularization Layer [20] with Euclidean distance metric, k-NN with $K = 5$ nearest neighbours and dropout on the attention weights of 0.2.

The generated latent code are 768 feature maps of size 64×64. First 256 feature maps is the content latent representation, while the remaining 512 is for the style. The style latent representation is further split into halves, having first 256 feature maps left unchanged and the second 256 feature maps are passed through the *Global style transform* block producing feature maps of size 1×1 that hold the global part of the style latent representation.

The last convolutional block of the decoder is equipped with TanH nonlinearity and produces the reconstructed RGB image.

The auxiliary generator copies the architecture of the main generator, while omitting the *Style transfer* block (see Figure 4).

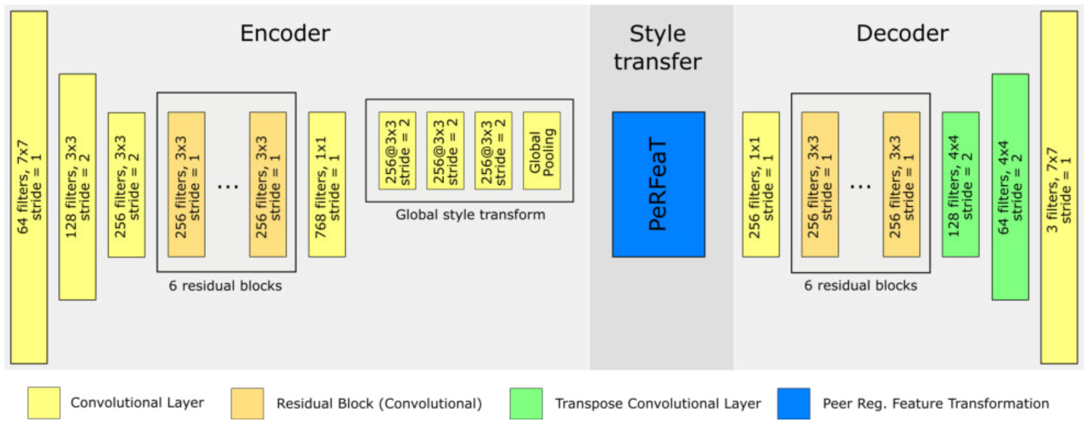

Encoder
64 filters, 7x7 stride = 1
128 filters, 3x3 stride = 2
256 filters, 3x3 stride = 2
256 filters, 3x3 stride = 1
...
256 filters, 3x3 stride = 1
6 residual blocks
768 filters, 1x1 stride = 1
256@3x3 stride = 2
256@3x3 stride = 2
256@3x3 stride = 2
Global Pooling
Global style transform
Style transfer
PeRFeaT
Decoder
256 filters, 1x1 stride = 1
256 filters, 3x3 stride = 1
...
256 filters, 3x3 stride = 1
6 residual blocks
128 filters, 4x4 stride = 2
64 filters, 4x4 stride = 2
3 filters, 7x7 stride = 1
Convolutional Layer
Residual Block (Convolutional)
Transpose Convolutional Layer
Peer Reg. Feature Transformation

Figure 4: Detailed architecture of generator.

A.2 Discriminator

The discriminator architecture is shown in Figure 5. It takes two RGB images concatenated over the channel dimension as input and produces a 64×64 map of predictions. Our implementation uses LS-GAN and therefore there is no Sigmoid activation at the output

To stabilize the discriminator training, we add random Gaussian noise to each input:

$$X = X + N(\mu, \sigma), \tag{4}$$

where N is a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.1$.

B Loss function

We denote x_i, x_t, x_f an input image, a target and fake image, respectively. Our model consists of an encoder $E(\cdot)$ generating the latent representations, an auxiliary decoder $\tilde{D}(\cdot)$ taking a single latent code as input, and a main decoder $D(\cdot,\cdot)$ taking two latent codes as inputs. Generated latent codes are denoted $z_I = E(x_i), z_t = E(x_t)$. We further denote $(\cdot)_C, (\cdot)_S$ the content and style part of the latent code, respectively. Additionally, to impose a stronger prior on the feature similarity expected while performing $E(D(E(x)))$, a cycle loss is used on the encoder's middle-layer features maps, denoted z', which have double the spatial size of their subsequent latent representations.

