GAN
for
Generating Art
Final Project Report

UNIVERSITY OF
EASTERN FINLAND

School of Computing

11.11.19 - 23.12.19

by

Pix-to-Art Team.

Artificial Intelligence Course

Supervisors
Pekka Toivanen - Taha Nakabi

1. Overview

Our team implemented and commenced the operations of the project Pix2Art. Prior to the execution of such a project we have searched the already existing samples by using a wide diversity of sources like Github, ResearchGate, etc.During the project we used the Google Colab which allows us to work together on one code.So, in this report first we will share the main overview of a Pix2Art and then we will give a brief explanation of CycleGan usage in our project, finally, we will share the results and references. Furthermore, the general objectives and goals of this project will be shared in the following section called project details. Also, you can find the code description phase by phase at the corresponding section.

Note ! The resulting images are in the last section.

1.1. Group Information

Name and Surname	Id	Role
Fatima Rabia Yapicioglu	**306627**	**Coding the cycle-GAN, Creating the skeleton of the report**
Jukka Arponen	**207805**	**Art acquisition + processing**
Rhythm Rajiv Bhatia	**308847**	**Experiment+Results+References**
Tomi Rönkkö	**279581**	**Art acquisition + processing**
Ryan Kelvin Ford	**308833**	**Info about the cycle-GAN**
Zongyue Li	**308839**	**Data collection**
Yuya Tanaka	**309630**	**Introductions**

1.2. Project Description

The relevant and pertinent information regarding the project is provided below:

1.2.1. Project name: Pix2Art

1.2.2. Purpose: Learning the mapping between an input image and an output image using a training set of aligned image pairs. And then implement the following tasks:

- Read the paper about cycleGAN (You will find the network architectures and training data in the Appendix of the paper).
- Implement cycleGAN and train it on art data.
- Take pictures in your city and transform them into art styles.
- Write a report about your work.

1.2.3. Illustration:

Figure 1.1. Original Image from the city

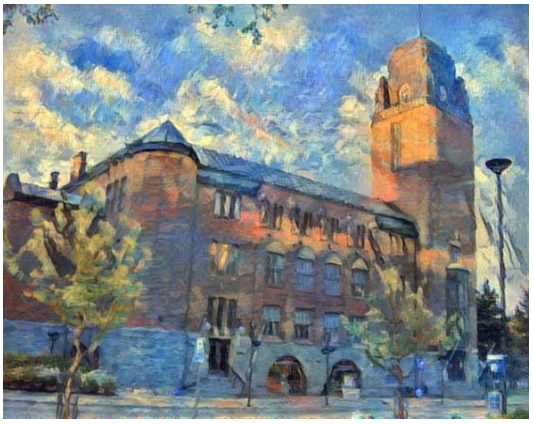

Figure 1.2. CycleGAN generated image

1.2.4. Platforms and Libraries:

1. Google Colab
2. Anaconda Spyder
3. Keras
4. Keras Contrib
5. Tensorflow
6. Matplotlib

Supervisor: Taha Nakabi

2. Introduction

The Cycle-Consistent Adversarial Networks (CycleGAN) is proposed by Zhu J. Y., Park T. et al. (2017) which enables machine image-to-image transformation. This transformation can generate images non-existing before from an image to another unpaired image. (Zhu, Park et al., 2017)

We firstly aimed to implement transformation from city images in Joensuu and Kuopio to Giger's style images. However, it was abandoned due to the hardness to implement. In consequence, our aim becomes to implement transformation from the city images to Monet’s style images.

This report is constructed with seven parts. Part one, Overview, shares the main overview of this project Pix2Art. The second and third parts, Introduction and Information About The CycleGAN, briefly explain the usage of a CycleGAN. The fourth part, Data Collection, and Preprocessing shows how to collect and proceed image data. The fifth part, Experiments and Outcomes, and Results describe the experiment performed in the Pix2Art project and its outcomes. The seventh part, Reference, shows what articles are cited in this report.

The source images shown in this report are from images taken by our members and/or Berkeley’s website.

