I'm Something of a Painter Myself

Jakub Kučera

ČVUT - FIT

kucerj56@fit.cvut.cz

31. prosince 2022

1 Introduction

This report in on an assignment based on a Kaggle competition called *I'm Something of a Painter Myself*. The goal of this competition is to train a model, which will take an existing real photograph and transform into a Monet style painting.
The description of the mentioned Kaggle competition suggests to use generative adversarial networks (GANs). A simple GAN consists of two models: generator and a discriminator. [1] [3]
The data set includes a set of real photographs and a set of Monet paintings. These two sets are not paired in anyway. Because of that a simple GAN would not be sufficient for this task, and instead a CycleGAN was chosen, for it's ability to learn on unpaired data and transform images. The main differences of CycleGANs are that they consist of two generators and two discriminators and there is a additional new "training flow". where an image from domain A is transferred using one generator to domain B and then back to domain A using the other generator
[9]

2 Data preparation

The structure of the original Kaggle dataset looks like this 2:

- `gan-getting-started/` Root dataset directory
  - `photo_tfrec` Real photos in the TFRecords format
  - `photo_jpg/` Real photos in the JPG format
  - `monet_tfrec/` Monet paintings in the TFRecords format
  - `monet_tfrec/` Monet paintings in the JPG format

The photos in the JPG and TFRecords format are duplicated and the latter is a TensorFlow format, and since I decided to use PyTorch, I deleted the latter versions.
To load the data in Pytorch I used TorchVisions *ImageFolder*. Since it automatically labels data based on subdirectories, I have added an additional nested directory. The final structure looks like this 2:

- `data-monet/` Root dataset directory
  - `monet`
    - `monet/` Monet paintings in the JPG format
  - `photo`
    - `photo/` Real photos in the JPG format

3 Implementation

As already mentioned in the introduction, there are four models: two generators and two discriminators. The architecture for models of the same type is the same.

The generators can be split into three main sections: down-scaling, ResNet (residual neural network) and up scaling layers. [4] The discriminators are built as PatchGANs. [9]

A lot of choices of architecture and argument values are from the mentioned CycleGAN paper [9], but also from some implementation examples [2] [8] [7] [5] [6]

One thing that I would definitely like to improve retrieval of images in the training loop. With the current implementation, in each epoch, there will be the same amount of real images as there are Monet painting, so only about 300 out of 7000.

4 Results

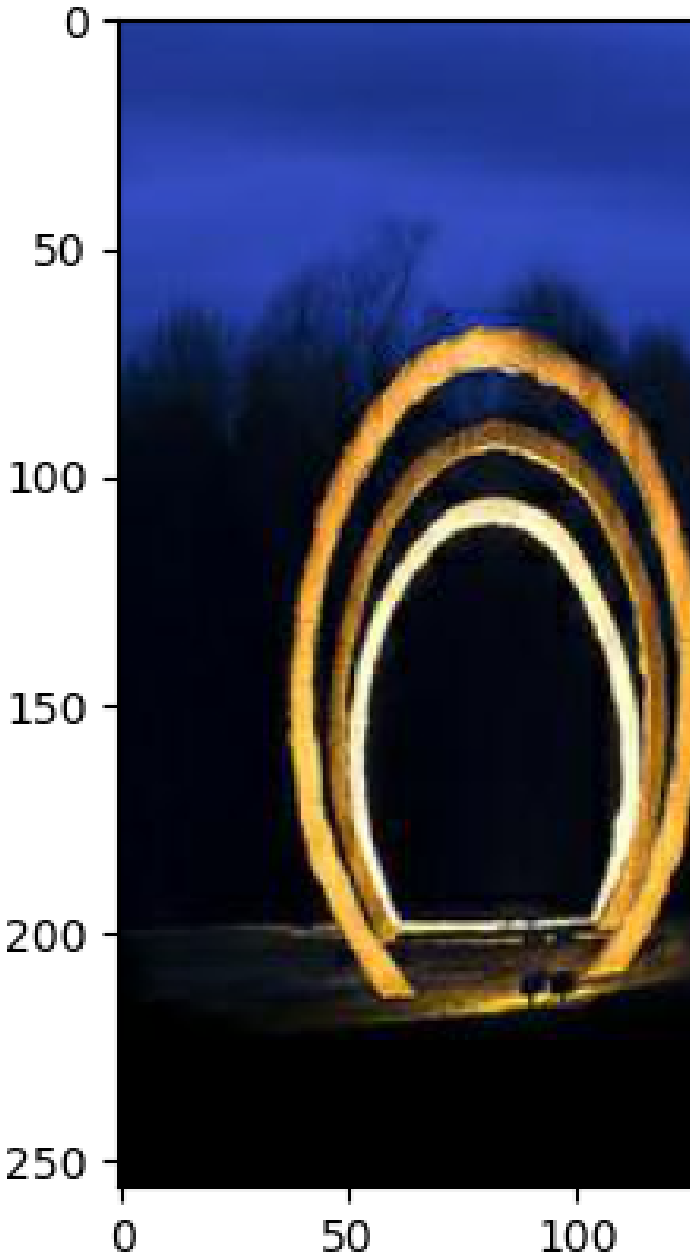
There is an example generated painting 2, which was genareted from the photo 1. As can be seen the result is not very convincing, although this is randomly selected sample and it was trained on a model after only 100 epochs. This chart also represents the different losses across those 100 epochs 3

