1. Autoencoder Network

Most approaches in style transfer work on the features of an image taken from an encoder network. These features are reconstructed to the image space with a decoder network. Architectures $\Theta = (\phi, \phi^{-1})$ using an encoder ϕ and a decoder ϕ^{-1} that map their input to its identity via an intermediate representation are called autoencoder networks.

$$\Theta(\mathbf{x}) = (\phi^{-1} \circ \phi)(\mathbf{x}) = \mathbf{x}', \quad \mathbf{x} = \mathbf{x}' \tag{1.1}$$

This work presents a learned autoencoder network that works with images. At first, the architecture is shown. Further on, different losses are introduced that are used in training. The chapter concludes with experiments.

1.1. Architecture

Figure 1.1 shows the whole autoencoder architecture consisting of an encoder and a decoder network.

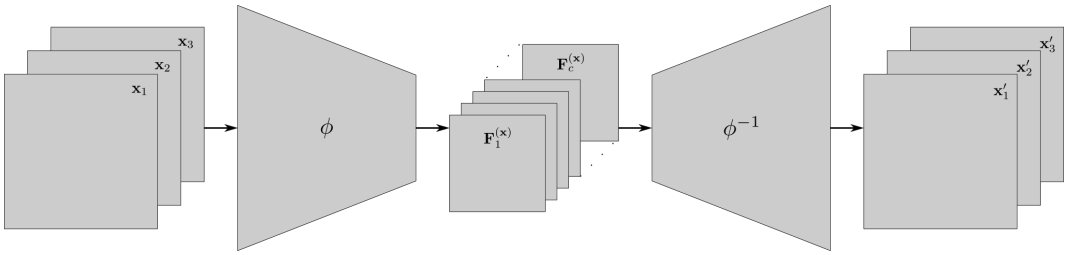


Figure 1.1.: Autoencoder network.

1.1.1. Encoder

An encoder network is an ANN that learns a representation for a set of data. The aim of encoding data is to reduce its dimensionality and to ignore noise in the input. If one wants to manipulate the given data, encoding can produce representations that make it easier to change the data in a specific way.

In this work, encoding images is done for two main reasons:

1. The reduction of dimensionality in the encoding procedure yields feature maps that are easier to manipulate than the original image.

2. The obtained feature maps represent different characteristics of an input image. An example could be a feature map that detects a certain kind of brushstrokes. Images with this kind of characteristic will produce a high activation in this feature map.

In this work, a pre-trained VGG-19 is used up to the layer `relu_4_1` as an encoder network. Figure 1.2 shows the encoder network broken down to its layers. Since only convolutional layers are used, input images can have arbitrary sizes.

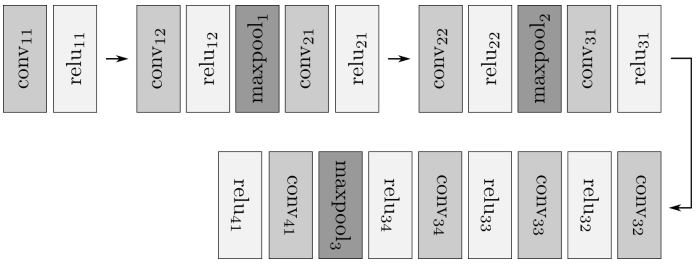


Figure 1.2.: Encoder network broken down to its layers.

An image $\mathbf{x}$ is encoded by passing it through the encoder network. The activations of each layer l in the forward-pass provide the respective feature space. Let the features at layer l of image $\mathbf{x}$ be referred to as matrix of feature maps.

$$ {}_l\mathbf{F}^{(\mathbf{x})} \in \mathbb{R}^{c_l \times (h_l \cdot w_l)} $$

The features consist of c_l feature maps, each with height h_l and width w_l. For ease of readability each feature map is described vectorized by a column vector, where

$$ {}_l\mathbf{F}^{(\mathbf{x})}_{ij} \in \mathbb{R}^{(h_l \cdot w_l)} $$

is the j-th entry of the i-th feature map at layer l. If the layer index l is not specified, the feature response at layer `relu_4_1` is meant.

