Conditional Generative Adversarial Networks for Face Images

Konstantin Dobler
Hasso Plattner Institute
University of Potsdam
14482 Potsdam, Germany
`konstantin.dobler@uni-potsdam.de`

Stefanie Lewandowski
Department of Business Informatics
University of Potsdam
14469 Potsdam, Germany
`slewandowski@uni-potsdam.de`

Kenneth Schröder
Hasso Plattner Institute
University of Potsdam
14482 Potsdam, Germany
`kenneth.schroeder@uni-potsdam.de`

Pascal Schulze
Hasso Plattner Institute
University of Potsdam
14482 Potsdam, Germany
`pascal.schulze@uni-potsdam.de`

Ole Wegen
Hasso Plattner Institute
University of Potsdam
14482 Potsdam, Germany
`ole.wegen@uni-potsdam.de`

Abstract

We propose and evaluate two different generative networks for training on and creating novel face images, a conditional deep convolutional GAN (cDCGAN) and a conditional Progressive Growing GAN (cPGAN). The desired output image can be specified by choosing conditional attributes like hair color, hairstyle, accessories, or gender without needing to change the underlying model. The models are trained on the CelebA dataset, however a custom preprocessing step is applied to increase the training performance. While the cDCGAN has significantly shorter training times, the cPGAN produces more detailed and realistic images. The quality of the produced images varies greatly; some images are indistinguishable from the training data, while others barely look human. The bulk of the results would not pass a "Turing Test" (human authenticity evaluation), however facial structure and characteristics are clearly recognizable.

1 Introduction

Deep learning has continuously gained popularity over the last years, which resulted in several new research areas, one of which are generative adversarial networks (GANs). The aim of generative modeling is to create new authentic fake samples which portray the same characteristics and features as the original data. Mathematically, the original data is drawn from a normal distribution $p(x)$ and the distribution of the generated pictures should be indistinguishable from $p(x)$ (1). The basic idea behind GANs is to make two neural networks compete with each other. A generator is trained to synthesize realistic samples, while a discriminator tries to distinguish the real from the fake ones - we will call them the "adversaries" from now on.

GANs can be used for a variety of tasks including dataset expansion, character generation, aging simulation, and many others. The largest research field in generative models is visual object synthesis

(2; 3; 4). In this paper, we will focus on the generation of novel face images. One major application for this are facial composites for perpetrator search, where the police receives face descriptions from eye witnesses and tries to generate a realistic image based on those descriptions. This can be further advanced by transforming the generated image using different facial attributes to raise the success rate of the police searching for the suspect (5).

In this paper, we summarize most important developments of GANs in the *Related Work* section. In *Approach and Model Design*, we describe the architecture, parameter choices and techniques we used for our two models, which are then evaluated and compared with each other in *Evaluation* in order to analyze the strengths and weaknesses in our design. Finally, we reflect on our approach in the *Conclusion* section.

2 Related Work

Several architectures for face synthesis have been developed already and remarkable improvements have been made in recent years. Besides the general approach to generative adversarial networks (GAN), we will shortly describe some fundamental network architectures, their characteristics, and some proven techniques to increase the performance and result quality.

Generative Adversarial Networks. In 2014, Ian Goodfellow et al. proposed the first GAN architecture that consists of two distinct networks - a generator and a discriminator. While the generator tries to produce convincing results, the discriminator tries to detect whether an input comes from the original data set or is produced by the generator (6). Song et al. summarize: "the optimization process of GAN models is a mini-max two-player game between the adversaries: where the generator G implicitly defines a probability distribution $p(z)$ and we hope that $p(z)$ converges to the real data distribution p_{data}." (7). To effectively enable this relationship of mutually learning from the other between the two adversaries, a tight balance is necessary. If one is able to overpower the other, the training process will degenerate, often leading to bad results. Other problems concerning GANs are vanishing or exploding gradients, mode collapse, non-convergence of model parameters, and unstable training. Furthermore, the model is highly sensitive to the selected hyperparameters and an unbalanced adversary setting can lead to overfitting (8).

