Highly Descriptive Vetor-to-Face Generation to Synthezsize Authentic Faces (Photofits for Criminology Purposes) via a GAN

Max Erler and Daniel Schubert
Supervisor: Pingchuan Ma
Chair of Machine Vision & Learning Group: Prof. Dr. Björn Ommer

August 11, 2022

Abstract

This is the report corresponding to the final project of the teaching event "Computer Vision and Deep Learning: Visual Synthesis" in the summer of 2022 at LMU Munich. We designed a framework that currently provides two GAN-Models – cDCGAN and TediGAN – that are easy to train, evaluate and use to generate photofits. It is implemented in PyTorch and highly configurable to accommodate many test cases. Due to its architecture other models, datasets and metrics can easily be added. The corresponding repository of the framework can be found under https://github.com/maximotus/cv-dl22-text-to-photofit-GAN.

Contents

1 Introduction

1.1 The Goal of Photofit Creation using GANs

Our proposed goal for the final project of the "Computer Vision & Deep Learning: Visual Synthesis" lecture was to train existing Generative Adversarial Network (GAN, compare [8]) models and fine-tune their architectures in respect to generate authentic and unambiguous samples of real looking faces. These samples should be of such quality that they could be used as photofits (also phantom images, i.e. pictures representing a person's memory of a criminal's face, compare [5]). Even if this common task is already done by phantom sketch artists working for the police or for lawyers, our assumption is that a GAN creating such photofits could easily outperform every sketch artist in terms of costs, speed, accuracy and photo-realism. Thus, such a network for the creation of photofits could be a very useful tool for criminology workers.

1.2 Text-to-Face Synthesis

Inspired by DALL-E and DALL-E 2 (see [19]), our first intention was a text-to-image approach using two separate models – analogous to [26]. On the one hand, we considered to use a text-encoder model for the embedding of a continuous text describing a criminal's face into the latent space – such that the semantics of the textual description remain intact. On the other hand, we thought about a GAN or VAE model creating faces from those latent embeddings.

1.3 Vector-to-Face Synthesis

However, during the execution of the project we changed our plan to only focus on the generation part because of four crucial arguments. First, the lecture is about visual synthesis and not natural language processing, so our main focus should be on the creation of images and not on the semantic embedding of continuous text into the latent space. Second, training a text-

encoder and a GAN respectively means twice as much calculation time which is inappropriate for the relatively short project time. Third, in respect to our described goal (see 1.1) we think that possible downstream applications would benefit more if the photofit creation is conditioned by vectors with values either 0 or 1 representing the truth value (0=False, 1=True) for each descriptive attribute of an image / a face. Fourth, the attempt of only generating phantom images based on attribute vectors is sufficient to get a proof of concept and to use such model for criminology purposes.

Therefore, to keep things simple and appropriate we decided to focus solely on the principle of using a GAN network – consisting of two separate models, i.e., a discriminator / encoder and a generator / decoder [8].

We finalized a more stable adaption of a classical GAN, namely a Deep Convolutional GAN (DCGAN) which explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator respectively [21]. Since we do not just need to generate random images, but rather images that fit the vectorized description of a criminal's face we conditioned our DCGAN to also use the attribute vector as input – then it is called a Conditional DCGAN (CDCGAN).

Moreover, we decided to re-implement another CDCGAN architecture: tediGAN [26]. Due to time constraints and some other reasons – see 2.2.3 – we were not able to finalize the re-implementation of the tediGAN.

1.4 Related Work

The number of papers concerning the same topic as ours, text-to-face generation from attributes, is small. We found 17 papers based on a broad range of keywords. The papers can be divided into two groups. One working with photofits/ forensic or composite sketches and the other with attribute guided face generation. The first group is mainly focused on generating images from those sketches [9, 14, 22]. This is not what we intended to do. The other group employs networks to generate faces from attributes which matches our goal [1, 2, 3, 4, 6, 7, 15, 17, 20, 25, 26, 27, 28]. We selected three papers which give relevant information for our project. First "TediGAN: Text-Guided Di-

verse Face Image Generation and Manipulation" by Xia et al. published in 2021 [26]. The second paper is "Attribute-Guided Sketch Generation" by Tang et al. published in 2019 [23], which is the only paper bringing both aspects together. And third DCGAN paper from Goodfellow et al. [21].