3.0. Information About the Cycle-GANs

The GAN or Generative Adversarial Network was invented by Ian Goodfellow and his team in 2014. Generative Adversarial Networks consists of two neural networks that compete against each other in a game like a manner considered to be a zero-sum game. The aim of a GAN is for the generator to create an image that has the ability to trick the discriminator into believing that the image is real instead of a generated image.

The general structure and steps for a GAN are displayed below:

- The generator takes in a random number and returns an image
- The image from the generator is then fed into the discriminator with other images that were not created by the generator
- The discriminator then processes the images and categorizes them as a 0 or a 1. With 0 being fake and 1 being a real image.

If the image from the generator is classified as a 1 then this is classified as a success.

GAN's have a double discrimination loop as the discriminator is considered to be a feedback loop with the actual truth for each image and the generator is in a feedback loop with the discriminator in order to trick it.

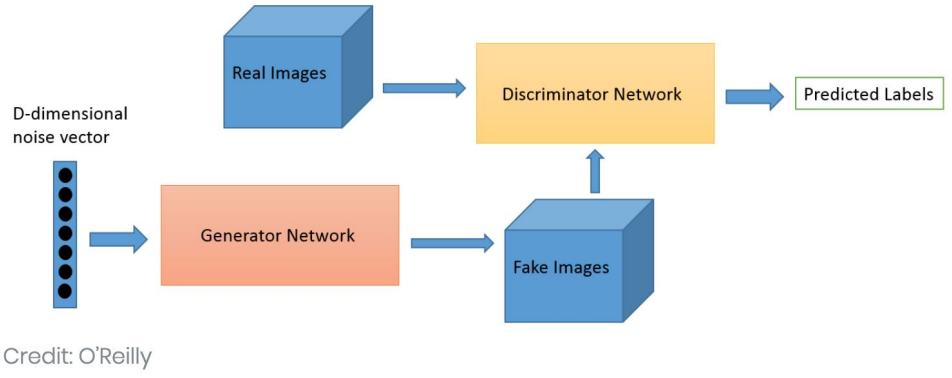

Figure 2.1. Cycle - GAN Structure

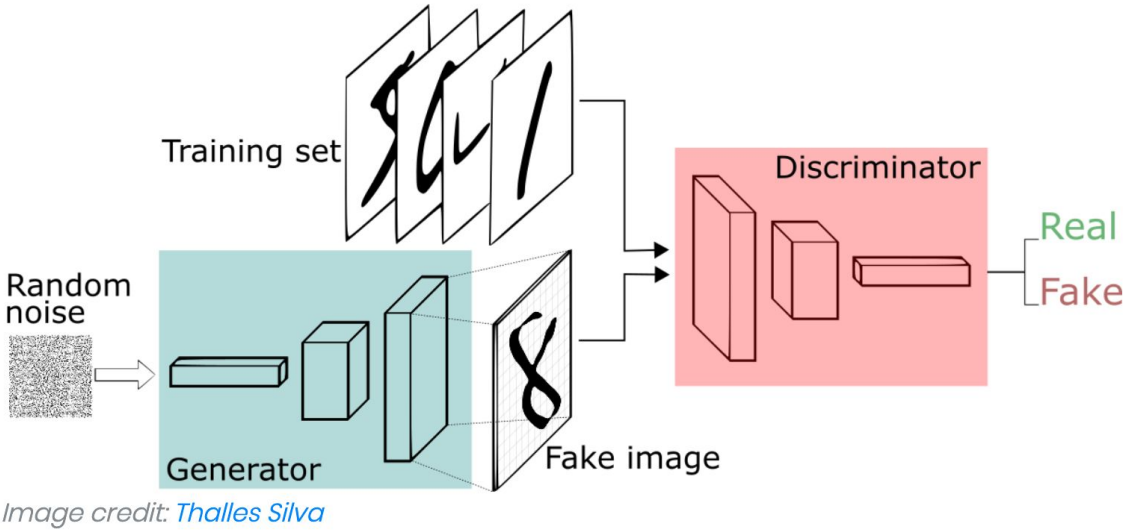

Image credit: Thalles Silva

Figure 2.2. Generator and Discriminator

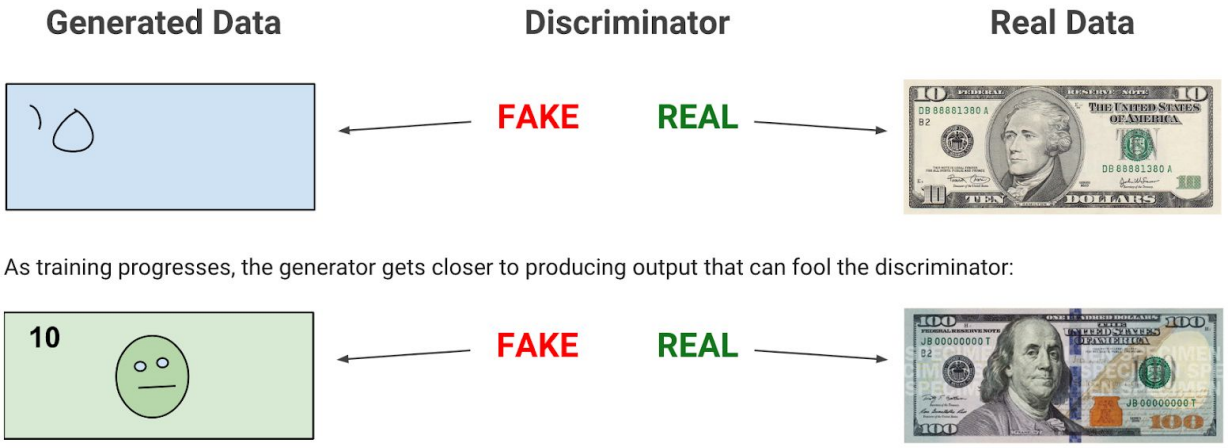

Figure 2.3. Fake or Real

4.0. Data Collection and Pre-processing Methods

4.1.Data Collection Methods

Our initial plan was to use HR Giger art as our training data. However, the data was difficult to collect as no existing dataset was ready. We found a collection of about 1070 images online, which is quite small.

We used photoshop to resize the images to 256x256px to speed up the process of training the network.

Ultimately we used Monet art as a reference for training. The dataset was much better than Giger as it had more pictures in it and it was already scaled to 256x256px. It was also easily available at Berkeley's website.