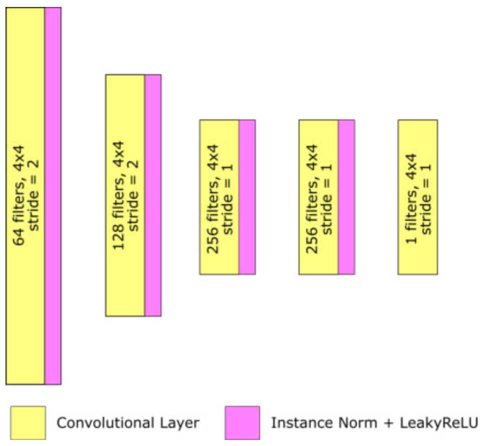

64 filters, 4x4 stride = 2
128 filters, 4x4 stride = 2
256 filters, 4x4 stride = 1
256 filters, 4x4 stride = 1
1 filters, 4x4 stride = 1
Convolutional Layer
Instance Norm + LeakyReLU

Figure 5: Detailed architecture of discriminator.

The distance f between two latent representations is defined as the smooth L1 norm [5] in order to stabilize training and to avoid exploding gradients:

$$f[d] = \frac{1}{W \times H} \sum_{i=1}^{W \times H} \sum_{j=1}^{C} d_{i,j}, \qquad d_{i,j} = \begin{cases} 0.5 d_{i,j}^2 & \text{if } d_{i,j} < 1 \\ |d_{i,j}| - 0.5 & \text{otherwise} \end{cases}, \tag{5}$$

where $d = z_1 - z_2$, z_1 and z_2 are two different feature embeddings with C channels and $W \times H$ is the spatial dimension of each channel.

Encoder. The encoder used to produce latent representation of all input images is composed of several strided-convolutional layers for downsampling followed by multiple ResNet blocks. The latent code z is composed by two parts: the content part, $(z)_C$, which holds information about the image content (e.g. objects, position, scale, etc.), and the style part, $(z)_S$, which encodes the style that the content is presented in (e.g. level of detail, shapes, etc.). The style component of the latent code $(z)_S$ is further split equally into $(z)_S = [(z)_S^{loc}, (z)_S^{glob}]$. Here, $(z)_S^{loc}$ encodes local style information per pixel of each feature map, while $(z)_S^{glob}$ undergoes further downsampling via a small sub-network to generate a single value per feature map.

Auxiliary generator. The Auxiliary generator simply reconstructs an image from its latent representation and is used only during training to train the encoder module. It is composed of several ResNet blocks followed by fractionally-strided convolutional layers to reconstruct the original image. The loss is composed of a metric learning loss, enforcing clustering of the style part of the latent representations, a latent cycle loss, and a classical reconstruction loss. It is defined as follows:

$$\begin{aligned} L_{z_{style}}^{pos} &= f[(z_{i_1})_S - (z_{i_2})_S] + f[(z_{t_1})_S - (z_{t_2})_S] \\ L_{z_{style}}^{neg} &= f[(z_{i_1})_S - (z_{t_1})_S] + f[(z_{i_2})_S - (z_{t_2})_S] \\ L_{z_{style}} &= L_{z_{style}}^{pos} + max(0.0, \mu - L_{z_{style}}^{neg}) \end{aligned} \tag{6}$$

$$\begin{aligned} \tilde{L}_{z_{cycle}} = {} & f[E(\tilde{D}(z_i))_C - (z_i)_C] + f[E(\tilde{D}(z_i))_S - (z_i)_S] \\ & + f[E(\tilde{D}(z_t))_S - (z_t)_S] \\ & + f[E(\tilde{D}(z_i')) - z_i'] + f[E(\tilde{D}(z_t')) - z_t'] \\ & + f[E(D(z_i, z_t))_C - (z_i)_C] + f[E(D(z_i, z_i))_C - (z_i)_C] \end{aligned} \tag{7}$$

$$\tilde{L}_{idt} = f[\tilde{D}(E(x_i)) - x_i] + f[\tilde{D}(E(x_t)) - x_t] \tag{8}$$

$$L_{\tilde{G}} = L_{z_{style}} + \tilde{L}_{z_{cycle}} + \lambda \tilde{L}_{idt} \tag{9}$$