2 Main

2.1 Dataset

2.1.1 Suitable Datasets

During our research in the project planning phase we stumbled across various datasets that could be useful to us in terms of their properties. Some examples of datasets containing faces and corresponding descriptions or attributes are:

- celebA, see [16]
- celebA HQ, see [12]
- LFW, see [11]
- MAAD-Face, see [24]

Nevertheless, these datasets are not made for criminology purposes, and so they do not perfectly fit our approach of generating photofits. This has two main reasons.

First, baseline face datasets are mainly constructed for face recognition applications. On the one hand, as the authors of MAAD-Face outline – compare [24] – this leads to the consequence that datasets like celebA or LFW indeed contain a large amount of face images, but struggle with the overall annotation correctness and the total number of attributes. On the other hand, MAAD-Face aims to be better in those terms by merging face image datasets with their attribute annotations together and check their correctness by a human evaluation.

Second, as we already expected beforehand and was confirmed during the project execution, such relatively low numbers of distinctive attributes – compare table 1 – would not fit the demand for accuracy needed for phantom image creation. Moreover, considering the 40 attributes of celebA one can see that there is some redundancy and incompleteness within – e.g. one extra attribute for each Black_Hair, Blond_Hair, Brown_Hair, and Gray_Hair, but there is no attribute like Red_Hair.

Dataset	Number of images	Number of Attributes
LFW	$\approx 13,200$	74
CelebA	$\approx 200,000$	40
CelebA-HQ	$\approx 30,000$	40
MAAD-Face	$\approx 3,300,300$	47

Table 1: Dataset statistics

2.1.2 Our Decision

Comparing the statistics of the datasets (see 1), we concluded to initially use a set that has a good trade-off between the total number of face images and the total number of distinctive attributes. Even if they are not perfectly fitted for criminology purposes, our assumption is that if the concept of attribute-conditioned face generation works on one of these datasets, it will also work on more accurate datasets that could be developed especially for the task of photofit creation in the future. And especially, it will work better on a better suited dataset.

Even if MAAD-Face aims to be better than celebA and LFW, we decided to use celebA. MAAD-Face has too many images as than we could manage to train our GAN on it in the given time frame of the project and LFW has too few images.

2.2 Framework

2.2.1 Architecture / Structure

In respect of a uniform and consistent setting for the experiments on different models and datasets we decided to implement a highly configurable and extendable framework. Having such a framework that is easily and uniformly executable for different combinations of configurations is a very powerful tool – especially regarding adapted experiments / work in the future.

The key features consist of a clear experiment configuration via a yaml-file (1), the simple extensibility of the models, datasets and metrics (2) as well as a complete and useful documentation of the results (3).

Regarding the key feature (1), a detailed documentation on such a configuration file can be found below – see 2.2.5.1. On the program start the specified configuration file is parsed in the module misc/main.py. For the cases of any misconfiguration an error designed especially for such cases – compare misc/error.py – will be thrown and the program stops.

The key feature (2) led to the decision to implement a predominantly generic module training.py which holds a parent-child class-structure that can handle any learning or evaluation of a photofit creation model based on machine learning. The inheritance concept represents the three fundamental execution types / modes of the framework, namely train (short for training, corresponding class: Trainer), eval (short for evaluation, corresponding class: Evaluator) and gen (short for generation, corresponding class: Creator). Moreover, the training of a pretrained model can be continued providing a path to the pretrained model and epoch checkpoint. The huge benefit of this key feature is the re-usability of pretrained models as well as their generic evaluation and the generation of photofits.

The evaluation of training results is easy due to key feature (3). The module main.py creates a clearly arranged directory with an unique name – i.e., the timestamp of the execution – holding all results (e.g., samples), statistics (e.g., losses and probabilities), model-checkpoints, the logs and a snapshot of the configuration file from the beginning of the execution.