Reference

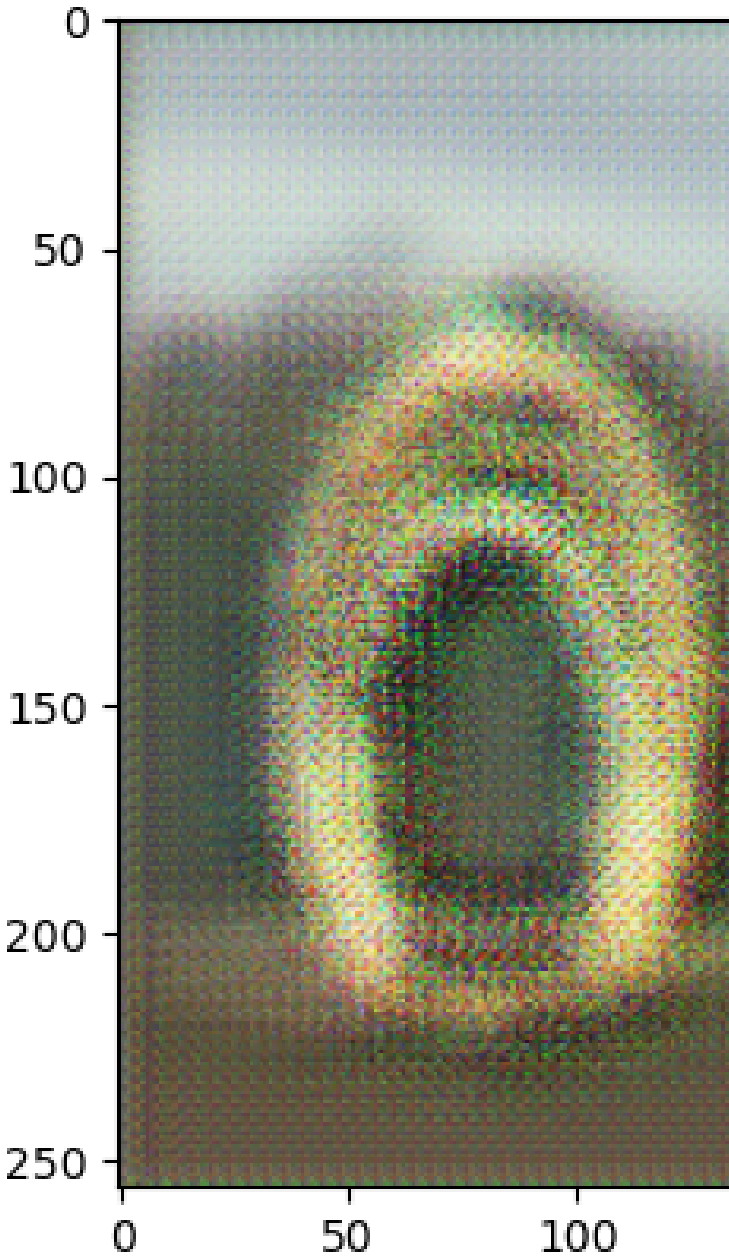
[1] Phil Culliton Amy Jang, Ana Sofia Uzsoy. I'm something of a painter myself, 2020.

[2] Bjørn Hansen. Cycle gan with pytorch. online, 2020.

[3] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks, 2017.