Main generator. This network replicates the architecture of the auxiliary generator, and uses the output of the Peer Regularized Feature Transform module (see Section 2.2). During training of the main generator the encoder is kept fixed, and the generator is optimized using the following loss:

$$L_{gen} = \mathbb{E}_{x_i \sim \mathbb{P}} \left[g_1 \left(C(x_i) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f) \right) \right] + \mathbb{E}_{x_f \sim \mathbb{Q}} \left[g_2 \left(C(x_f) - \mathbb{E}_{x_i \sim \mathbb{P}} C(x_i) \right) \right]. \tag{10}$$

$$\begin{aligned} L_{z_{cycle}} = & f[E(D(z_i, z_t))_C - (z_i)_C] + f[E(D(z_i, z_i))_C - (z_i)_C] \\ & + f[E(D(z_i, z_t))_S - (z_t)_S] + f[E(D(z_i, z_i))_S - (z_i)_S] \\ & + f[E(D(z_t, z_t))_S - (z_t)_S] \\ & + f[E(D(z_i', z_i')) - z_i'] + f[E(D(z_t', z_t')) - z_t'] \end{aligned} \tag{11}$$

$$L_{idt} = f[D(z_i, z_i) - x_i] + f[D(z_t, z_t) - x_t] \tag{12}$$

$$L_G = L_{gen} + L_{z_{cycle}} + \lambda L_{idt} \tag{13}$$

Discriminator. The discriminator is a convolutional network receiving two images concatenated over the channel dimension and producing an $N \times N$ map of predictions. The first image is the one to discriminate, whereas the second one serves as conditioning for the style class. The output prediction is ideally 1 if the two inputs come from the same style class and 0 otherwise. The discriminator loss is defined as:

$$L_D = \mathbb{E}_{x_i \sim \mathbb{P}} \left[f_1 \left(C(x_i) - \mathbb{E}_{x_f \sim \mathbb{Q}} C(x_f) \right) \right] + \mathbb{E}_{x_f \sim \mathbb{Q}} \left[f_2 \left(C(x_f) - \mathbb{E}_{x_i \sim \mathbb{P}} C(x_{\underline{i}}) \right) \right], \tag{14}$$

where $\mathbb{P}$ is the distribution of the real data and $\mathbb{Q}$ is the distribution of the generated (fake) data.

C Style transfer results

This section provides more qualitative results of our style transfer approach that did not fit in the main text. Figure 6 shows more exhaustive comparison wrt. most of the state-of-the-art. Figure 7 are images generated with resolution 768×768 and shows the generalization of our approach to different styles and ability of our approach to perform zero-shot style transfer. From the set of painters that are shown, only Cezzane and Van Gogh were seen during training. In addition, images in Figure 8 were generated with resolution 256×256 and show results of transfer taking a random painting from the test set of painters that were seen during training (Cezzane, Monet, Van Gogh, Ukiyoe).

C.1 Ablation study

There are several key components in our solution which make arbitrary style transfer with a single model and end-to-end training possible. Namely, the *auxiliary decoder* used during training, which prevents degenerate solutions. *Separation of the latent code* into content and style part, which allows to transfer the style features while preserving the content. Last but not least, *metric learning* on the latent space style class separation which allows to cluster different styles in the latent space. The effect of suppressing each of them during the training is examined, and results for the various models are compared, highlighting the importance of each component in Figure 9.