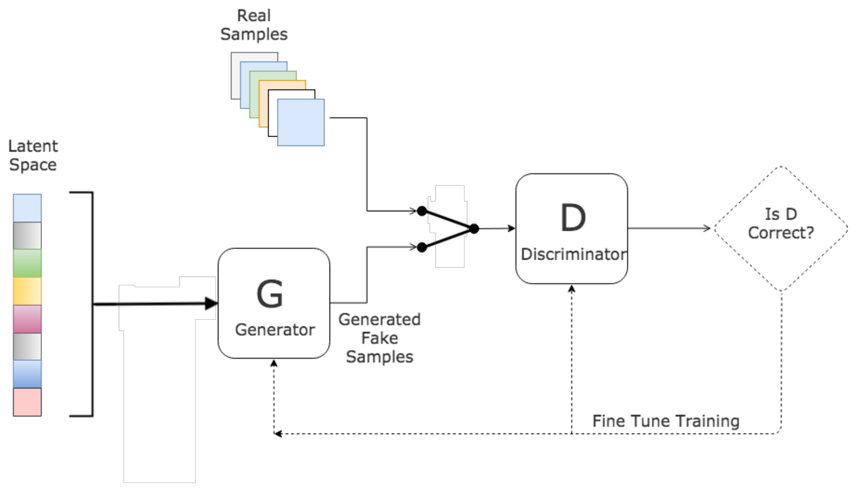


Figure 1: Adversarial Networks Architecture (9)

Conditional GANs. Shortly after Goodfellow's initial publication, Mirza and Osindero introduced conditional GANs (cGANs), which use additional attribute data as an input into both the generator and discriminator. This enables generating more precisely specified data (e.g. images of a specific class) without training multiple nets (10). The attributes can be viewed as an extension to the latent vector space. Another approach to GAN conditioning are auxiliary conditional GANs (AC-GANs), which do not supply attributes to the discriminator. Instead, the discriminator has to classify its input image. Then, a classification loss is computed to condition both adversaries (11).

High Resolution GANs. In the following years, a variety of different architectures emerged, as for example LAGPANs which utilize laplacian image pyramids (12) to enable generation of larger images. Additionally, deep convolutional GANs (DCGANs) were developed by Radford et al., which drastically improved the ability of GANs to generate high-resolution images. Although convolutional layers had been used before, DCGANs use strided convolutional layers instead of deterministic pooling layers, allowing the models to learn their own down- and upsampling functions. Also, Radford et al. have removed any fully-connected layers on top of the convolutional stack. Most state-of-the-art GAN architectures use a DCGAN-inspired approach (13).

Progressive GANs. Highly impressive results were produced with GAN models that use a coarse-to-fine generation strategy, working on different image resolutions in each step. T. Wang et al. used this approach to generate high-resolution realistic images of human faces, street scenes, or room interiors with multiple discriminators for different resolutions (14). In the same year, Karras et al. built upon that and proposed progressive GANs (PGANs), which start training at a low resolution but scale up as the training progresses, adding new layers to process increasingly higher resolutions and fine detail. The effect of this training method is a steadier and faster training, resulting in high resolution images with less noise and artifacts (15).

Further Developments. The problem of mode collapse is one of the main issues when training a GAN. Mode collapse happens when the generator does not produce varied samples. Instead, only a small set of samples that is able to fool the discriminator is repeated again and again. By introducing Wasserstein loss as an alternative loss function, mode collapse could be drastically reduced (16).

Because training and fine tuning a GAN can be difficult and tedious, C. Wang et al. proposed evolutionary GANs in 2018, where the training procedure is treated like an evolutionary problem (17). Another big step forward was the introduction of StyleGAN by Karras et al. in 2018 (4). StyleGAN builds on the progressive GAN architecture and introduces a *mapping layer* for the latent input vector to allow fine-grained control over the resulting image's styles. Karras et al. were able to achieve impressive, photorealistic results. In 2019, Karras et al. improved this architecture even further by removing normalization artifacts (water droplets in generated pictures) (18). Additionally, Sagong et al. introduced a special conditional convolutional layer in 2019 which improves the picture quality and efficiency compared to cGANs especially when generating specific characteristics (19).

3 Approach and Model Design

Our task is to train a generative adversarial network on the CelebA dataset (20). As an additional requirement, our trained model should be able to generate faces based on changeable conditions (e.g. blond hair, eyeglasses, and female) without being trained on subsets of the dataset. Therefore, our solution needs to be a conditional GAN where the generator is also supplied attributes during his forward flow. We implemented two different model architectures: a conditional deep convolutional GAN (cDCGAN) and a conditional progressive growing GAN (cPGAN).

