



UNIVERSITY
OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Artu Käyrä

IMAGE STYLE TRANSFER WITH NEURAL NETWORKS

Bachelor's Thesis
Degree Programme in Computer Science and Engineering
March 2020

Käyrä A. (2020) Image Style Transfer with Neural Networks. University of Oulu,
Degree Programme in Computer Science and Engineering, 26 p.

ABSTRACT

The main goal of this thesis is to introduce the reader to different style transfer methods that are based on artificial neural networks.

The thesis begins with an introduction to some key concepts regarding artificial neural networks. These concepts include a single artificial neuron, a feedforward neural network, convolutional neural network and generative adversarial networks.

Next, the main ideas behind five different style transfer methods are explained. Finally, style transferred images created by these methods are presented and some characteristics of different methods are demonstrated.

Keywords: deep learning, computer vision, image transformation

Käyrä A. (2020) Kuvien tyylimuunnos neuroverkoilla. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 26 s.

TIIVISTELMÄ

Tämän kandidaatintyon tarkoitus on esitellä lukijalle erilaisia keinotekoisiin neuroverkkoihin pohjautuvia kuvien tyylimuunnosmenetelmiä.

Aiheeseen johdatellaan käymällä ensin läpi päätärteittäin muutamia keinotekoisiin neuroverkkoihin liittyviä konsepteja, kuten yksittäinen keinotekoinen neuroni ja näistä neuroneista muodostettava neuroverkko, konvoluutioneuroverkko sekä generatiivinen kilpaileva verkosto.

Seuraavaksi viiden erilaisen tyylimuunnosmenetelmän toimintaperiaate käydään läpi. Lopuksi eri menetelmien tuottamia tyylimuunnettuja kuvia esitetään ja lisäksi joitain menetelmien erityispiirteitä demonstroidaan.

Avainsanat: syväoppiminen, konenäkö, kuvamuunnos

TABLE OF CONTENTS

ABSTRACT	
TIIVISTELMÄ	
TABLE OF CONTENTS	
FOREWORD	
LIST OF ABBREVIATIONS AND SYMBOLS	
1. INTRODUCTION.....	7
2. ARTIFICIAL NEURAL NETWORKS.....	8
2.1. Artificial Neuron.....	8
2.2. Artificial Neural Network	8
2.3. Convolutional Neural Networks	9
2.4. Generative Adversarial Networks.....	11
2.5. VGG-Network Architecture.....	12
3. METHODS	14
3.1. Optimisation Based Method	14
3.2. Optimisation via Feed Forward Network	15
3.3. Patch Based Style Transfer	16
3.4. Style Transfer via Feature Transforms	17
3.5. Style Transfer via Cycle-Consistent Adversarial Networks	18
4. EXPERIMENTS.....	19
5. SUMMARY	24
6. REFERENCES	25

FOREWORD

I want to thank D.Sc. (Tech) Janne Heikkilä for supervising the thesis and for giving valuable feedback and guidance. I also want to thank my fiancé, as well as my family and friends for supporting me throughout my studies. Last but not least, I want to thank Valossa Labs Oy for providing the hardware to run the experiments.

Oulu, March 11th, 2020

Artu Käyrä

LIST OF ABBREVIATIONS AND SYMBOLS

CNN	convolutional neural network
FC	fully connected layer
ReLU	rectified linear unit
WCT	whitening and coloring transform
GAN	generative adversarial network

1. INTRODUCTION

Style transfer is a type of image transformation where a stylised image is generated by applying some style characteristics to a content image. It is easy to imagine that an artist could paint two different landscape paintings with a similar style. But how would one quantify an arbitrary style and apply it to an image programmatically?

Recent style transfer methods leverage deep neural networks that are trained for image classification. These methods make it possible to transform photographs into unique pieces of artwork in minutes or seconds, depending on the technique and used hardware. In addition to speed, the given technique also dictates if it generalises on multiple styles and how visually appealing results it provides.

The next chapter serves as an introduction to artificial neural networks. Most importantly, the objective of the following sections is to introduce the reader to convolutional neural networks (CNNs) that are extensively used by different style transfer methods. Additionally, other relevant concepts are explained. With this foundation, the methods described in Chapter 3 should be more accessible to readers without an extensive background in artificial neural networks. In Chapter 3, five different neural style transfer methods are presented and existing implementations of these methods are tested and results are shown in Chapter 4. In addition to the evaluation of visual quality, some of the characteristics of different methods are demonstrated.

2. ARTIFICIAL NEURAL NETWORKS

2.1. Artificial Neuron

An artificial neuron is a mathematical function inspired by biological neurons, hence the name neuron. The idea of an artificial neuron was first introduced in 1943 by McCulloch & Pitts [1]. Later, in 1957, perceptron was invented by Rosenblatt [2]. Artificial neurons used today are generalisations of these perceptrons.

One or multiple input nodes are connected to the neuron. These connections are weighted meaning that some input nodes have a higher impact on the activity of the neuron. Next, the weighted sum of the input nodes' values are calculated and the bias term is added to this sum. This value is then passed into an activation function¹ which determines the output value of the neuron. An artificial neuron is depicted in Figure 1.

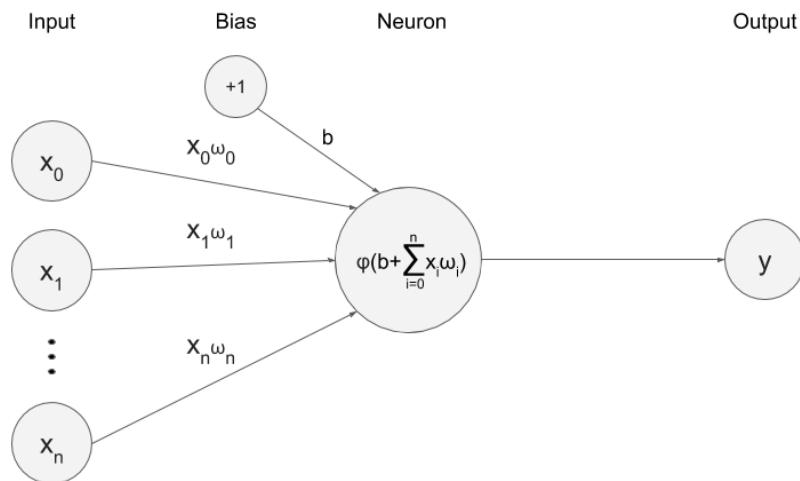


Figure 1. Visual representation of an artificial neuron.