D Latent space structure

Our latent representation is split into two parts, $(z)_C$ and $(z)_S$, content and style respectively. Metric learning loss is used on the style part in order to enforce a separation of different modalities in the style latent space.

$$\begin{aligned} L^{pos}_{z_{style}} &= f[(z_{i_1})_S - (z_{i_2})_S] + f[(z_{t_1})_S - (z_{t_2})_S] \\ L^{neg}_{z_{style}} &= f[(z_{i_1})_S - (z_{t_1})_S] + f[(z_{i_2})_S - (z_{t_2})_S] \\ L_{z_{style}} &= L^{pos}_{z_{style}} + max(0.0, \mu - L^{neg}_{z_{style}}) \end{aligned} \tag{15}$$

where $(z_{i1})_S$, $(z_{i2})_S$ are style parts of latent representations of two different input images and $(z_{t1})_S$, $(z_{t2})_S$ are style parts of latent representations of two different targets from the same target class. Parameter $\mu = 4$ and it is the margin we are enforcing on the separation of the positive and negative scores.

Figure 10 shows 2D embedding of the style latent space generated using T-SNE [23], where different colors represent different style classes. One can observe that the photos can be separated from the painters very well. Separation of different painters into good clusters is rather difficult problem in very low dimensional space. We see that only partial separation is learned for the other styles (different painters).

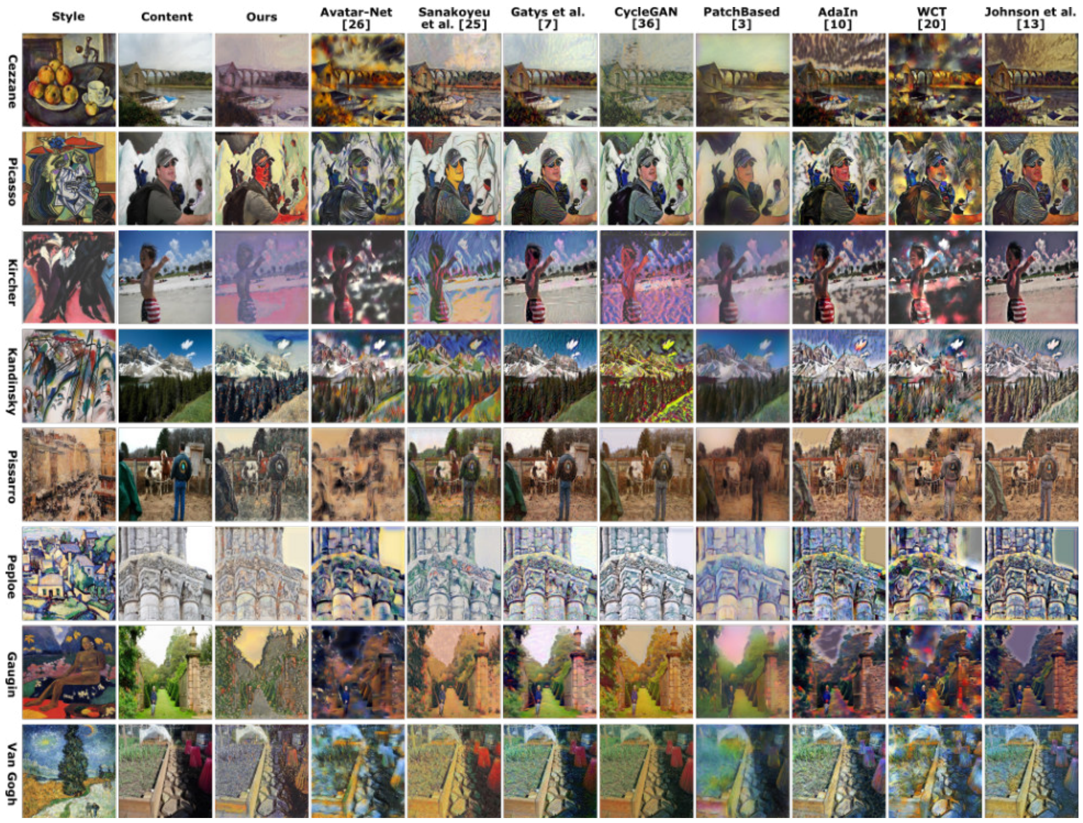


Figure 6: Qualitative comparison with respect to other state-of-the-art methods. It should be noted that most of the compared methods had to train a new model for each style. While providing competitive results, our method performs zero-shot style transfer using a single model.