Generally, each layer of the VGG-19 network defines a non-linear filter bank. Since the encoder network originally was trained on an image recognition task, the network developed several low-level filters and higher-level filters that represent the image in a meaningful way. By going deeper in the network, the complexity of the filters increases. While the first layers in the network focus on low-level features in the input images like colors and edges, deeper layer activations combine low-level features into basic grid-structures which gradually get combined into more complex structures like human faces [**?**].

Training the encoder network could destroy the learned filter banks and therefore manipulating them could not have the desired effect and results in lower image quality. To preserve the filter banks learned by the encoder network, it is fixed in the training procedure. Pre-trained VGG models of different deep learning frameworks perform very different in the autoencoder architecture. This is described further in Appendix A.1.

1.1.2. Decoder

The decoder ϕ^{-1} is trained from scratch. It reconstructs the image $\mathbf{x}$ given its feature response $\mathbf{F}^{(\mathbf{x})}$.

$$\phi^{-1}(\mathbf{F}^{(\mathbf{x})}) = \mathbf{x}', \quad \mathbf{x} = \mathbf{x}' \tag{1.2}$$

The image $\mathbf{x}'$ should be identical to $\mathbf{x}$.

The architecture of the image decoder network ϕ^{-1} mostly mirrors the architecture of the encoder network ϕ. There are no deconvolution layers used, because they lead to strong checkerboard artifacts [**?**]. Instead of using deconvolutions, upsampling and reflection padding is utilized. The upsampling layer applies two-dimensional nearest neighbor upsampling to the input tensor which also contracepts the production of checkerboard artifacts. Further on, reflection padding opposes checkerboard artifacts by padding the input tensor with one value in every dimension. The padding values are the reflection of the input boundary. Figure 1.3 shows the architecture described.

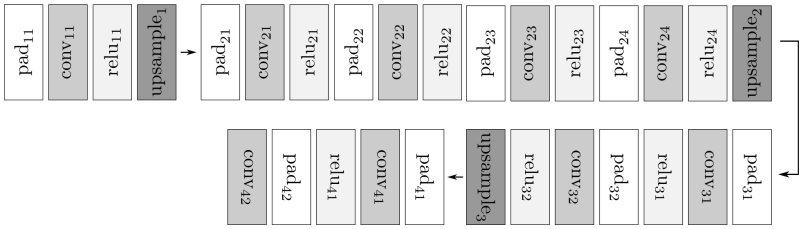


Figure 1.3.: Decoder network broken down to its layers.

1.2. Loss

Generally, one can think of two different losses which reconstruct the input image. Let $\mathbf{x} \in \mathbb{R}^{c' \times h' \times w'}$ be the input image and $\mathbf{x}' \in \mathbb{R}^{c' \times h' \times w'}$ the output image.

1.2.1. Per-Pixel Loss

The per-pixel loss is defined as mean squared error between the input image and the output image. Figure 1.4 shows a visualization of the loss.

$$\mathcal{L}_{\text{pp}}(\mathbf{x}, \mathbf{x}') = \frac{1}{c' \cdot h' \cdot w'} \cdot \sum_{i=1}^{c'} \sum_{j=1}^{h'} \sum_{k=1}^{w'} \left(\mathbf{x}_{ijk} - \mathbf{x}'_{ijk}\right)^2 \tag{1.3}$$

The loss forces the pixel values of $\mathbf{x}'$ to resemble the ones of $\mathbf{x}$.

1.2.2. Feature Loss

The feature loss measures the difference between the feature maps of the input image and the feature maps of the output image. It is defined as perceptual loss in the encoder network. Figure 1.4 shows a visualization of the loss.

$$\mathcal{L}_{\text{feat}}\left(\mathbf{F}_{ij}^{(\mathbf{x})}, \mathbf{F}_{ij}^{(\mathbf{x}')}\right) = \frac{1}{c \cdot h \cdot w} \cdot \sum_{i=1}^{c} \sum_{j=1}^{h \cdot w} \left(\mathbf{F}_{ij}^{(\mathbf{x})} - \mathbf{F}_{ij}^{(\mathbf{x}')}\right)^2 \tag{1.4}$$

The similarity between images in the feature space of a high performing CNN correlates well with the human perceptual similarity [**?**]. By matching the feature maps of the input image, the output image is urged to be perceptually similar to the input image.