An artificial neuron is a simple unit that does not have the capacity to learn complex mappings between the inputs and outputs. This shortcoming can be overcome by using multiple neurons to form a network. The increasing number of learnable parameters enables the network to learn more complicated tasks.

2.2. Artificial Neural Network

An artificial (feedforward) neural network consists of an input layer, one or multiple hidden layers and an output layer, forming an acyclic graph. Mathematically, it can

¹Activation functions are used to restrict the activation of a neuron to a finite value. Non-linear activation functions, such as ReLU, sigmoid and many others, are used in neural networks to allow the network to learn non-linear functions.

be described as functions that are chained together: $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. In this case, $f^{(1)}(x)$ takes in the input vector and produces an output vector that is then used as input in the next function [3].

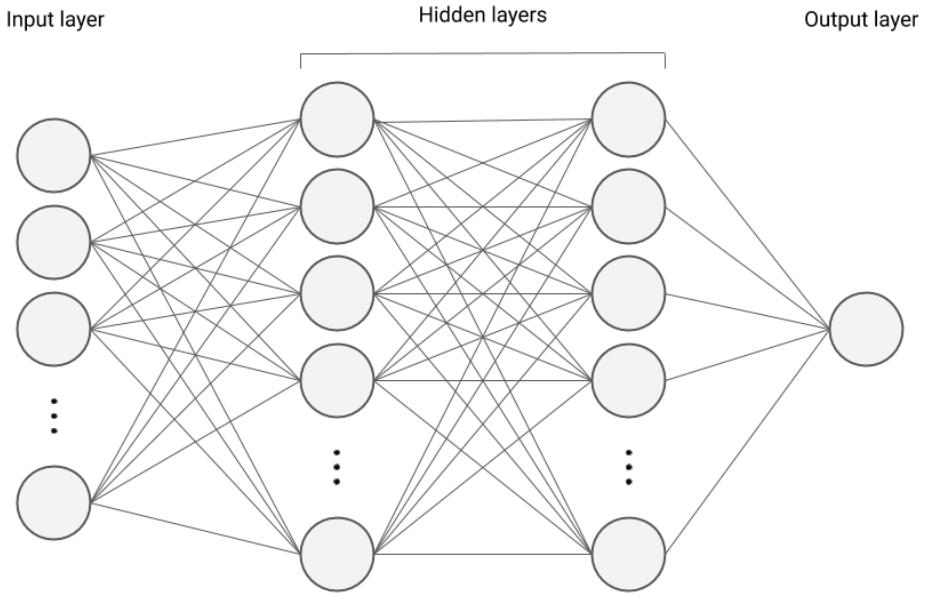


Figure 2. Visual representation of a feedforward neural network with an input layer, two hidden layers and a single output. A network with a single output could be used for regression and binary classification. Most of the notation is omitted to keep the figure readable. Note that in many cases each hidden layer neuron is also connected to a bias neuron.

Single neurons in hidden layers and output layer work in the same manner as described in the previous section. In feedforward networks, inputs of a neuron always come from the previous layer and output is passed to the next layer. The number of trainable parameters, weights and biases, increase as the number of neurons in the network increase. This increased complexity allows the network to learn more challenging tasks. However, if the network is too complex for the dataset that is being used for training, overfitting is likely to occur. The term overfitting means that a model that has learned the sampling noise of the training dataset and does not generalise well on unseen test data [4]. Increased complexity also increases the memory requirements and processing power needed for training and inference. The learning itself is the process of adjusting the weights and biases with a technique called backpropagation [5]; however, details of this process are outside of the scope of this thesis.

2.3. Convolutional Neural Networks

CNNs have been used for character recognition tasks already in nineties [6] but the recent popularity of CNNs is due to work [7] that describes a deep CNN that managed to beat the state-of-the-art methods in the ImageNet image classification challenge in

2012. Although CNNs can be used with different types of data, such as time-series data, this chapter details CNNs from the perspective of image data.

CNNs are neural networks that contain at least one convolutional layer. These convolutional layers typically consist of three stages: convolutional stage, activation stage and pooling stage [3]. At the first stage, the convolution operation is performed between an input image or a feature map and a filter resulting in a new feature map. In a typical CNN trained for object detection, the first convolutional layer following the input layer receives an image as an input and the following layers get the resulting feature maps from the previous layers. Each convolutional layer generally has multiple filters, also called kernels, that are small $N \times N$ matrices that can have multiple channels. The filters contain the weights that have been learned during the training of the network. An intuition to filters is that filters in the earlier layers detect, for example, edges. In the deeper layers, filters learn to identify shapes and objects. During convolution, these filters are slid over the input matrix. At each position, element-wise matrix multiplication is performed and the sum of the values in the output matrix is placed into the feature map. An example of 2D convolution is depicted in Figure 3. After the convolution stage, the resulting feature maps are passed into an activation function and at the last stage, pooling is applied onto the resulting feature maps. The objective of pooling is to reduce the size of feature maps by summarizing sub-regions of feature maps by taking the maximum or average value in each sub-region [8].

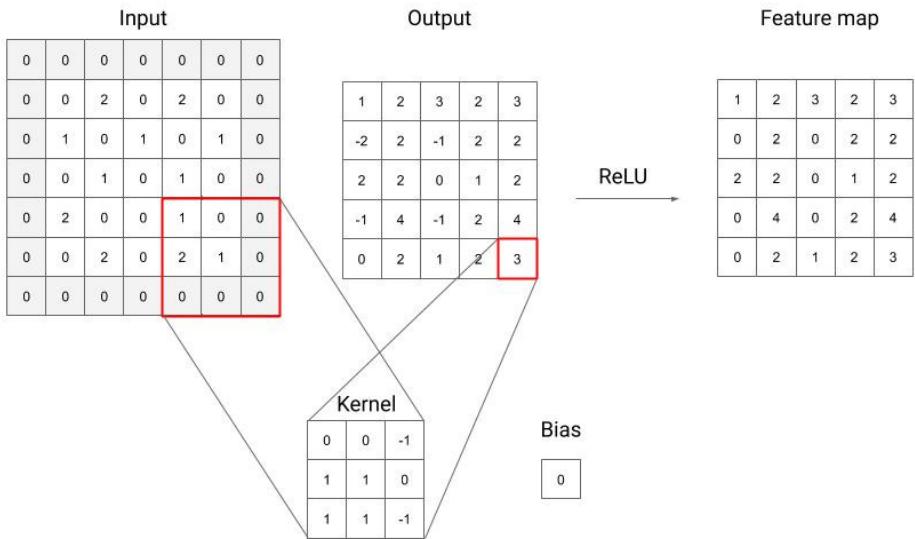


Figure 3. An example of convolution between zero-padded 5x5 input and 3x3 kernel with a stride of one.