We gathered images from Joensuu and Kuopio. Some of these images were taken by ourselves and some of them were collected from Google and Baidu image search. Most of the group members were from Joensuu so mainly the images from there were used.

4.2. Data Pre-processing Methods

Preprocessing was important to speed up the training. We used Adobe Photoshop and Irfanview to resize the images to 256 x 256px.

Some of the images had to be manually cropped to preserve image aspect ratios, this was done in Windows Photos. The cropping was done to reduce the original image into a square so that the 256 x 256px resizing did not distort the image. We saved the picture as a .jpeg again, although leading two a "double compression", this did not lead to a noticeable degrade in image quality.

The Giger's art canvases were of various different sizes, so we cropped almost all of them to square proportions. Photoshop batch processing was not 100% reliable, leading to an alternating centering of the cropped images. This lead to the need of manually cropping some of the images. This may be because of the pictures processed were of various pixel dimensions, and the Photoshop batch processing tool for cropping was not designed to handle these kinds of varying dimensions, or maybe it is a bug in the software. Monet artwork came already out in much higher, preprocessed quality and was more usable in this task (neural network training).

Luckily there were only 20 pictures of Joensuu and Kuopio each that had to be manually cropped. We cropped the images so that the location of each photo would be somewhat identifiable.
Both Irfanview and Photoshop had a batch resize functionality which made the final step of resizing easy.

5. Experiments and Outcomes

5.1. Real to Monet Style Art Machine

First, we will show some important code descriptions and parameters in this part, and then we will share the experiments and some important outcomes. We imported *monet2photo* but used it in a reverse manner so that our machine would transform real images into monet style art.