D.1 Embedding ablation study

The T-SNE embeddings are shown for our ablation study experiments as well. This clearly demonstrates that without the metric learning loss, no clustering of the latent space is learned. Furthermore, without using the auxiliary decoder, the latent space is partially clustered thanks to the metric learning loss, however the decoder cannot make any use of it. This suggests that all the stylization is hard-coded in the decoder weights and the style information of the latent representation is not taken into account. Lastly, we can observe that for the experiment where we transfer the whole latent code, latent space is again clustered. The content is however preserved only partially. This is due to the fact that we compute the adjacency matrix and attention only on the content part, but then we reconstruct the whole latent code as a convex combination of neighbouring nodes, instead of doing so only for the style part, and preserving the original content part.

D.2 Visualization in image space

Figure 11 visualizes the influence of the $(z)_C$ and $(z)_S$ parts of the latent representation after decoding back into the RGB image space. The PeRFeaT transformation, which performs the style transfer, is executed first. The resulting latent code is then modified before feeding it to the decoder. Replacing the $(z)_C$ with 0's gives us some rough representation of the style with only approximate shapes. On the other hand, if we replace $(z)_S$ with 0's and we keep $(z)_C$, a rather flat representation of the input with very sharp edges is reconstructed. This confirms that $(z)_C$ focuses on the content, while $(z)_S$ holds most of the stylization information.

The fact that PeRFeaT transformation is done first means that the resulting style is mapped to the content of the content image. As a result, the decoded image slightly resembles the structure of the content image even if the $(z)_C$ is set to 0's.