CNNs are effective when the data can be represented in a grid-like topology [3]. This grid-like topology presumes that the values in the grid have spatial or temporal relationships. Images are naturally represented as a grid of pixel values and pixels that are in close proximity to one another can form an object and therefore have a meaningful correlation. Images can and have been used as inputs for traditional feedforward networks as well. However, as discussed earlier, feedforward networks

use vector inputs. This means that the image has to be flattened and, as a result, the spatial and temporal relationships between values are lost [9]. Given the natural grid-like topology of images, there are other advantages that CNNs possess over traditional neural networks. One of these is called sparse connectivity. In essence, it means that one neuron is connected only to a few other neurons in the previous layer. This idea is also called shared weights, which stems from the fact that the weights of each filter are used in every position of the input. In contrast, traditional neural networks that have fully connected layers, each neuron is connected to every neuron in the previous layer and thus require more weights [3]. Although the connections are sparse, neurons deeper in the network can be connected to most of the input neurons. This effect is increased if there are pooling layers or strided convolutions in the network [3]. This idea is depicted in Figure 4. Additionally, sparse connectivity, local receptive fields and pooling make CNNs, in some degree, shift and distortion invariant² [9].

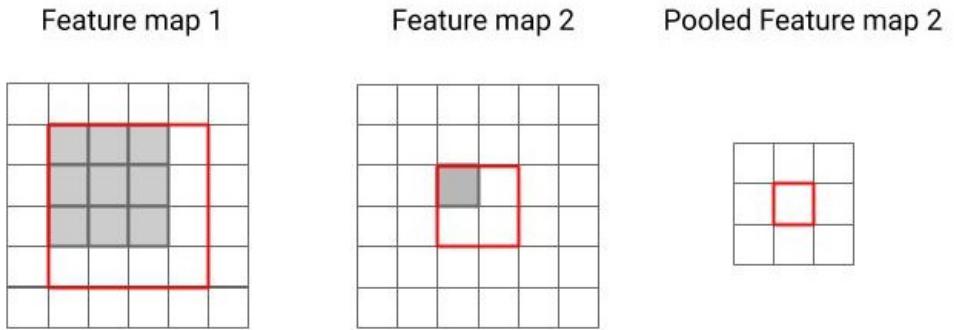


Figure 4. Neurons deeper in the network have larger receptive fields. Downsampling increases the size of the receptive field as well. In this example, a neuron in pooled feature map 2 is connected to 4 neurons in feature map 2 and 16 neurons in feature map 1.

2.4. Generative Adversarial Networks

Generative adversarial networks (GANs) refer to a setting where a generative model is set in opposition to a discriminative model, where the goal is to improve the performance of both models through an adversarial process [10]. As a simple example, the dataset could be a set of real paintings and the generator creates new, fake, paintings. Next, the discriminator is provided with real paintings and fake paintings. Then it outputs a probability of a painting being real rather than fake. The discriminator gets feedback from the output it provided and learns to better classify both real and fake paintings into their corresponding classes. The generator, on the other hand, learns

²Shift and distortion invariance means that filters will pick up the features, for example, of an object regardless of its position in an image and also if there is some slight distortion that makes the object less recognisable.

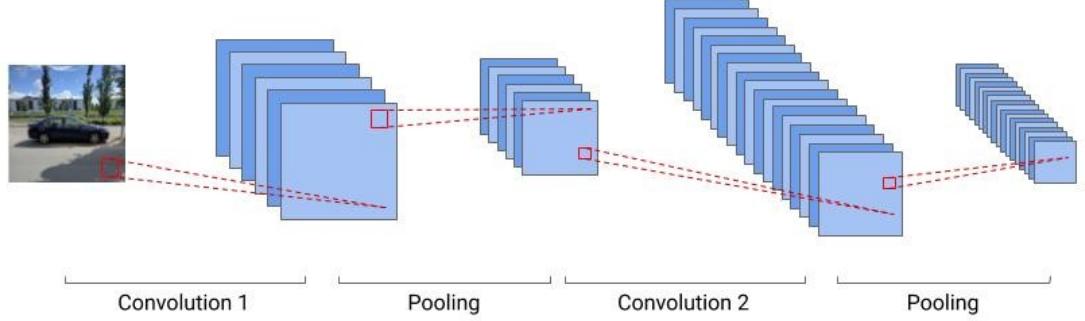


Figure 5. Visual representation of feature extraction.

only from the feedback regarding the paintings it created and learns to generate more convincing fake paintings.

In more formal terms, GANs participate in a minimax game, where one agent tries to maximise the value function and the other tries to minimise it [10]. Specifically, the generator tries to maximise the probability of the discriminator making misclassifications and, in contrast, the discriminator tries to maximise the probability of making correct classifications. This idea is formulated into an adversarial loss function that is used during training. In an optimal scenario, where both models improve at the same pace and have enough capacity, the distribution of generated samples matches the distribution of the used dataset. As a result, the discriminator cannot distinguish between samples from the model distribution and data distribution and learning comes to a halt.

2.5. VGG-Network Architecture

Since all but one of the following style transfer methods use VGG-Network, the architecture of two variants, VGG-16 and VGG-19, are is shown in Table 1.

The network is divided into two parts, feature extraction and classification. Feature extraction begins from the input layer and extends to the last maxpool layer. The fully connected (FC) layers and the softmax layer handle the classification. However, the classification part of the network is not used in the following style transfer methods.

Because the names of the convolutional layers in the table do not indicate where they are located in the network, different papers refer to specific convolutional layers slightly differently. In [11] the naming convention is the following: conv{block number}_{position in block}. For example, the convolutional layer after the first max-pooling operation is referred to as 'conv2_1' and the next one as 'conv2_2'. In [12] these layers are called 'ReLU2_1' and 'ReLU2_2' respectively.