Importing the dataset, and using in a reverse manner,

```
# Loading the  dataset via tfds, already imported module
# We used the monet2photo dataset reversely so that it can transform the photos
into monet images
data, metadata = tfds.load('cycle_gan/monet2photo', with_info=True,
as_supervised=True)

# Splitting the dataset into train/test
train_x, train_y, test_x, test_y = data['trainB'], data['trainA'],
data['testB'], data['testA']
```

Defining some important parameters like epochs, learning rate, row and column size, number of channels and the optimizer,

```
# Settings
epochs = 50
# We trained the model with the 50 epochs
LAMBDA = 10

img_rows, img_cols, channels = 256, 256, 3
```

```
# We used 256 x 256 images, and since it is colourful there are 3 channels
weight_initializer = RandomNormal(stddev=0.02)

# The learning rate is 0.002
gen_g_optimizer = gen_f_optimizer = Adam(lr=0.002, beta_1=0.5)
dis_x_optimizer = dis_y_optimizer = Adam(lr=0.002, beta_1=0.5)
```

After coding the generators and discriminators, we defined our model as follows,

```
# Define the models
generator_g = generator()
generator_f = generator()

discriminator_x = discriminator()
discriminator_y = discriminator()
```

And then we defined a method called generate_images() to produce art style converted images. Generator for art

5.2. Sample Experiments

5.2.1 GANs

Generative Adversarial Networks (GANs) have impressive results in the image to image translation. Adversarial loss is the reason for these results. This loss makes the network learn the mapping between the source and target images.

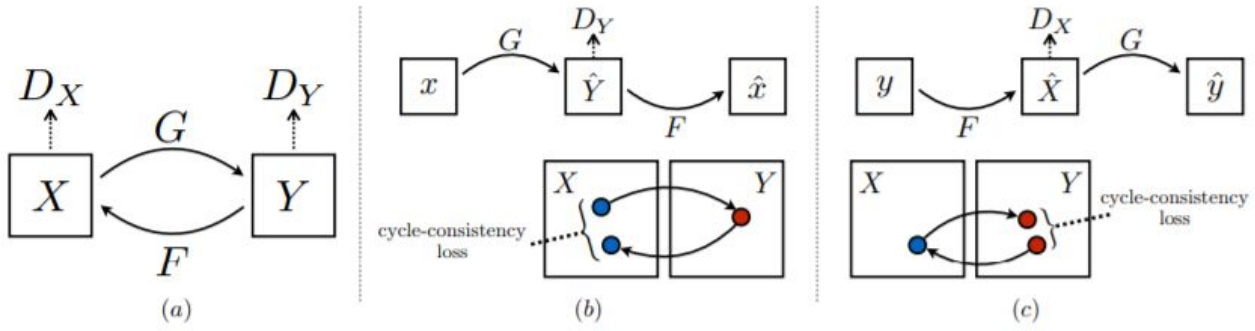

Figure : (a) Two mapping functions G (X to Y) and F (Y to X).

Dy and Dx are discriminative functions. Dx encourages X to translate to Y. Dy encourages Y to translate to X. (b) and (c) cycle-consistency loss is introduced to provide intuition in the network. (b) has forward-cycle consistency loss, (c) has backward-cycle consistency loss.

5.2.2 Image to Image translation

In image-to-image translation method we implement a non-parametric model on a single input-output pair of image. We either make it learn a parametric translation function using CNNs or conditional function using conditional GANs, also known as "pix2pix" framework.
Similar ideas have been used for image to art transfer, photographs from sketches, etc.

6. Results

6.1. Real to Monet Art Machine

We trained our model with 50 epochs and during training, after the 23'rd epoch, we started to get good results.

In this phase first, we will share some in order results from the dataset, especially the best ones, and then we will share the real to art conversion pictures from Joensuu. We tested the model nearly with 20 photos from Joensuu.

6.1.1. Some samples from the training set

- Epoch 26, model created the following conversion,

Figure 6.1. Epoch 26 Real to Monet Art Conversion

- Epoch 30, model created the following conversion,

Figure 6.2. Epoch 30 Real to Monet Art Conversion

- Epoch 34, model created the following conversion,

Figure 6.3. Epoch 34 Real to Monet Art Conversion

- Epoch 35, model created the following conversion,

Figure 6.4. Epoch 35 Real to Monet Art Conversion

- Epoch 37, model created the following conversion,

Figure 6.5. Epoch 37 Real to Monet Art Conversion

It is clearly observable that after the 37’th epoch images started to really look like monet style arts.