2.2.2 CDCGAN

As already mentioned in 1.3, a CDCGAN is a GAN using convolutional and convolutional-transpose layers – compare [21]. We implemented a classical approach in the module model/cdcgan.py using a discriminator model and a generator model – compare [8]. The discriminator's job is to distinguish between real and fake samples whereas the generator's job is to produce samples that make the discriminator believe they are real. Since our goal is the vector-to-face synthesis we also conditioned both of the models so they are able to learn the context of an image and its attributes. So the generator will especially be able to sample images from the latent space that fit a given attribute vector. To make the network architecture work on the celebA dataset we had to rethink the whole layer architectures and the principle of concatenating the attribute vector with the images and the noise vector in the discriminator and generator respectively. The final architecture of our CDCGAN is represented in 2.2.2.

```
Generator(
  (initial_z): Sequential(
    (0): Upsample(scale_factor=4.0, mode=nearest)
    (1): Conv2d(128, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (initial_c): Sequential(
    (0): Upsample(scale_factor=4.0, mode=nearest)
    (1): Conv2d(40, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (block): Sequential(
    (0): Upsample(scale_factor=2.0, mode=nearest)
    (1): Conv2d(1024, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU(inplace=True)
    (4): Upsample(scale_factor=2.0, mode=nearest)
    (5): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): ReLU(inplace=True)
    (8): Upsample(scale_factor=2.0, mode=nearest)
    (9): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): Upsample(scale_factor=2.0, mode=nearest)
    (13): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (14): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (15): ReLU(inplace=True)
    (16): Conv2d(64, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): Tanh()
  )
)
Discriminator(
  (sig): Sigmoid()
  (relu): LeakyReLU(negative_slope=0.2, inplace=True)
  (conv_x): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
```

```
  (dropout_x): Dropout(p=0.2, inplace=False)
  (conv_y): Conv2d(40, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (dropout_y): Dropout(p=0.5, inplace=False)
  (conv1): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (dropout1): Dropout(p=0.2, inplace=False)
  (conv2): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (dropout2): Dropout(p=0.2, inplace=False)
  (conv3): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (dropout3): Dropout(p=0.2, inplace=False)
  (conv4): Conv2d(1024, 2048, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  (bn4): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (dropout4): Dropout(p=0.2, inplace=False)
  (conv5): Conv2d(2048, 1, kernel_size=(2, 2), stride=(1, 1), bias=False)
)
```

2.2.3 TediGAN

TediGAN is a GAN and Framework proposed by Xia et al. To generate their images they use an inverted pretrained StyleGAN. The Framework includes multiple options to choose between layers and StyleGANs. Unfortunately neither the git-repository nor their paper provides clear information which was their final and best version. The framework also uses config files in an incomprehensible way. We tried to implement the network into our framework but couldn't get it to start training. Problems we encountered were the configs, which we replaced by one single choice. They also wrote certain layers in C++ and Cuda which we initially struggled with but in the end got to work. But then we encountered a dimension error which was weird due to all shapes matching one another. To resolve the dimension error we logged and followed the flow of the images in the fit() method. To get further information about the configs we contacted the authors but never got an answer. To check for implementation errors we also cloned their repository and tried executing their proposed way to train with their framework. It failed to start due to missing config options.

2.2.4 Metrics

Regarding the metrics we orientated us among the most frequently used ones from the papers we read and chose the four most relevant. In regard of image generation normal metrics, like accuracy, are very relative and should not be

used due to their lack of information value. In most ML context this would not apply. When training a discriminator to decide if a picture is from a real dataset or generated from the GAN's generator, the scores most of the time do not result in "good" as in real images. A human could easily differentiate both. So for image generation and their realness one should use one of the following metrics for evaluating the new images: on the overall similarity the Fréchet Inception Distance (FID), patch similarity (Learned Perceptual Image Patch Similarity, LPIPS), Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) or a metric using a discriminator specifically trained for this task, where you know the results are satisfactory. FID calculates the Fréchet distance which is originally used for the distance between curves of functions but can also be used for the probability distribution, in our case two datasets [10].