3.1 The CelebA Dataset

The CelebA dataset contains over 200,000 images of celebrities. Additionally, for every image 40 binary attributes are supplied, such as eyeglasses, gender or hair color. The dataset is available in two versions: "In-the-wild" without any preprocessing and one where all images are aligned and cropped to only contain the faces of the celebrities. Clearly, the aligned and cropped version is more suitable for the face generation task. However, there are two drawbacks in using this preprocessed version: firstly, the providers of the dataset chose to crop the images to a 218×178 resolution. As models with convolutional layers generally prefer square training images, this is an unfitting choice for our use case. Secondly, even though many images exist in high resolutions in the "In-the-wild" dataset, the aligned and cropped versions have a slight degree of noise and blur.

Therefore, we used the "In-the-wild" version and employed a custom open-source preprocessing step to align and crop the images (21). This step results in higher quality and square training images; a sample can be seen in Figure 2. With this custom preprocessing step, we are able to specify whether we want to train on 64×64, 128×128, or 256×256 images. This allows a trade-off between the resolution of generated faces and training time.

Figure 2: Sample from the CelebA “In-the-wild” dataset, with our custom preprocessing applied

3.2 Training Flow

A typical GAN training phase is split into two sequentially executed main parts: *discriminator training* and *generator training*. Each part is responsible for training one model while the other one is “frozen”, meaning no backpropagation is performed. These two steps are executed sequentially and therefore define a so-called iteration (similar to an epoch).

The *discriminator* is trained by receiving real data as well as fake data created by the generator. For every image, the discriminator has to predict the probability of the picture being real. Since we know which of the images are real and which ones are fake we can compute the loss, backpropagate, and let the optimizer train the discriminator network. Training the *generator* is very similar. We let the generator create a number of fake images from a random latent space point and then pass these generated images to the discriminator. From the predictions of the discriminator, we calculate the loss for the generator. The loss is further used to train the generator to create better fake data, this is done with iterations. The goal is to generate fake images the discriminator can not recognize as fake.

Since the two nets depend on each other, a successful and efficient training can only be accomplished with well-balanced adversaries. If the system moves towards the direction of a superior-subordinate relation it can no longer be guaranteed that both networks are getting better. If one of the networks flops, the whole system breaks down. Repeating the iteration process over and over will let the adversaries train each other by improving each one separately on the basis of its temporary dominance against the other one and vice versa. Saving interim losses and generated images is crucial for the manual evaluation step, to gain insights on the quality of the training afterwards. It further enables us to detect possible imbalances between the generator and discriminator. By evaluating the training’s metadata we can tweak the system in the right way to prevent these incidents by changing hyperparameters or introducing an uneven training loop. One network can be trained more than the other in order to support its competitiveness towards the more advanced one. Therefore an equilibrium is created again when using an uneven training loop in such situations. Possible model-specific points of change are described in the following sections.

3.3 Conditional Deep Convolutional GAN

The first model we implemented was a deep convolutional GAN (DCGAN) (13), which we conditioned using the attribute information from the CelebA dataset. The model uses multiple sequential convolutional layers for both the generator and the discriminator. Both adversaries are also granted access to attribute information in their forward step to enable conditional face generation. As the CelebA dataset provides 40 conditions, the attribute information is a 40-dimensional vector. The generator further receives a randomly sampled *latent vector* while the discriminator receives an image. The training data is normalized within the range $[-1, 1]$. In the following paragraphs, we dive into details about the specific architecture choices made for both the generator and the discriminator.

Generator. The generator’s task is to transform its input (the latent vector and attribute information) into a square image of the targeted size. The model’s architecture will vary slightly depending on the targeted image size. In order to condition the model, we concatenate the latent vector and attribute vector. Then the created input vector is passed through several sequential fractional-strided convolutional layers. While the first layer transforms the input vector into a 4x4 matrix with a number of channels depending on the targeted image size, the following convolutional layers each double the dimensions of the matrix while halving the number of channels. We use as many convolutional layers as necessary to reach the targeted image size through doubling the matrix dimensions in each layer. In the case of 128x128 images, since we start with a 4x4 matrix from the first layer this would mean

5 additional layers, as $\log_2(128/4) = 5$. The last convolutional layer outputs three channels, which correspond to the number of channels in RGB images.

After every convolutional layer, batch norm layers are added to improve regularization. Then ReLU is applied as an activation function. The last layer uses *Tangens Hyperbolicus* (TanH) as an activation function instead and does not apply a batch norm layer. The TanH activation function was chosen because it maps the input to the value range [-1, 1], which is the same as the training data.