Table 1. Architectures of VGG-16 and VGG-19 Networks. The parameters of each convolutional layer are receptive field size and number of channels, respectively. ReLU non-linearity is not present in the table but is applied after each convolution and in the fully connected (FC) layers.

VGG-16	VGG-19
input (224 x 224 RGB image)	
conv3-64	conv3-64
conv3-64	conv3-64
maxpool	
conv3-128	conv3-128
conv3-128	conv3-128
maxpool	
conv3-256	conv3-256
conv3-256	conv3-256
conv3-256	conv3-256
maxpool	
conv3-512	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
maxpool	
conv3-512	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
maxpool	
FC-4096	
FC-4096	
FC-1000	
soft-max	

3. METHODS

In this chapter implementations of different neural style transfer methods are introduced. The goal is to illustrate the idea behind every implementation and indicate differences between methods. Following methods are covered:

1. Optimisation based method
2. Optimisation via feed forward network
3. Style swap via inverse network
4. Style transfer via feature transforms
5. Style transfer via cycle-consistent adversarial networks

3.1. Optimisation Based Method

This method described in [11] is arguably the first to apply deep neural networks in style transfer. The main finding of this paper is that the style and content representations of an image are separable in a convolutional neural network that is trained for image classification.

The implementation in the paper uses layers of VGG network [13] to separate style and content representations of the input images. Here, the content representation is the filter response of layer 'conv4_2' for the content image. A higher-level layer is used for the content representation because higher layers capture the high-level content of the image. This content representation is used in the content loss function that is the mean square error between the feature map of the content image and stylised image.

The style representation of the style image is expressed as a Gram matrix for each chosen layer. This Gram matrix is formed by flattening all feature maps on the layer and calculating inner dot products of each pair of vectorised feature maps. These Gram matrices are then used in the style loss function. This loss function is the weighted sum of the mean square errors between Gram matrices of the style image and Gram matrices of the styled image. The global loss function to be minimised is the weighted sum of the content and style loss. By changing the ratio of these weights the intensity of the style and prevalence of content can be altered.

The style transfer process begins with a randomly initialised image that is fed into the network. Next, the necessary activations for this image are taken and the content and style representations are calculated. Then the global loss is calculated and backpropagated which leads to changes in the randomly initialised image so that it starts to resemble the style and the content of the input images more closely after each iteration. This process continues for a certain number of iterations or until a local minimum of the loss function is found. Often this optimisation converges to a satisfactory result within 500 iterations [14].

Characteristics of this method are that it can be used with arbitrary styles without retraining and it provides visually appealing results. However, style transfer with this method is inefficient since it requires multiple forward and backward passes on the network.

3.2. Optimisation via Feed Forward Network

The greatest disadvantage of the previous method is the inefficiency of the style transfer. Method 2 discussed in this section is rather similar to the previous method but it manages to perform the style transfer three orders of magnitude faster while still retaining acceptable visual quality [14]. However, this comes at a cost: the style transfer networks used do not generalise on multiple styles, meaning each style transfer network can learn only one style.

The exact architecture of this style transfer network is not important in this context but it can be found on the author's website [15] and it is also mostly detailed in [14]. In essence, the style transfer network is a CNN that takes in an image and outputs a transformed image. This kind of networks are called image transform networks, and in this particular case, the transformation is the style transfer the network has learned to perform.

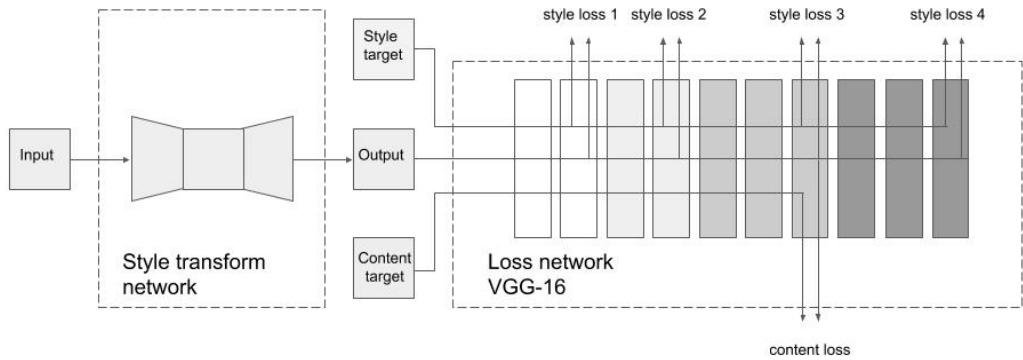


Figure 6. Visualisation on how the loss is calculated while training the image transform network.

The previous method used VGG-19 network to guide the optimisation process. This method uses the VGG-16 network but as a loss network during training. Commonly, when training a neural network, a loss function is used to calculate the error between networks prediction and ground truth. However here the loss is calculated using another network.

The training procedure begins with randomly initialised weights and thus the style transfer network makes some random transfer to the input image. The resulting image is then fed into the VGG-16 network and the global loss function is calculated similarly as discussed in the previous section, where the global loss function is the weighted sum of content loss and style loss functions. This loss value is then backpropagated to the style transfer network and weights are adjusted. This process is depicted in Figure 6. The weights of the VGG-16 network remain unchanged during the training. After a

successful training phase, the style transfer network has learned to approximate the result of the optimisation problem discussed in the previous section.

Another paper [16] proposed a minor change to the architecture of the style transfer network that increases its performance and improves the visual appeal of the stylised images. Although the architecture of this network was not detailed in this section, the only change was to use instance normalisation instead of batch normalisation in the convolutional layers. This alteration of the network architecture improves the perceptual quality of the style transfer, as is demonstrated in Chapter 4.

3.3. Patch Based Style Transfer

Method 3, introduced in [12], offers transfer of arbitrary style via optimisation as well as a feedforward approximation. What makes it different from the methods discussed previously is a different optimisation objective. The previous methods defined content and style losses separately and there were multiple style losses defined on different layers of the network. This method uses only one layer of the pre-trained VGG-19 network where style and content structure are combined using local matching of patches.

The authors call this matching procedure style swap and it goes as follows. First style and content patches are extracted from the input images using a truncated VGG-19 network. Next, the closest-matching style patch is determined for each content patch by using normalised cross-correlation measure. Then the content patches are replaced with their closest-matching style patches and finally, overlapping areas of the reconstructed content activations are averaged. The resulting feature maps represent the information of an image that has the structure and style of the corresponding input images.