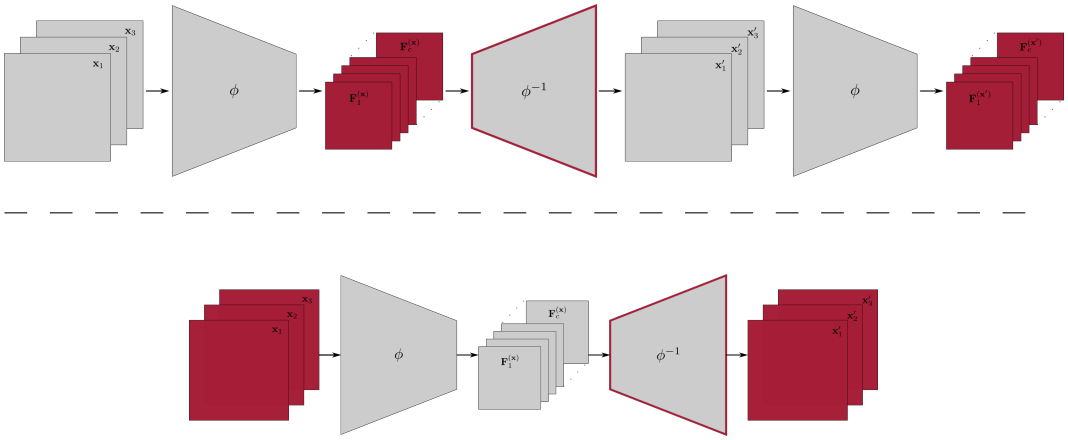


Figure 1.4.: Loss visualization of feature loss (above the dashed line) and per-pixel loss (below the dashed line) in the autoencoder network. Parts that are colored red represent where the loss is computed on. Bordered red parts represent which weights are changed by gradient descent.

1.2.3. Total Variational Regularizer

A Total Variational (TV) regularizer penalizes high frequency structures in the input image by minimizing the sum of derivatives of the image in h' and w' direction. The TV loss thereby reduces checkerboard artifacts. Also, it smoothens the image especially edges and regular structures.

$$\mathcal{L}_{\text{TV}}\left(\mathbf{x}'\right) = \frac{1}{c'} \cdot \sum_{i=1}^{c'} \left(\left| \frac{\partial \mathbf{x}_i'}{\partial h'} \right| + \left| \frac{\partial \mathbf{x}_i'}{\partial w'} \right| \right)^2 \tag{1.5}$$

The total loss is a weighted sum of the presented losses and can be written as follows.

$$\mathcal{L} = \lambda_{\text{pp}} \cdot \mathcal{L}_{\text{pp}} + \lambda_{\text{feat}} \cdot \mathcal{L}_{\text{feat}} + \lambda_{\text{TV}} \cdot \mathcal{L}_{\text{TV}} \tag{1.6}$$

1.3. Training

Since the decoder has to revert content images that are rendered in a certain style from the feature space to the image space, it is trained on a content image dataset as well as on a style image dataset.

Content images are taken from the MS-COCO dataset [**?**] and style images are taken from the Kaggle painter-by-numbers competition [**?**]. Each dataset contains about $80,000$ images.

The autoencoder is trained for 200 epochs. The Adam optimizer [**?**] is used with a learning rate of 10^{-4} and $\beta_1 = 0.9, \beta_2 = 0.999$.

1.4. Experiments

The best results are achieved by balancing the three loss terms as follows.

$$\lambda_{\text{pp}} = 25, \quad \lambda_{\text{feat}} = 1, \quad \lambda_{\text{TV}} = 10^{-6} \tag{1.7}$$

An ablation study exploring the balancing problem can be found in Appendix A.2. Figure 1.5 shows some content images and style images produced by the autoencoder and the ground truth.