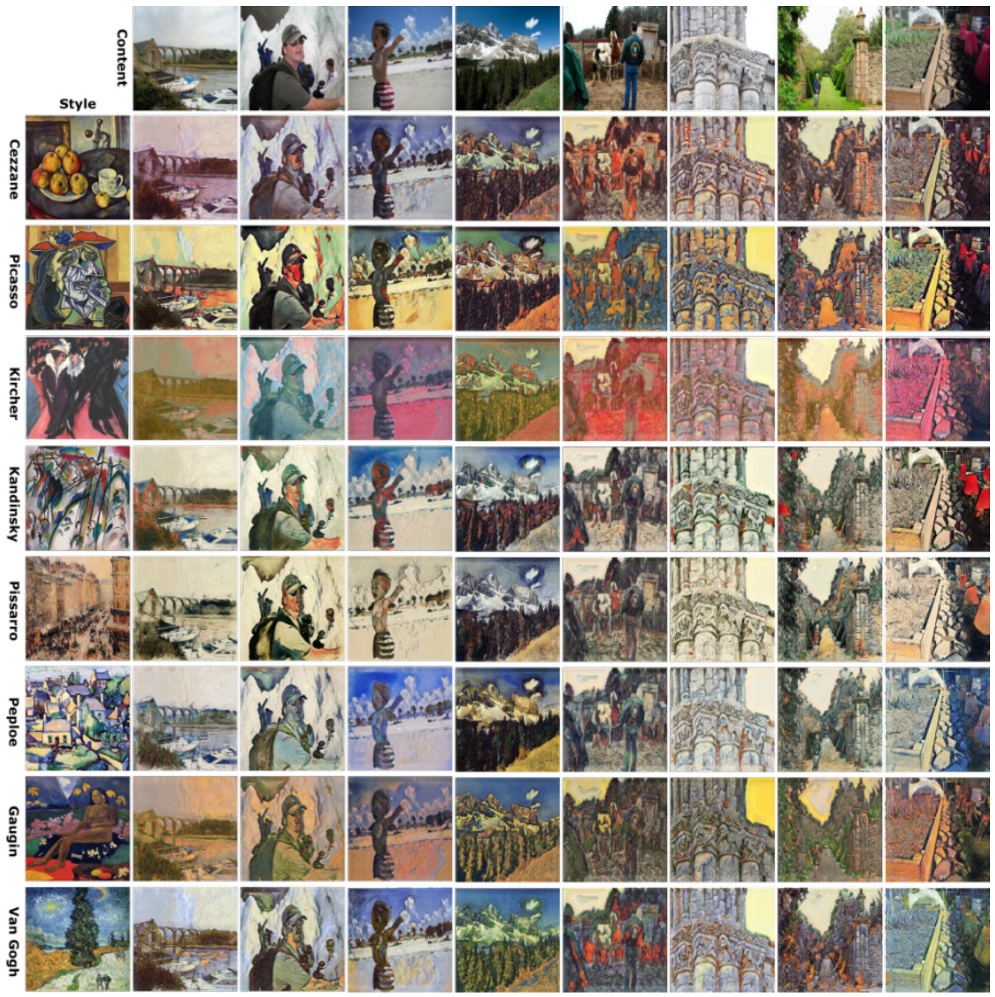


Figure 7: Qualitative evaluation of our method generalizing to different, even previously unseen styles.