Discriminator. The discriminator's task is to judge whether an image is part of the real training data or fake and generated. First, the attribute vector is passed through a linear layer that maps it to the dimensionality of the targeted image size. This "attribute mapping" is then concatenated to the input image as a fourth channel to achieve conditioning.

The remaining architecture is analogous to the generator's but reversed: While the generator uses fractional-strided convolutional layers to achieve an upsampling effect, the discriminator downsamples using regular strided convolutional layers that halve the image dimensionality, while doubling the number of channels. Batch norm layers are used except for the first and last layer. All layers but the last one use LeakyReLU with a negative slope of 0.2 as an activation function. The last convolutional layer outputs a 1x1 matrix with a single channel and applies the *sigmoid* activation function. The output represents the probability of the input image being real.

	Generator		
	Operation	Activation	Batch Norm
Conditioning	concatenate attribute vector to random latent vector	-	-
First layer	map input to 4x4 matrix with 1024 channels	ReLU	yes
Middle layers (4)	double matrix size; halve number of channels	ReLU	yes
Last layer	double matrix size to 128x128 matrix; output 3 channels (RGB)	TanH	no
	Discriminator		
	Operation	Activation	Batch Norm
Conditioning	expand attribute vector to image size with linear layer	-	-
First layer	halve input matrix size; expand to 64 channels	LeakyReLU	no
Middle layers (4)	halve matrix size; double number of channels	LeakyReLU	yes
Last layer	output scalar value: probability of the input being real	Sigmoid	no

Table 1: Summary of cDCGAN architecture for image size 128x128

Several of the choices, like the use of batch norm layers and activation functions, are inspired by various sources (13; 22; 23; 24) which provide stable implementations with good hyperparameters. However an exhaustive experimentation was impossible due to long training times. In general, the training of state-of-the-art GANs is partly based on heuristics and tricks which have proven themselves over time (22).

The training flow we applied was the basic GAN training flow described in section 3.2 with one modification: the discriminator was trained twice, with one batch containing only real samples while the other batch contained only fake samples from the generator. This separation is a recommended best-practice for GANs (22). The Adam optimizer was used with a learning rate of $2 \cdot 10^{-4}$ and a modified momentum of 0.5, as suggested by Radford et al. (13). As the discriminator's task can be seen as classification (is an image real or not?), we used a binary cross-entropy loss (BCE loss). In our initial training cycles, we were not achieving good results, which we attributed to an overpowering discriminator.

In order to improve results, we implemented two techniques to achieve balance between the two adversaries by increasing the difficulty for the discriminator. First, we employed *one-sided label smoothing* for the discriminator, which is a technique suggested by Goodfellow. One-sided label smoothing punishes the discriminator when it is too confident in its predictions (24). Additionally, we introduced noise to the discriminator through random label flipping as suggested by Chantala (22). By applying this technique, the labels whether an image is real or fake are flipped with a low probability, making the discriminator's task more difficult. However, we then encountered significant mode collapse after 10-15 epochs. We found that disabling the label flipping technique and increasing the size of the latent vector from the initial 100 to 512 helped combat the mode collapse. The increased latent vector size had the strongest effect; we deduce that a larger latent vector allowed our model to capture more variance in the training data. However, we still encountered mild mode collapse after 15 epochs.

It should be noted that the training process with our configuration was unstable, mode collapse and other failure modes were a common occurrence. As future work, Wasserstein loss could help reduce the the occurrence of mode collapse. Nevertheless, through multiple runs of the training process, we were able to achieve a reasonably good model. However, we realized that the cDCGAN architecture would quickly reach its limitations. Motivated by the impressive results and more stable training process achieved by GAN implementations using the coarse-to-fine approach, we set our sights on progressive growing GANs.

3.4 Conditional Progressive Growing GAN

We decided to use a progressive growing GAN, very similar to the one proposed by Karras et al. in 2017 (15), due to the good results and less complex architecture than the latest StyleGANs (18). We additionally conditioned the model on different attributes from the CelebA dataset, such as hair color, age, physiognomy or gender, using an auxiliary classifier GAN (ACGAN) structure.

scaling step (image resolution)	4	8	16	32	64	128
channels in layer	512	512	512	512	256	128
batch size	16	16	16	16	16	8
number of iterations	48000	96000	96000	96000	96000	192000

Table 2: The different scaling steps and their parameters

Both the discriminator and the generator grow synchronously during training, to operate on increasing image resolutions. After the training on one resolution has been finished, a new scaling layer group consisting of an upscaling layer, a convolutional layer and a normalization layer is added to the end of the generator network. Another scaling layer group consisting of a downscaling layer, a convolutional layer and a normalization layer is added to the beginning of the discriminator network. The number of channels for the layers at different scaling steps can be seen in Table 2. To facilitate the progressive growing progress, the model is not trained in epochs but instead a set number of iterations until the next scaling step, which are also listed in Table 2.