$$d^2 = |\mu_X - \mu_Y|^2 + tr(\sum_X + \sum_Y -2(\sum_X \sum_Y)^{\frac{1}{2}})$$

One way to check patch similarity between two images is using LPIPS. This metric is based on comparing similarity of activations in a predefined network. A lower score is better [29]. To check overall spatial image quality one could use BRISQUE. It checks for e.g. noise or blurriness. Kynkäänniemi et al. propose an improved precision and recall framework, which can additionally to precision and recall scores also calculate a realism score. They use a StyleGAN to evaluate a set of images [13]. Some papers also use humans to give feedback on realness of generated images, which can be unreliable and the number of samples to review is limited [26].

In our framework we implemented FID, LPIPS and BRISQUE. Kynkäänniemi's metric is implemented in an outdated version of TensorFlow. While implementing each metric a few key differences appeared which do not get clarified by any paper: e.g. LPIPS calculate the similarity between two pictures. But are they chosen at random or is an order selected in the beginning and then those images get compared? We chose to generate images based on the same attribute-vectors and compare those to one another.

2.2.5 Experiments

2.2.5.1 Configuration

Below you can see a template configuration file for the training of the CDCGAN model on the celebA dataset. This example configuration outlines that our framework is highly configurable.

```
mode: train # (train / eval / gen)
log_level: 20 # CRITICAL = 50, ERROR = 40, WARNING = 30, INFO = 20, DEBUG = 10, NOTSET =
    0
device: cpu # (cuda / cpu / auto)
experiment_path: ..\experiments # directory where the results should be persisted
epochs: 500 # number of epochs
num_imgs: 20 # how many randomly generated images should be saved
predefined_images: {
    max:
        [0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,1,1,0,0,1,0,1,0,0,0,0,1,0,0,0,0,0,0,1],

    daniel:
        [0,1,1,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,1,0,1,0,1,0,0,1,0,0,0,0,0,0,1]}
         # predefined attribute vectors for the image generation

frequencies:
  save_freq: 1 # (i.e. how often should the model checkpoint be saved, in epochs)
  gen_freq: 1 # (i.e. how often should test images be generated, in epochs)

dataloader:
  dataset: celebA # name of the dataset
  size_fraction: 4 # 4 means use 1/4th of the dataset
  batch_size: 128 # number of images processed in one forward-call of the models
  image_size: 64 # image size (64 x 64)

model:
  name: CDCGAN # name of the mode
  pretrained_path: ../experiments/train/template-CDCGAN-train/2022-08-10-10-08-52 # empty
      if start from scratch
  start_epoch: 0 # empty if start from scratch
  criterion: BCELoss # name of the loss function
  optimizer: Adam # name of the optimizer
  learning_rate: 0.001 # learning rate for the optimizer
  parameters:
    dropout: 0.2 # how many neurons should be deactivated during forwarding
    alpha: 0.1 # model specific parameter
    beta1: 0.1 # model specific parameter
    ngf: 64 # number of features in the generator
    ndf: 64 # number of features in the discriminator
    z_channels: 128 # size of the noise vector
    use_spectral_norm: False # whether to wrap convolutional layers in spectral_norm or
        not
```

2.2.5.2 Results

We planned the training process to run every model variation at least once for 100 epochs on a quarter of the celebA dataset. Each Training run took between 9 and 11 hours. After this preliminary phase we looked at the generated images, loss, accuracy and metrics of our networks. We then decided