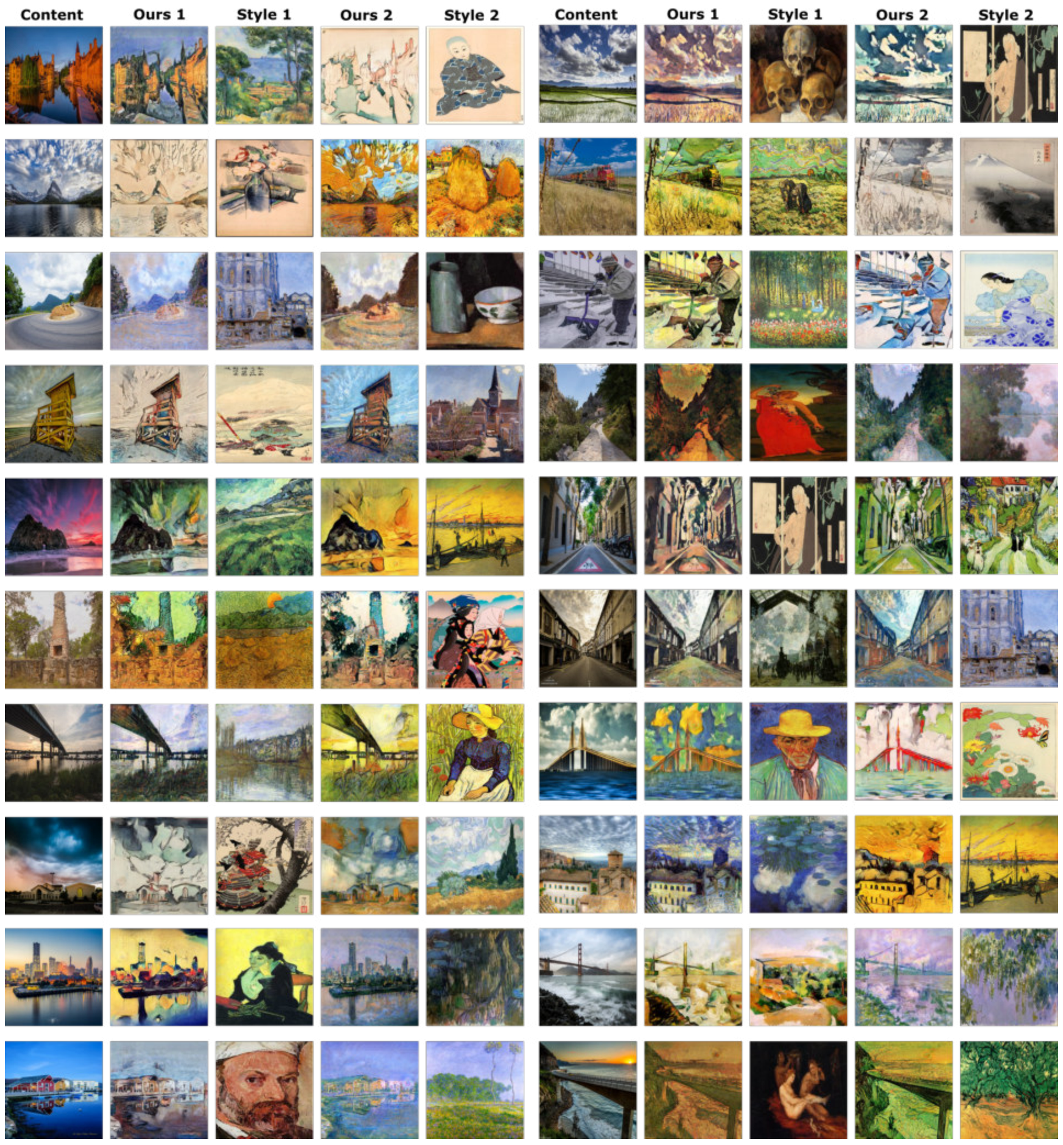


Figure 8: Qualitative evaluation of randomly coupled content and style images from the test set containing only painter styles seen during training.

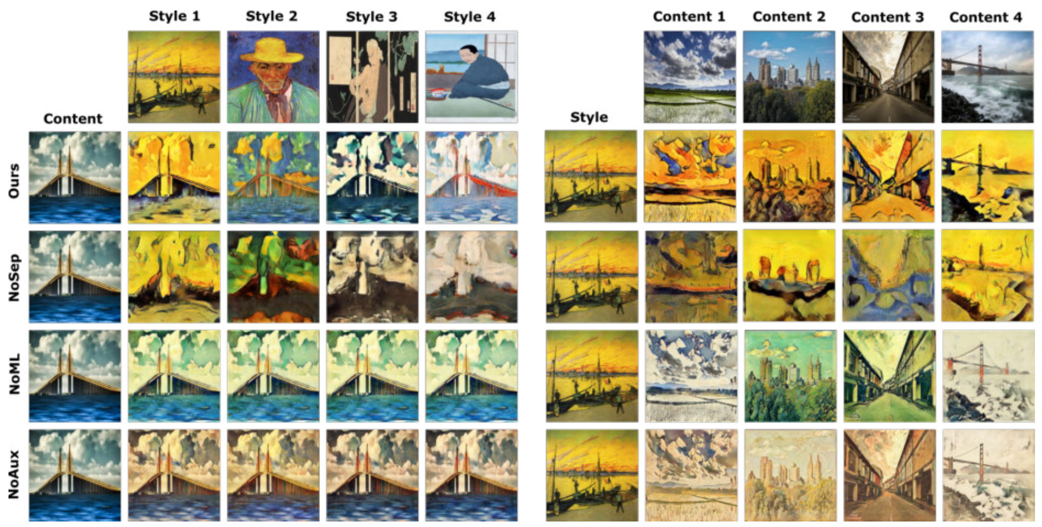
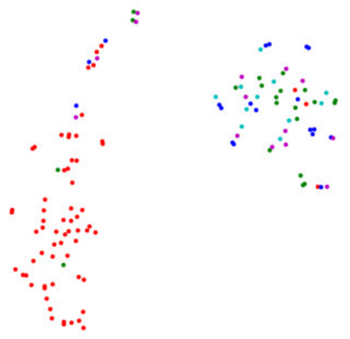
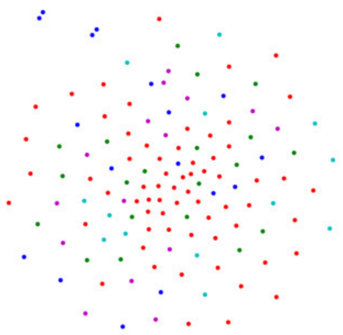


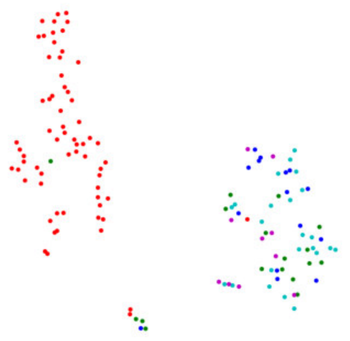
Figure 9: Ablation study evaluating different architecture choices for our approach. It shows how a fixed content reacts to different styles (left) and vice-versa (right). *Ours* refers to the final approach with all losses, *NoSep* ignores the separation of content and style in the latent space during feature exchange in PeRFeaT layer, *NoML* does not use metric learning and *NoAux* makes no use of the auxiliary decoder during training.