For a smooth transition between different scaling steps, new layers are faded in by means of an alpha value that is reduced from 1.0 to 0.0 over the first 20,000 iterations. The alpha value is used as a parameter for linear interpolation between the outputs of the previous and current scaling layers. This process is depicted in Figure 3. When the training in a) has been finished, additional layers are added. The alpha value is used to fade in these new layers by linear interpolation, as shown in b). After a fixed number of iterations, the fading process is completed and the training continues without this interpolation, as seen in c).

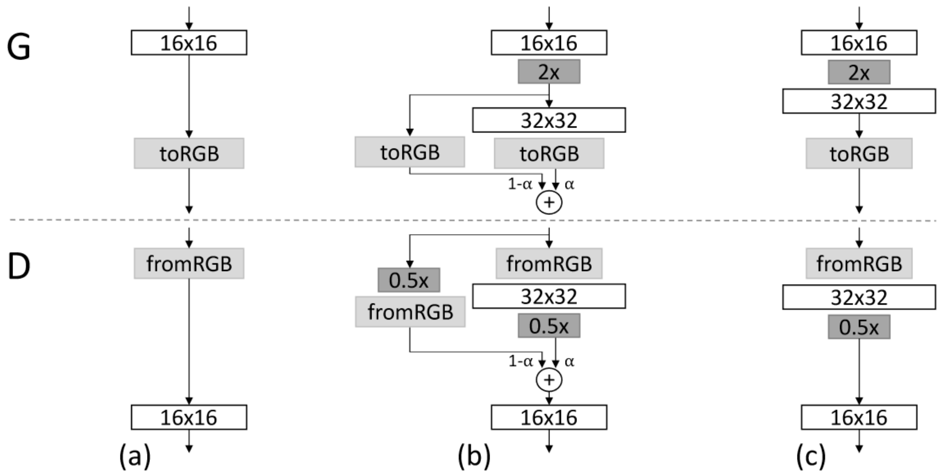


Figure 3: Fading in of new layers (15)

Mode collapse is a common problem when training GANs. To combat this problem we used the Wasserstein loss with gradient penalty and an additional epsilon loss, as proposed by Karras et al. (15). As a consequence, we could not employ the label-smoothing technique we used in the cDCGAN architecture, which uses a binary cross-entropy loss. To prevent the discriminator from getting to dominant we did not train it more often than the generator. To further address the problem of mode collapse, minibatch standard deviation is used.

To introduce structure to the latent space through conditioning, we combined the progressive approach with an auxiliary classifier GAN (11). The generator is supplied with additional information on the attributes of the face it should generate and the discriminator is trained not only to recognize an image as fake, but also on classifying the content. This is achieved by using different layers at the end of the discriminator's training step. A linear decision layer is used for the recognition of fake images and another linear layer for the classification of the content. Then the classification loss is used to condition both adversaries.

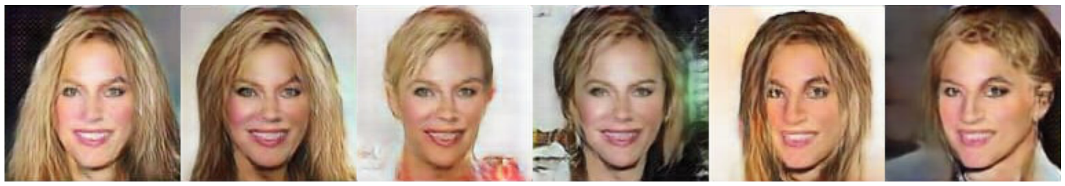
We trained our model based on the preprocessed variant of the CelebA dataset described in Section 3.1 with a resolution of 128×128. Prior to training, the images were prescaled to the different required resolutions in order to increase the training speed. For the hyperparameters, we used a constant learning rate of $1 \cdot 10^{-3}$ and a batch size of 16 until the resolution of 64×64. Because of limitations in our available GPU memory, we then reduced the batch size and increased the number of iterations as can be seen in Table 2. The weight used for the gradient penalty was 10, while Adam was used as the optimizer with betas (beta1: 0, beta2: 0.99), as recommended by Karras et al. (15).