network	accuracy real accuracy fake before disc accuracy fake after disc	loss real loss fake loss gan	FID	LPIPS	BRISQUE	percieved realness (between 0 and 5)
dropout=0 spectral=False, DS_size=1/4	0.994 0.006 0.0007	0.0072 0.0074 13.868	339.957	0.379	16.946	1
dropout=0 spectral=True DS_size=1/4	0.968 0.031 0.009	0.046 0.411 6.815	139.774	0.161	34.917	2.5
dropout=0.2 spectral=False DS_size=1/4	0.894 0.105 0.062	0.210 0.211 6.640	124.138	0.156	31.376	2.5
dropout=0.2 spectral=True DS_size=1/4	0.969 0.028 0.062	0.0418 0.0356 6.6371	168.228	0.265	44.231	2
dropout=0.3 spectral=False DS_size=1/4	0.826 0.180 0.129	0.393 1.003 3.766	139.830	0.294	29.432	1
dropout=0.3 spectral=True DS_size=1/4	0.952 0.046 0.022	0.0612 0.0616 5.7318	173.991	0.205	41.228	1
dropout=0.3 spectral=True DS_size=1	0.999 7.308 7.210	1.12 0.0 44.171	218.512	0.271	28.791	1
dropout=0.5 spectral=False DS_size=1/4	0.506 0.501 0.496	0.750 1.737 0.768	145.371	0.327	73.911	1
dropout=0.5 spectral=True DS_size=1/4	0.821 0.179 0.131	0.263 0.262 3.003	141.477	0.353	45.244	2
dropout=0.5 spectral=True DS_size=1	0.929 0.070 0.048	0.108 0.107 5.587	135.339	0.213	64.889	1.5

Table 2: Experiments and results

that a dropout of 0.3 or 0.5 and spectral convolution layer were beneficial. Thus we ran those on the entire dataset size. Those runs took 12 hours on our system. We also tried out running a model for 200 epochs but noticed mode collapse happened every time. Mode collapse also happend when restarting on an epoch without collapse.

Percieved realness is an intuitive score between 0 and 5: 0$\widehat{=}$ just noise, 1$\widehat{=}$ shape recognizeable, $\geq$ 2$\widehat{=}$ can recognize faces, $\geq$ 3$\widehat{=}$ face with noise, $\geq$ 4$\widehat{=}$ face with small artifacts, 5$\widehat{=}$ real faces without errors. DS_size means dataset size.

When deciding which network was the best you can proceed based on statistics, on the proposed metrics or visually judgeing the generated images per epoch and foremost the last epoch. One could also proceed based on

Figure 1: Mutliple eyes in one face.

theoretically taught metrics, e.g. generator accuracy and discriminator accuracy should meet at 0.5 or at least converge against each other. In this case the network with spectral convolution and dropout value being 0.3 is supposed to be the best one, see fig.7 and 8 But even on first glance every human would be able to differentiate between real and fake images. Running metrics proposed in the metrics section:2.2.4, on trained networks resulted in some mismatching images even compared to the generated ones in the last epoch. When going on the metrics FID is probably the most meaningful. The best network according to this metric would be with dropout 20% and without a spectral convolutional layer. Some generated images are just noise. Every now and then you can recognize parts of a face but the rest is still just random artifacts. So the results are only partly acceptable. The network with 0% dropout without a spectral layer is correctly the worst variant with the highest FID score of 339.957 and images looking like weird color sprinkles, see figure 6. When looking through all images of every last epoch the best network is tied between (dropout=50%, spectral=True, DS_size=1/4) and (dropout=0%, spectral=True, DS_size=1/4). But even these images still have some artifacts in the image or on the face, the images are highly noisy around the face. Sometimes the network tries to generate two faces into one, fig.1. Also our networks seemed unable to learn the difference between certain attributes. E.g. they are not able to generate blonde people fig.2 or they put sunglasses on people fig.3. The sunglasses phenomenon happend with a constant c-vector which didn't specify it and the glasses would appear and disappear epoch-wise. A relative good result is fig.4 and fig.5. But even those fake images are distinguishable compared to the used dataset.

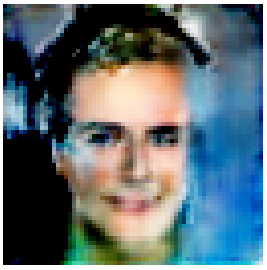

Figure 2: Network struggles to make entire head of hair in blonde.

Figure 3: Network puts glasses on face.