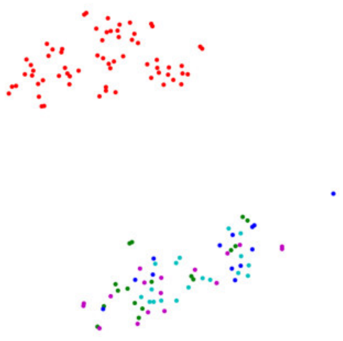
(a) Ours



(b) Without metric learning loss



(c) Without auxiliary decoder



(d) Transfering the whole latent code

Figure 10: T-SNE 2D embedding of the style latent space. The real photos are in red, different paints are represented by all the other colors. The latent space structure is shown for our final version and also for all 3 ablation study experiments we have performed.

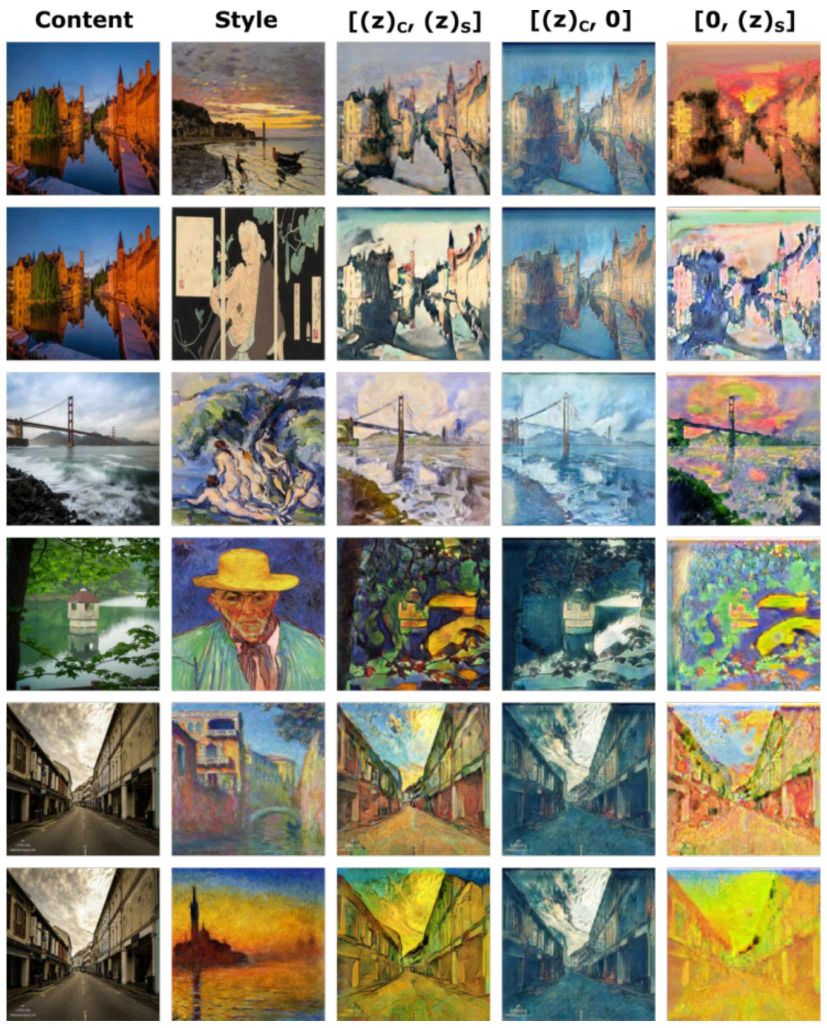


Figure 11: Visualization of information contained in content and style parts of the latent representation. Even if $(z)_C$ is set to 0, there is still some vague resemblance of the structure of the *Content* image, because the PeRFeaT layer transforms a partially local style features based on the content features.