- Epoch 40, model created the following conversion,

Figure 6.6. Epoch 40 Real to Monet Art Conversion

- Epoch 43, model created the following conversion,

Figure 6.7. Epoch 43 Real to Monet Art Conversion

- Epoch 44, model created the following conversion,

Figure 6.8. Epoch 44 Real to Monet Art Conversion

- Epoch 49, model created the following conversion,

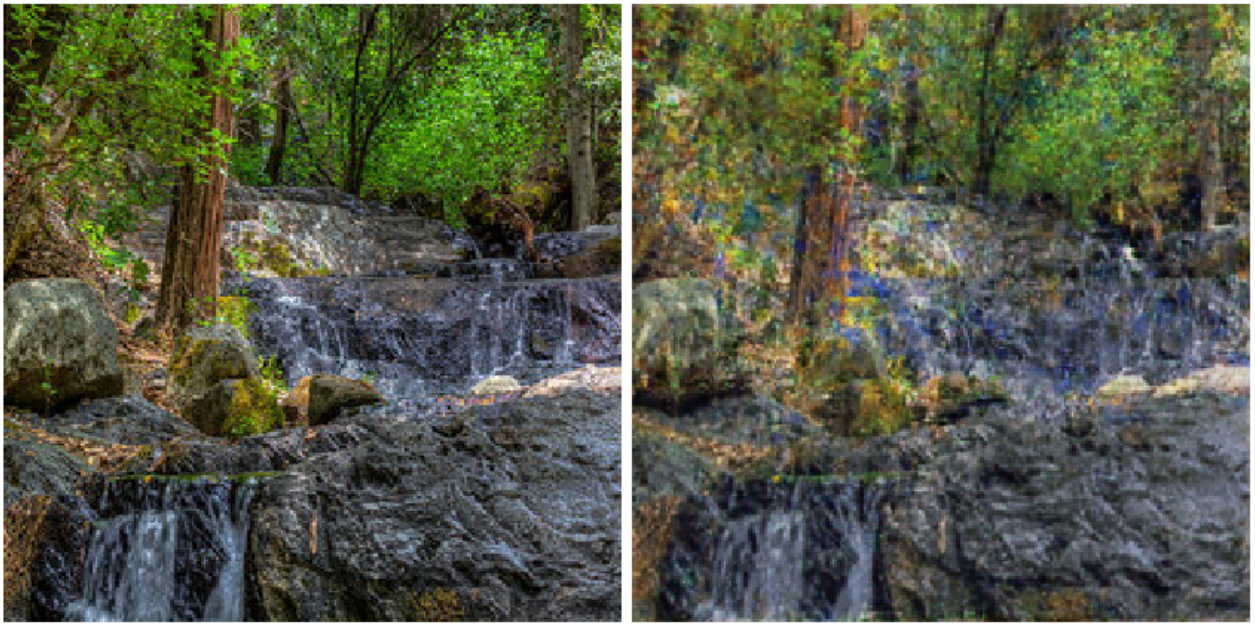

Figure 6.9. Epoch 49 Real to Monet Art Conversion

In the last epoch we got the above conversion and then we tested the model with Joensuu's photos.

6.1.2. Testing the model from with Joensuu's Images

Here are the best results of our conversions from Joensuu after 50 epochs.