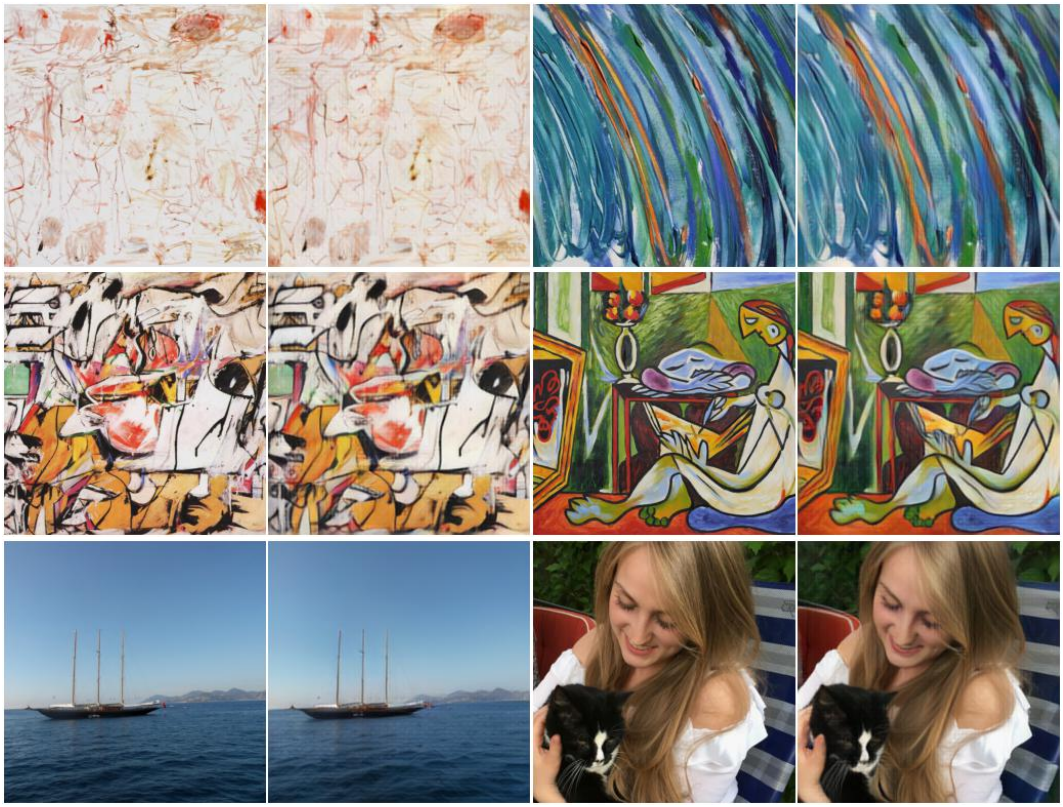


Figure 1.5.: Image examples produced by the autoencoder.

A. Autoencoder

This appendix gives further information on the autoencoder architecture presented in Chapter 1.

A.1. Pre-Trained VGG Networks

Section 1.1.1 explained why one wants to leave the encoder with its pre-trained weights fixed in the autoencoder network.

Nevertheless, it does make a difference how the VGG-19 was trained on the object recognition task. This work uses the pre-trained model from the Caffe framework [**?**] that has the best feature responses. The pre-trained VGG-19 model of the PyTorch framework [**?**] seems to be trained for less epochs and therefore does not yield the same quality of feature responses.

The quality of the feature responses directly correlates with the image quality that can be reached. Especially, the strength of checkerboard artifacts seems to be correlated negatively with the quality of the feature responses.

A.2. Loss Balancing

The main complexity in training the decoder network with a fixed encoder network is to balance the three presented losses in a way that prevents strong checkerboard artifacts from evolving.