Optimisation and feedforward approximation can be used to transform the feature representation into an image. The optimisation and image creation process is mostly the same as in Method 1. The only meaningful difference is the different optimisation objective. Method 1 tries to match the corresponding style and content representations on multiple layers while this method tries to match the combined style and content representation created on a single layer. According to the numbers presented in the paper, their optimisation procedure is faster per iteration and requires fewer iterations to reach a local optimum when compared to Method 1.

The feedforward approximation works rather differently compared to Method 2. As a reminder, Method 2 used an image transformation network that has learned to apply a style onto the input image and outputs the stylised image. In contrast, this method employs an inverse network to decode the resulting feature maps of the style swap back into an image. The inverse network has been trained to decode any feature map created by the truncated VGG-19 network into an image. Because of this and the fact that the style and content structure are represented in the style swapped feature maps, the inverse network generalises on unseen styles. According to the paper, this method is not as fast as Method 2 but its greatest advantage is the ability to perform style transfer on any pair of style and content images without retraining.

3.4. Style Transfer via Feature Transforms

Method 4 [17] uses a similar technique as the feedforward approximation in Method 3. While Method 3 encodes and decodes the image once and performs style swap in between, Method 4 uses different operation between the encoding and decoding steps and this process is carried out on different encoder-decoder pairs consecutively. This processing pipeline is depicted in Figure 7. The style transfer is achieved through feature transforms between encoding and decoding steps. The feature transform used is called whitening and colouring transform (WTC) and it matches the content feature statistics of the content image to those of the style image in the feature space. First, the whitening operation is performed on the content image. It is demonstrated in the paper that when these whitened features are inverted back to RGB space, the resulting image has preserved the global structure of the content but stylistic features have diminished. Next, the colouring operation is performed on the whitened feature maps. The resulting feature maps can now be decoded back into an image that has the structural content of the content image and stylistic features of the style image.

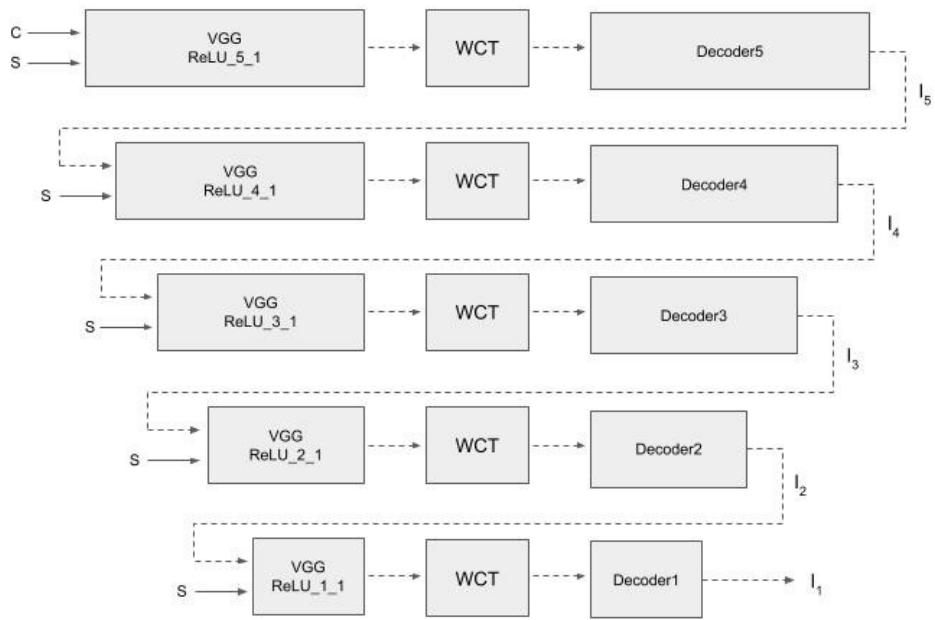


Figure 7. Style transfer pipeline used in Method 4. C and S depict the input content and style images. $I_N(N = 5, 4, 3, 2)$ depicts the intermediate style transferred image after $DecoderN$. These intermediate style transferred images are used as content images in the following encoder. I_1 is the finalised style transferred image.

3.5. Style Transfer via Cycle-Consistent Adversarial Networks

Method 5 [18] employs cycle-consistent GANs to transform photographs from one domain to another. The paper showcases image transformations in various domains, such as horses to zebras, summer to winter and landscape photograph to painting. As this thesis focuses on artistic style transfer, photograph to painting transformations are presented in Chapter 4. The general nature of this style transfer is indeed intriguing. But what makes it even more interesting is that this method not only learns the mapping from the source domain to the target domain but also the mapping from the target domain back to the source domain. This means that, with a suitable model, paintings can be transformed into realistic-looking photographs.

The architecture of this method consists of two generators and two discriminators. Let us call these generators G and F , and their respective discriminators D_Y and D_X . Two datasets, X and Y , are needed to train the models. The goal of the training is to train generator G to transform samples from X domain to be indistinguishable from domain Y . In contrast, generator F is trained to perform the transform from domain Y to domain X . The full objective includes two adversarial losses, one for each generator-discriminator pair, and a two-way cycle-consistency loss. The cycle-consistency loss is the distance between a sample and the same sample that has been translated to the other domain and back. This is done in both ways i.e $x \approx F(G(x))$ and $y \approx G(F(y))$ making it a two-way cycle-consistency loss. The intuition is that by chaining these two mapping functions the output would be approximately the same as the input. Ultimately, the role of the cycle-consistency loss is to reduce the space of possible mapping functions and to ensure that generators do not contradict each other. According to the paper, both the adversarial loss and cycle-consistency loss are critical for successful training.

In contrast to the previously presented methods, this method learns to translate the style characteristics of a collection, instead of a style image, to a content image. This makes the direct visual comparison between images generated by this method and the other methods more complicated.

4. EXPERIMENTS

In this chapter experimental results of existing implementations are presented and evaluated. In addition, some characteristics of each method are discussed later in this chapter.

Style transfers in Figure 8 were performed on four different content images and with four different style images. These content and style images are presented in Figure 9 and Figure 10 respectively. Each style transfer in Figure 8 was performed with the default settings of each implementation³. Style transfers with Method 2 were performed with models that used batch normalisation except for the image on the first row where a model with instance normalisation was used.

Method 1 seems to consistently deliver a transferred image that captures the original style quite well. The images were generated on 500 iterations. The iterative nature of this method is demonstrated in Figure 11.