Figure 7.1. Real, Joensuu Art Museum

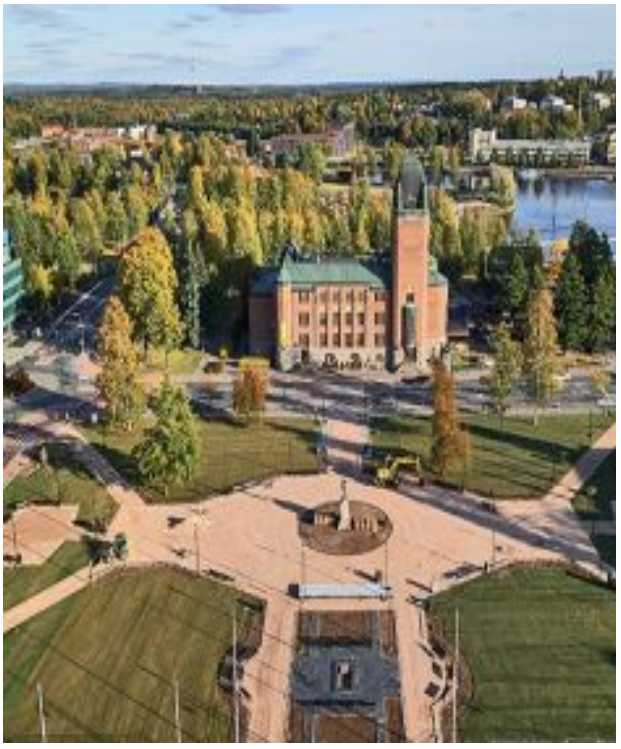

Figure 7.2. Art, Joensuu Art Museum

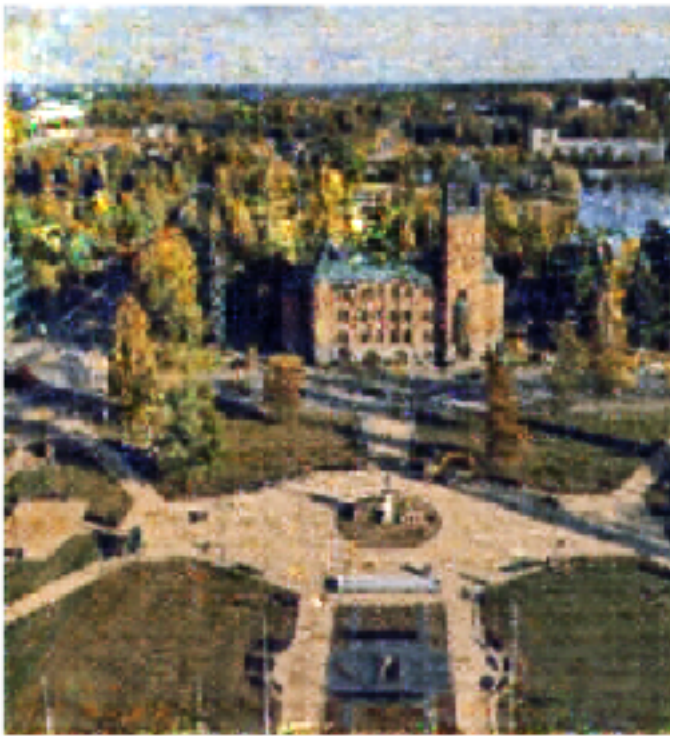

Figure 7.3.Real, Joensuu Aerial Image

Figure 7.4. Art, Joensuu Aerial Image

Figure 7.5. Real, Joensuu church

Figure 7.6. Art, Joensuu church

Figure 7.7. Real, Joensuu shopping center **Figure 7.8. Art, Joensuu shopping center**

Figure 7.9. Real, Joensuu pedestrian path **Figure 7.10. Art, Joensuu pedestrian path**