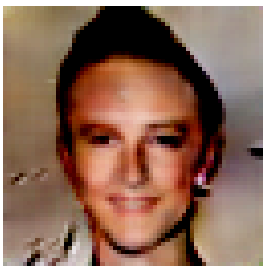

Figure 4: Relatively good picture after 100 epochs. Dropout=0.5, spectral=True

Figure 5: Relatively good picture after 100 epochs. Dropout=0.5, spectral=True

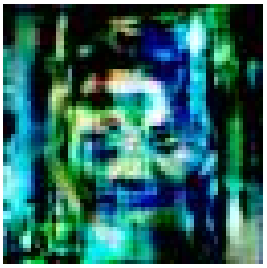

Figure 6: Really bad result.

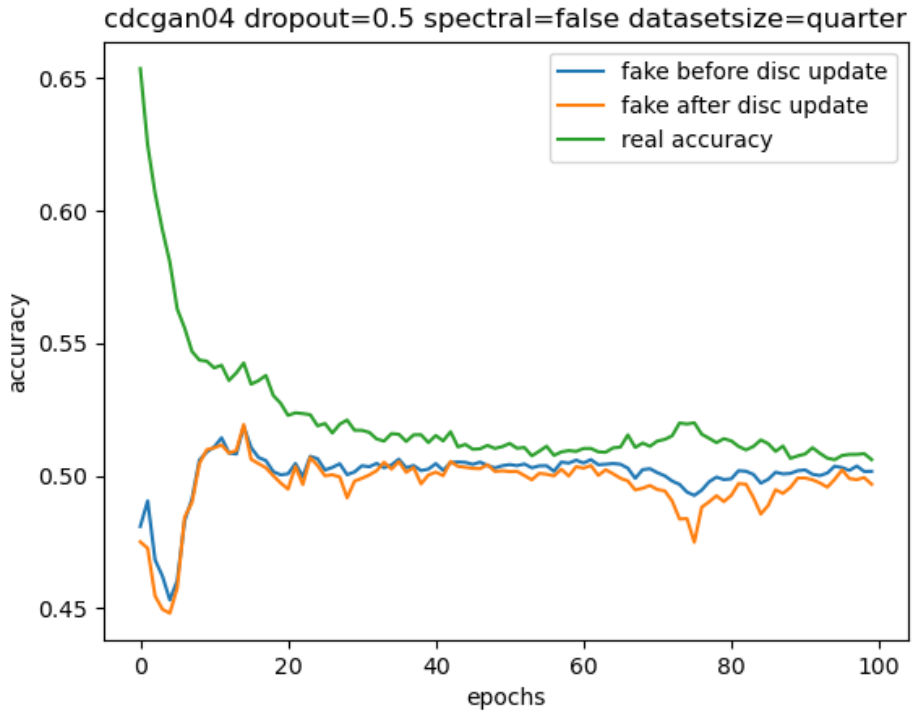

Figure 7: Graph with converging accuracies from GAN and discriminator.