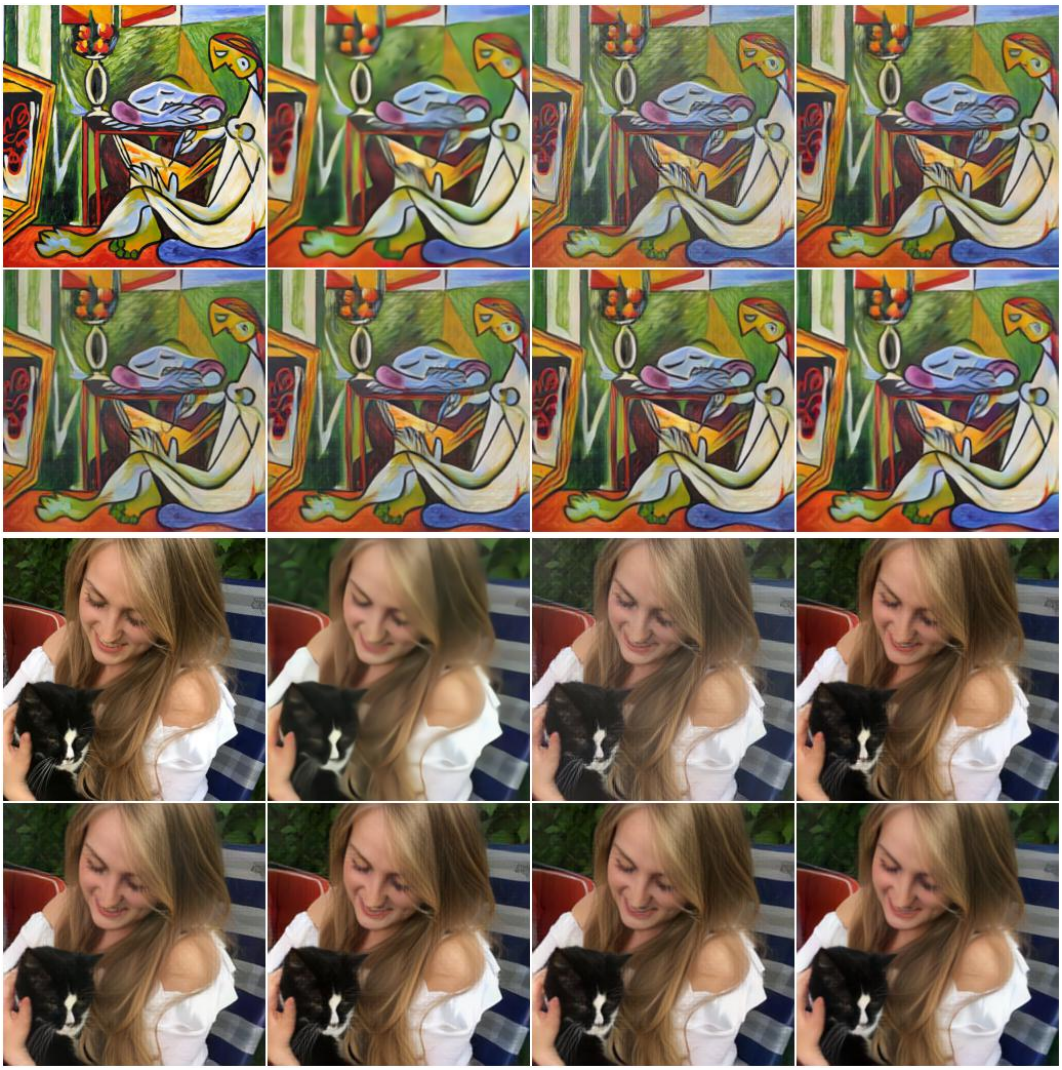
$$\mathcal{L} = \lambda_{\text{pp}} \cdot \mathcal{L}_{\text{pp}} + \lambda_{\text{feat}} \cdot \mathcal{L}_{\text{feat}} + \lambda_{\text{TV}} \cdot \mathcal{L}_{\text{TV}} \tag{A.1}$$

Using only a per-pixel loss with $\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 0, \lambda_{\text{TV}} = 0$ results in images that are equal to the input image in color and spatial structure. However, the images are in contrast to the ground truth images very blurry and do not reflect fine structures from the input images. This is due to the fact that no high-level features of the images are matched but only the pixel values. Using a L_1 loss instead of a L_2 loss does not bring any improvement.