Figure 7.11.Real, Joensuu office block

Figure 7.12. Art, Joensuu office block

Figure 7.13. Real, Joensuu Aerial Image

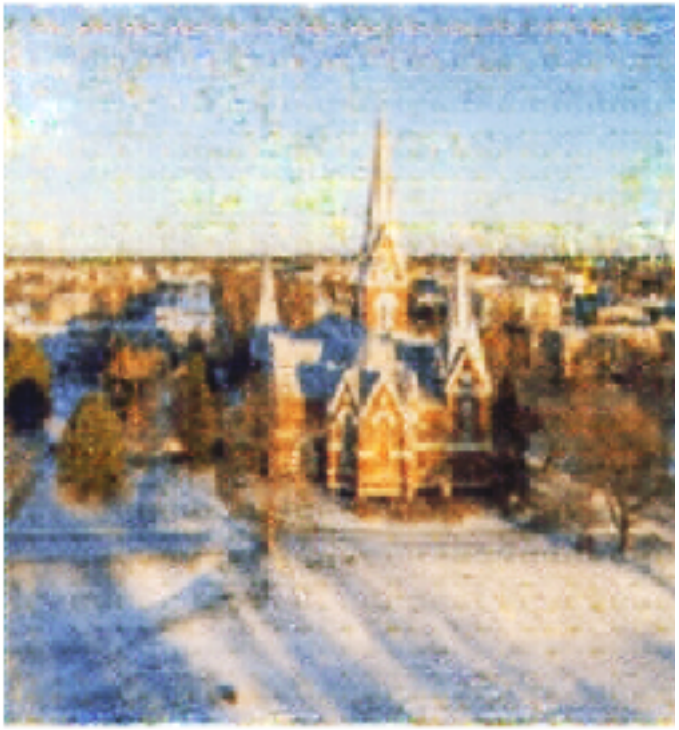

Figure 7.14. Art, Joensuu Aerial Image

Figure 7.15. Real Joensuu Apartments

Figure 7.15. Art, Joensuu Apartments

Figure 7.16. Real, Joensuu Apartments

Figure 7.17. Art, Joensuu Apartments

Figure 7.18. Real, Joensuu Church

Figure 7.19. Art, Joensuu Church

Figure 7.20. Real, Joensuu Apartments

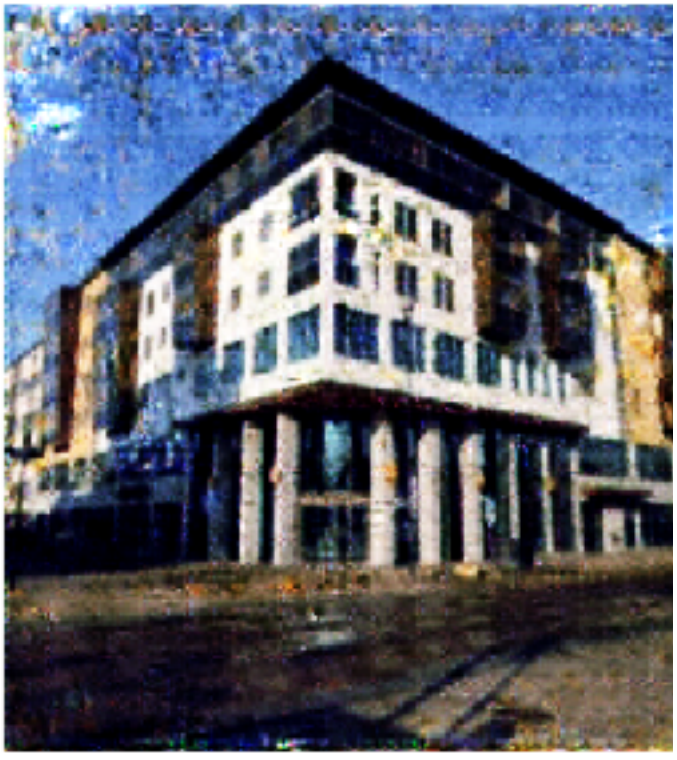

Figure 7.21. Art, Joensuu Apartments

Figure 7.22. Real, Joensuu Aerial Skyline **Figure 7.23. Art, Joensuu Aerial Skyline**

7. References

1) Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu, Jun-Yan and Park, Taesung and Isola, Phillip and Efros, Alexei A, Computer Vision (ICCV), 2017 IEEE International Conference,pages={2223-2232} 2017
2) Image-to-Image Translation with Conditional Adversarial Networks, Isola, Phillip and Zhu, Jun-Yan and Zhou, Tinghui and Efros, Alexei A, Computer Vision and Pattern Recognition (CVPR), 2017
3) https://medium.com/@jonathan_hui/gan-cyclegan-6a50e7600d7
4) Aesthetics of Neural Network Art, Hertzmann, Aaron, arXiv preprint arXiv:1903.05696,2019
5) Learning to Synthesize and Manipulate Natural Images, Zhu, Jun-Yan and Foley, Jim, IEEE computer graphics and applications, volume 39, no. 2, pages={14-23}, 2019
6) A Survey of state-of-the-art GAN-based approaches to image synthesis, Esfahani, Shirin Nasr and Latifi, Shahram

(https://pdfs.semanticscholar.org/eda0/6ef7d3938b343d3985584f1954a5bd5c88c3.pdf.)
7) A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In SIGGRAPH, 2001.
8) G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006
9) I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.
10) J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.