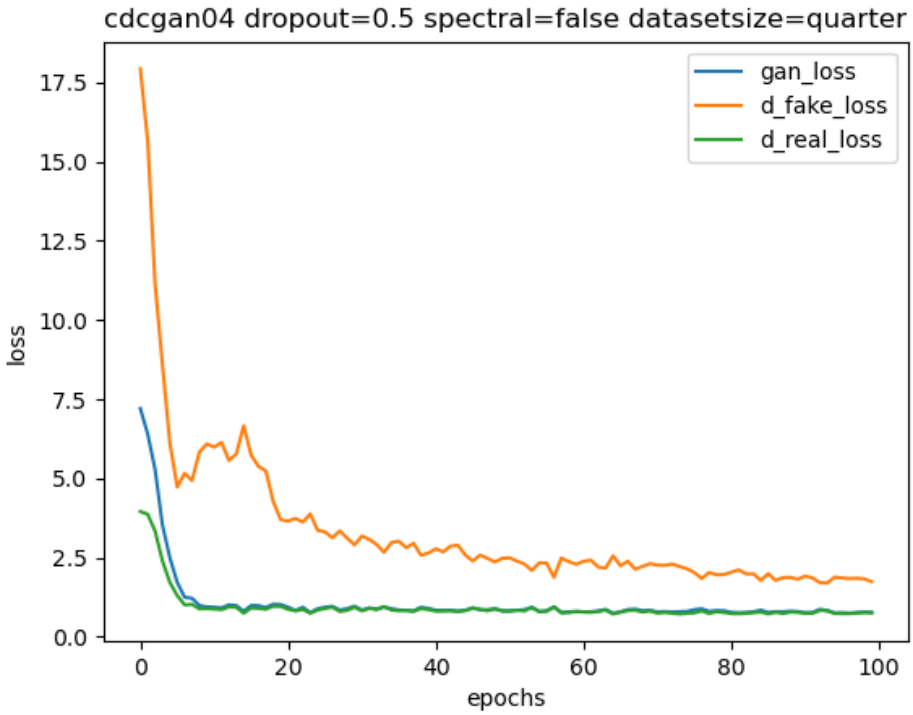

Figure 8: Graph with converging and minimizing loss from GAN and discriminator.

Figure 9: Example image from dataset with artifacts.

3 Conclusion

In this chapter we want to conclude an reflect on the results and things we noticed while implementing or experimenting.

3.1 Datasets

First, we observed some defects in the training images that some faces were stretched or had some artifacts e.g see images 9, 10 and 11. Second, the attributes are redundant and incomplete. E.g., 4 attributes for hair color, Red_Hair is completely missing. Third it would be better if the direction in which a person looks, would only be straight ahead or be labeled. fig.12 and fig.13. Four, due to the small time frame we were unable to run our experiments with the other datasets mentioned above, chapter: 2.1.1. Fifth, often it is the case that many faces match to one and the same attribute-vector so for criminology purposes one should consider to use a dataset with a much larger attribute-vector.

3.2 Mode Collapse

Mode Collapse is a common problem when working with GANs (see this article). Normally a GAN is considered successful if its samples can fool a discriminator and the generator samples diverse images with a distribution like in the real world. This means that given an attribute-vector c the GAN

Figure 10: Example image from dataset with artifacts.

Figure 11: Example image from dataset with artifacts.

Figure 12: Example image from dataset: person looking down and face hidden behind hair and hat.

Figure 13: Example image from dataset: person from the side.

Figure 14: Example for mode collapse after 100 epochs.

should sample different images. Mode collapse happens if the GAN starts to produce the same image again and again for the same c because it successfully fools the discriminator. An example can be seen in fig.14. The image is a result after 100 epochs with spectral convolutions and 0% dropout. As you can see the images all look alike and there is no real difference between them. Also we have checked some attribute-vectors and as one might assume there are multiple individuals annotated with the same vector. This seems pretty plausible with a dataset of more than 200 thousand images and a relatively small attribute-vector size of 40. Therefore we also expected our GAN to sample different images for the same c. However thinking of photofits it could be useful to run into mode collapse, here a specific vector should always lead to the same result.

3.3 Image size

For time and hardware reasons we opted to scale the dataset images to size 64 by 64. We assume that we can achieve better results with bigger sized images due to a higher detail level.

3.4 More Resources

With more time and better hardware (maybe even multiple GPUs) we would have done more extensive testing and more training runs, and prove our hypothesis that a more detailed dataset and larger images lead to a general improvement.

4 Future Work

As described in the conclusion we would like to train our GAN on other datasets, section 2.1.1, and with a larger image size than 64 by 64. Also the development of a new dataset specialized for criminology purposes would be helpful for us and downstream applications. Such a dataset should contain much more than fourty attributes, we assume about 200 would be a good starting point. Moreover the images of the dataset should not include bad samples as pointed out above but only passport photo should be included. Regarding mode collapse it would be interesting if someone could use this phenomenon for photofit generation.

5 Collaboration

We started our project commonly by exchanging our ideas, thoughts and how to approach our topic. We did most of the implementation of our framework by pair programming. Certain parts which only one of us did are marked in the table below.