Obrázek 1: Original photo



Obrázek 2: Fake Monet paiting

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

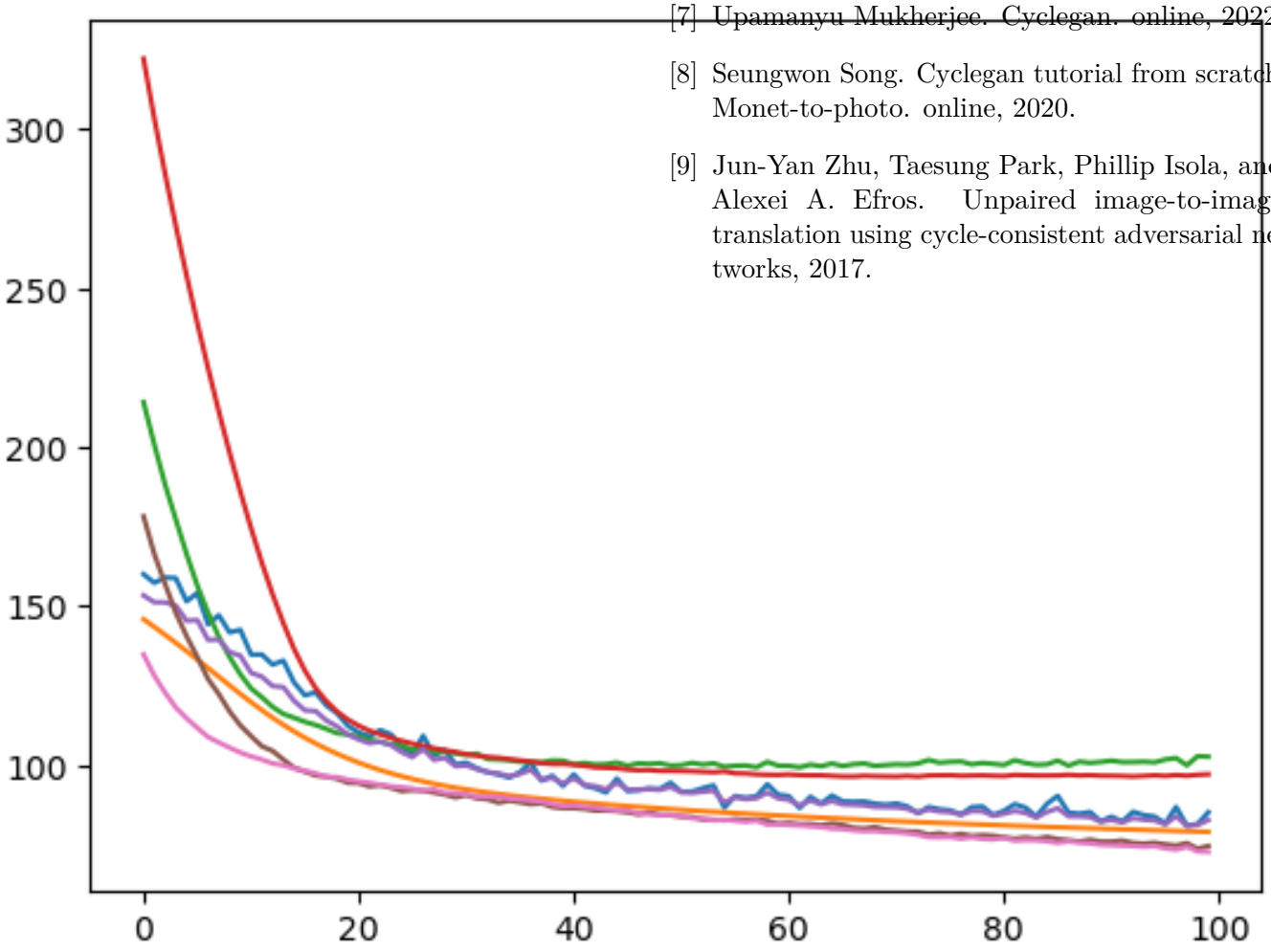
[5] Amy Jang. Monet cyclegan tutorial. online, 2020.

[6] Jun-Yan Zhu, Lambda Will, Hungryof, Tongzhou Wang, Taesungp, Simon Treu, Oren Katzir, mathbbN, Srinath Mannam, Guopzhao, Jonathan Chang. pytorch-cyclegan-and-pix2pix. online, 2022.

[7] Upamanyu Mukherjee. Cyclegan. online, 2022.

[8] Seungwon Song. Cyclegan tutorial from scratch: Monet-to-photo. online, 2020.

[9] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.



Obrázek 3: Loss graph across 100 epochs