If one combines the per-pixel loss with a feature-loss and the balancing factors $\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 1, \lambda_{\text{TV}} = 0$, almost every image has very strong checkerboard artifacts. This is attributed to not training the encoder. Matching high level features and

not training the encoder leads to checkerboard artifacts since the features may not represent the whole information needed to reconstruct an image.

Therefore, the key idea is to balance λ_{pp} and λ_{feat} in a way to get the best possible tradeoff between visually clear images and no checkerboard artifacts. Figure A.1 shows the balancing process for several factors. Since the images with $\lambda_{\text{pp}} = 25$ and $\lambda_{\text{feat}} = 1$ are perceptually most appealing, the autoencoder with those factors is used in this work.



ground truth	$\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 0$	$\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 1$	$\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 10$
$\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 15$	$\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 20$	$\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 25$	$\lambda_{\text{pp}} = 1, \lambda_{\text{feat}} = 50$

Figure A.1.: Visualization of different loss factors of λ_{pp} and λ_{feat}.

A.2.1. TV Regularizer

The TV regularizer can reduce checkerboard artifacts that are still visible in some images. Surprisingly, balancing the TV regularizer with a higher factor does not reduce checkerboard artifacts in a better way but instead makes the checkerboard artifacts better visible.

While a factor of $\lambda_{\mathrm{TV}} = 10^{-3}$ results in gray images and a factor of $\lambda_{\mathrm{TV}} = 10^{-4}$ shifts the colors, the best results are achieved by a factor of $\lambda_{\mathrm{TV}} = 10^{-6}$. Figure A.2 shows some images which make the effect of the TV regularizer visible.

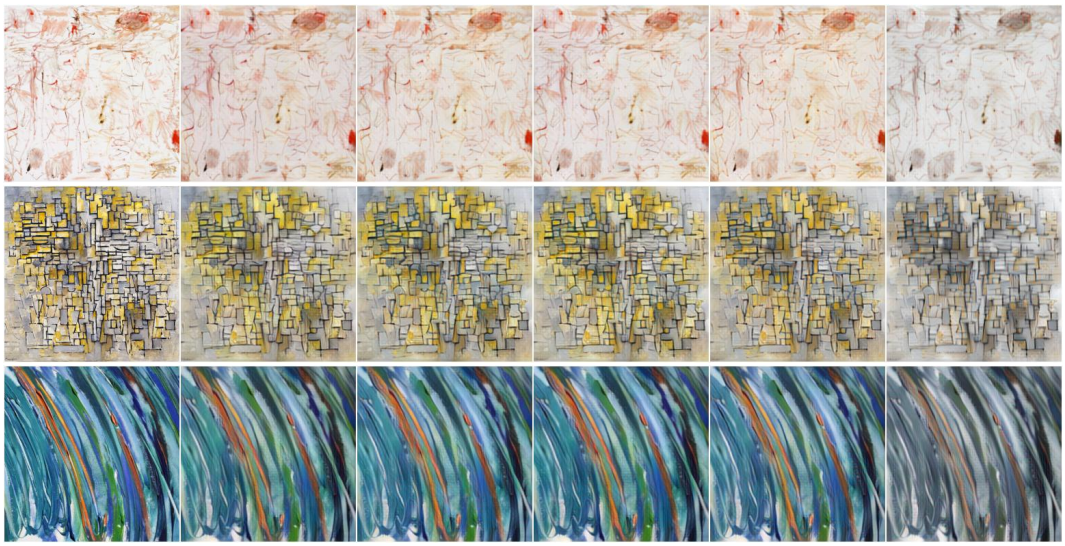


Figure A.2.: Visualization of different loss factors of λ_{TV}.