Implementation	Person
main.py error.py log.py	Max
CDCGAN.py	70-30 Max-Daniel
TediGAN.py	30-70 Max-Daniel
Metrics.py	Daniel
Config.py	Max
Dataset.py	Max
Training.py	80-20 Max-Daniel
Experiments	Daniel

Table 3: Implementation Collaboration

Report-chapter	Person
1.1	Max
1.2	Daniel
2.1	Max
2.2.1	Max
2.2.2	Max
2.2.3	Daniel
2.2.4	Daniel
2.2.5.1	Max
2.2.5.1	Daniel
3	Together
4	Together
5	Together

Table 4: Report Collaboration

References

[1] Navaneeth Bodla, Gang Hua, and Rama Chellappa. Semi-supervised fusedgan for conditional image generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[2] Zhu-Liang Chen, Qian-Hua He, Wen-Feng Pang, and Yan-Xiong Li. Frontal face generation from multiple pose-variant faces with cgan in real-world surveillance scene. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1308–1312, 2018.

[3] Jia Deng, Gaoyang Pang, Zhiyu Zhang, Zhibo Pang, Huayong Yang, and Geng Yang. cgan based facial expression recognition for human-robot interaction. *IEEE Access*, 7:9848–9859, 2019.

[4] Prithviraj Dhar, Ankan Bansal, Carlos D. Castillo, Joshua Gleason, P. Jonathon Phillips, and Rama Chellappa. How are attributes expressed in face dcnns? In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pages 85–92, 2020.

[5] Cambridge dictionary. photofit (picture). `https://dictionary.cambridge.org/de/worterbuch/englisch/photofit-picture`. Accessed: 2021-08-11.

[6] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class project for Stanford CS231N: convolutional neural networks for visual recognition, Winter semester*, 2014(5):2, 2014.

[7] Baris Gecer, Binod Bhattarai, Josef Kittler, and Tae-Kyun Kim. Semi-supervised adversarial learning to generate photorealistic face images of new identities from 3d morphable model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[9] Hu Han, Brendan F. Klare, Kathryn Bonnen, and Anil K. Jain. Matching composite sketches to face photos: A component-based approach. *IEEE Transactions on Information Forensics and Security*, 8(1):191–204, 2013.

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[11] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[12] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[13] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[14] Yingtao Lei, Weiwei Du, and Qinghua Hu. Face sketch-to-photo transformation with multi-scale self-attention gan. *Neurocomputing*, 396:13–23, 2020.

[15] Yaoyao Liu, Qianru Sun, Xiangnan He, An-An Liu, Yuting Su, and Tat-Seng Chua. Generating face images with attributes for free. *IEEE Transactions on Neural Networks and Learning Systems*, 32(6):2733–2743, 2021.

[16] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[17] Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Attribute-guided face generation using conditional cyclegan. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[18] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708, 2012.

[19] openai. Dall-e. `https://openai.com/dall-e-2/`. Accessed: 2021-08-11.

[20] Naima Otberdout, Mohamed Daoudi, Anis Kacem, Lahoucine Ballihi, and Stefano Berretti. Dynamic facial expression generation on hilbert hypersphere with conditional wasserstein generative adversarial nets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):848–863, 2022.

[21] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[22] M.S. Sannidhan, G. Ananth Prabhu, David E. Robbins, and Charles Shasky. Evaluating the performance of face sketch generation using generative adversarial networks. *Pattern Recognition Letters*, 128:452–458, 2019.

[23] Hao Tang, Xinya Chen, Wei Wang, Dan Xu, Jason J. Corso, Nicu Sebe, and Yan Yan. Attribute-guided sketch generation. *2019 14th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2019)*, 4:1–7, 2019.

[24] Philipp Terhörst, Daniel Fährmann, Jan Niklas Kolf, Naser Damer, Florian Kirchbuchner, and Arjan Kuijper. Maad-face: A massively annotated attribute dataset for face images. *CoRR*, abs/2012.01030, 2020.

[25] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[26] Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. Tedigan: Text-guided diverse face image generation and manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2256–2265, June 2021.

[27] Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Elegant: Exchanging latent encodings with gan for transferring multiple face attributes. In

Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[28] Zheng Yuan, Jie Zhang, Shiguang Shan, and Xilin Chen. Attributes aware face generation with generative adversarial networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1657–1664, 2021.

[29] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CoRR*, abs/1801.03924, 2018.