The image on the first row second column looks very similar to the one on its left side. It shows that Method 2 can perform similar looking style transfer with just one forward pass on the network. From row two to four, however, the images look like some pattern was placed on top of the original image that makes everything blend into the background. This difference seems to be caused by the usage of batch normalisation instead instance normalisation in the convolutional layers of the style transfer network. Further comparison of batch normalisation and instance normalisation can be found in Figure 12.

Method 3 produces visually appealing images that have a very subtle stylistic effect. This subtle effect was achieved with the default settings of patch size of 3 and one style swap. Images created with different combinations of these settings are presented in Figure 13. In addition, results with different number of iterations are presented in Figure 14.

On default settings, Method 4 seems to generate images with most aggressive stylistic effect to a degree that objects in the image lose structure.

Style transfers performed by Method 5 are presented in Figure 15 and on the second column in Figure 16. The latter figure also demonstrates the cycle-consistency property of the transfer.

³ Method 1 and 2: <https://github.com/jcjohnson/neural-style>
 Method 3: <https://github.com/rtqichen/style-swap>
 Method 4: <https://github.com/Yijunmaverick/UniversalStyleTransfer>
 Method 5: <https://github.com/junyanz/CycleGAN>

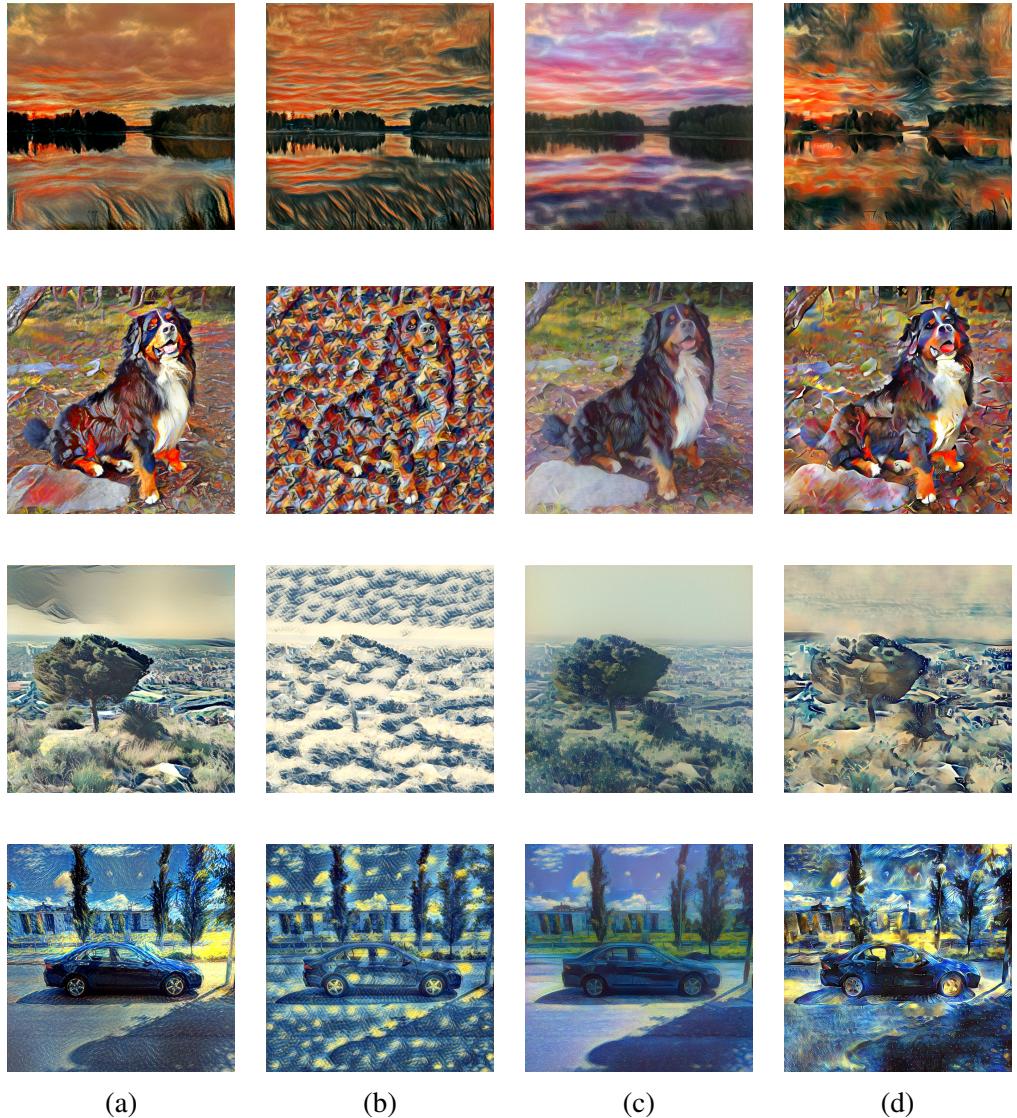


Figure 8. Results of the style transfers with Methods 1-4. Each column represents the results of one style transfer method: (a) Method 1 (b) Method 2 (c) Method 3 (d) Method 4.

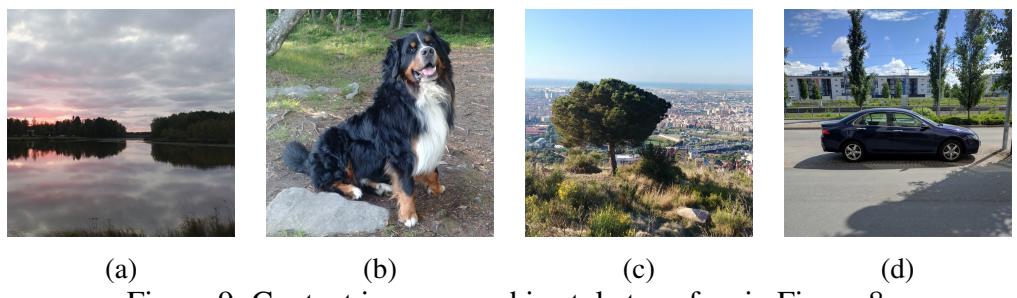


Figure 9. Content images used in style transfers in Figure 8.



Figure 10. Style images used in style transfer in Figure 8: (a) *The Scream* by Edvard Munch, 1893, (b) *Composition VII* by Wassily Kandinsky, 1913, (c) *The Great Wave off Kanagawa* by Katsushika Hokusai, 1829-1833, (d) *The Starry Night* by Vincent van Gogh, 1889.

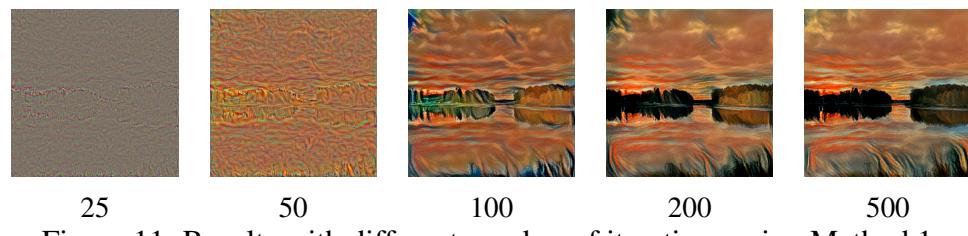


Figure 11. Results with different number of iterations using Method 1.

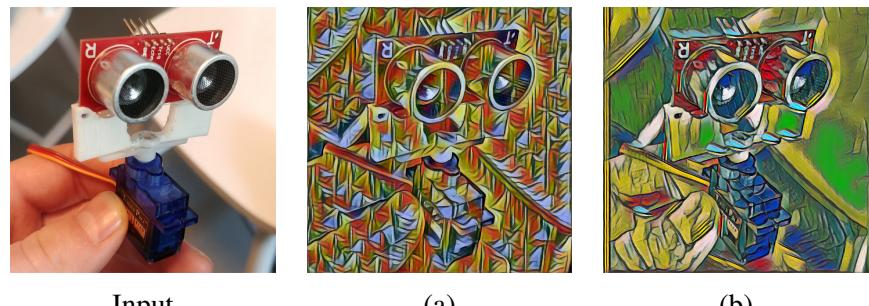


Figure 12. There is a distinct difference between results that have been created with a model that uses batch normalisation (a) compared to a model that uses instance normalisation (b) using Method 2.

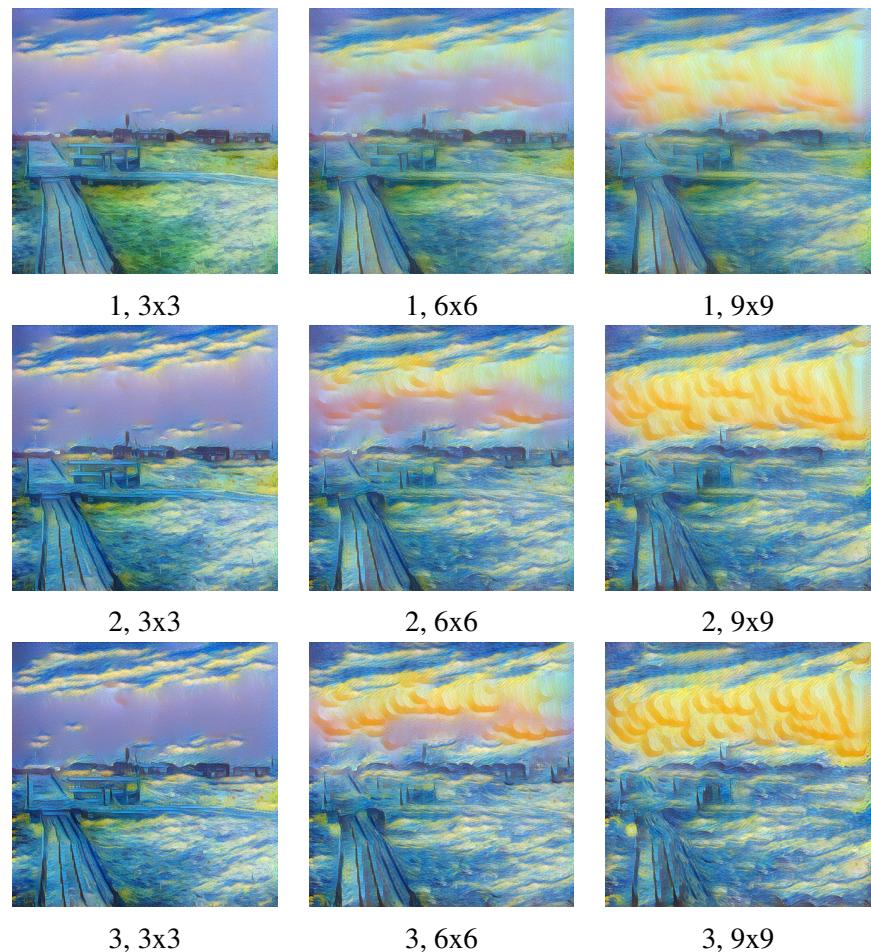


Figure 13. Results of the style transfers with Method 3 using different patch sizes and number of style swaps. The first number below an image indicates the number of style swaps and the second set of numbers is the patch size.

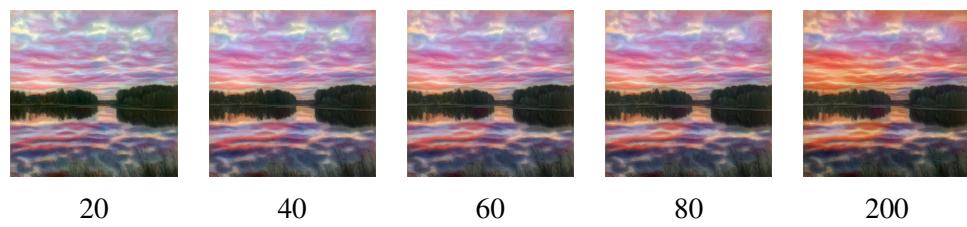


Figure 14. Results with different number of iterations using Method 4. It produces visually appealing results with a lower number of iterations compared to Method 1. Although the stylistic effect is more subtle.



Figure 15. Style transfers with Method 5 using model trained on a collection of paintings by Vincent van Gogh.

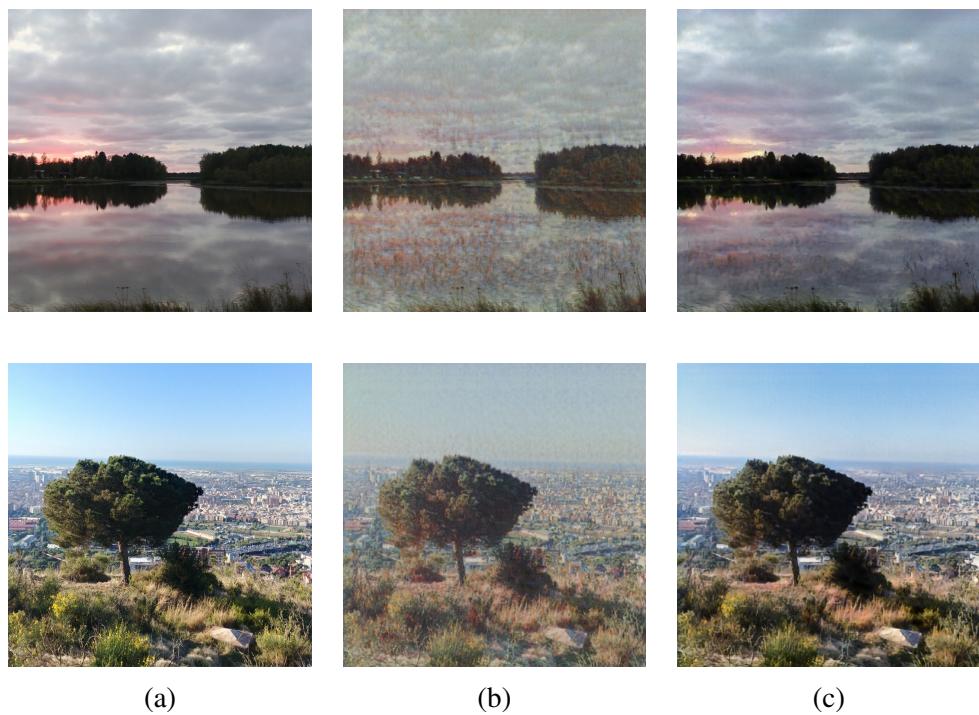


Figure 16. Demonstration of cycle-consistency with Method 5 using model trained on a collection of paintings by Claude Monet. (a) original photograph (b) translation from photograph to painting (c) translation from generated painting back to photograph.

5. SUMMARY

This thesis aimed to introduce the reader to style transfer via artificial neural networks. In addition, the earlier part of this thesis served as an introduction to some central concepts of artificial neural networks.

It was shown that style transfer can be performed by leveraging different neural network architectures and techniques.

In the latter part of the thesis style transferred images generated by the presented methods were shown. In addition, some of the characteristics of the methods were demonstrated.

6. REFERENCES

- [1] McCulloch W.S. & Pitts W. (1943) A logical calculus of the idea immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, pp. 115–133.
- [2] Rosenblatt F. (1957) The Perceptron, a Perceiving and Recognizing Automaton Project Para. Report: Cornell Aeronautical Laboratory, Cornell Aeronautical Laboratory. URL: https://books.google.fi/books?id=P_XGPgAACAAJ.
- [3] Goodfellow I., Bengio Y. & Courville A. (2016) Deep Learning. MIT Press. <http://www.deeplearningbook.org>.
- [4] Srivastava N., Hinton G., Krizhevsky A., Sutskever I. & Salakhutdinov R. (2014) Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, pp. 1929–1958. URL: <http://dl.acm.org/citation.cfm?id=2627435.2670313>.
- [5] Cun Y.L. (1988), A theoretical framework for back-propagation. URL: <http://yann.lecun.com/exdb/publis/pdf/lecun-88.pdf>.
- [6] LeCun Y., Boser B.E., Denker J.S., Henderson D., Howard R.E., Hubbard W.E. & Jackel L.D. (1990) Handwritten digit recognition with a back-propagation network. In: D.S. Touretzky (ed.) *Advances in Neural Information Processing Systems 2*, Morgan-Kaufmann, pp. 396–404. URL: <http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf>.
- [7] Krizhevsky A., Sutskever I. & Hinton G.E. (2012) Imagenet classification with deep convolutional neural networks. In: F. Pereira, C.J.C. Burges, L. Bottou & K.Q. Weinberger (eds.) *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pp. 1097–1105. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- [8] Dumoulin V. & Visin F. (2016), A guide to convolution arithmetic for deep learning. URL: <http://arxiv.org/abs/1603.07285>, cite arxiv:1603.07285.
- [9] LeCun Y. & Bengio Y. (1998) The handbook of brain theory and neural networks. MIT Press, Cambridge, MA, USA, chap. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258. URL: <http://dl.acm.org/citation.cfm?id=303568.303704>.
- [10] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A. & Bengio Y. (2014) Generative adversarial nets. In: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence & K.Q. Weinberger (eds.) *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pp. 2672–2680. URL: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>.

- [11] Gatys L.A., Ecker A.S. & Bethge M. (2015), A neural algorithm of artistic style. URL: <http://arxiv.org/abs/1508.06576>, cite arxiv:1508.06576.
- [12] Chen T.Q. & Schmidt M. (2016) Fast patch-based style transfer of arbitrary style. CoRR abs/1612.04337. URL: <http://arxiv.org/abs/1612.04337>.
- [13] Simonyan K. & Zisserman A. (2014) Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556. URL: <http://arxiv.org/abs/1409.1556>.
- [14] Johnson J., Alahi A. & Fei-Fei L. (2016) Perceptual losses for real-time style transfer and super-resolution, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9906 LNCS. 694-711 p.
- [15] Johnson J. (2016), Perceptual losses for real-time style transfer and super-resolution: Supplementary material. URL: <https://cs.stanford.edu/people/jcjohns/papers/eccv16/JohnsonECCV16Supplementary.pdf>.
- [16] Ulyanov D., Vedaldi A. & Lempitsky V.S. (2016) Instance normalization: The missing ingredient for fast stylization. CoRR abs/1607.08022. URL: <http://arxiv.org/abs/1607.08022>.
- [17] Li Y., Fang C., Yang J., Wang Z., Lu X. & Yang M.H. (2017) Universal style transfer via feature transforms. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (eds.) Advances in Neural Information Processing Systems 30, Curran Associates, Inc., pp. 386–396. URL: <http://papers.nips.cc/paper/6642-universal-style-transfer-via-feature-transforms.pdf>.
- [18] Zhu J., Park T., Isola P. & Efros A.A. (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR abs/1703.10593. URL: <http://arxiv.org/abs/1703.10593>.