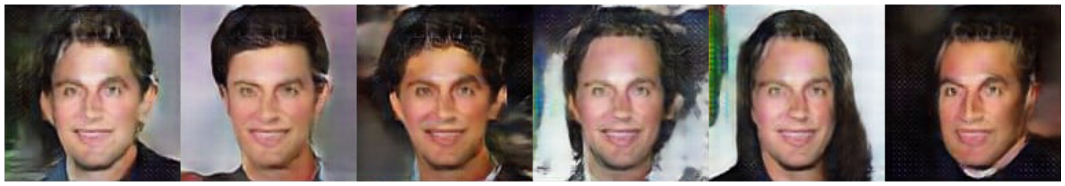
4 Evaluation

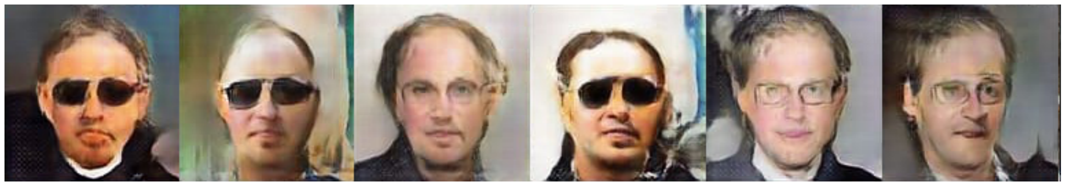
We chose a subjective evaluation approach to assess the results of the cPGAN and the cDCGAN - we evaluate the quality of the results by judging the faces based on our own conception of how a human face should look like. While there are objective evaluation methods (such as the inception score), we decided against using them in order to focus on developing our models.

Online GPU cloud platforms for machine learning and data science, like Google's Colab or Paperspace, provide a relatively good performance for no cost but are unfortunately limited in their usage time. For this reason, we also trained the networks on our own hardware[1], which was able to run without usage restrictions. Training the cPGAN model took over 60 hours and an equivalent of

[1]A Windows PC with an Intel i5-4460 processor (4 cores), 16 GB DDR3 RAM, and an NVIDIA RTX 2080Ti graphics card.


(a) Category: blond and young women


(b) Category: black-haired men


(c) Category: men with glasses

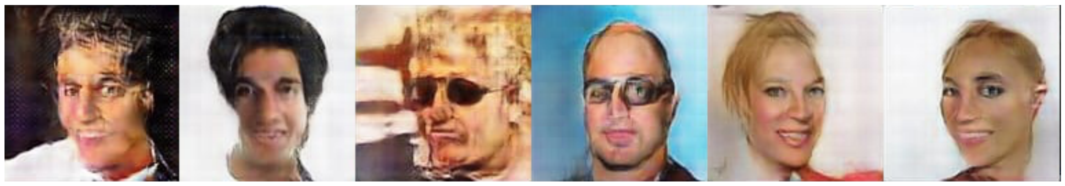
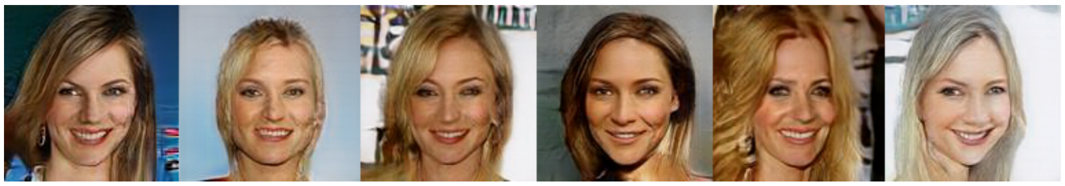

(d) Mediocre results from the categories in figures (a), (b), and (c)

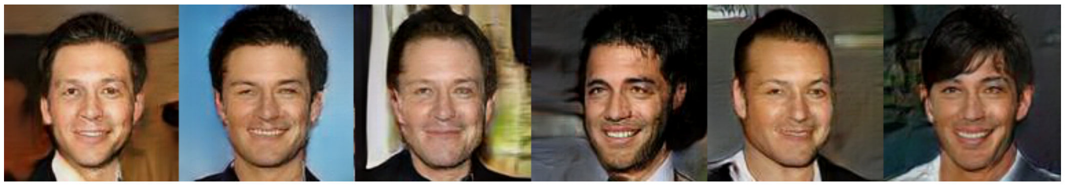
Figure 4: 128×128 images generated with the cDCGAN (selection)

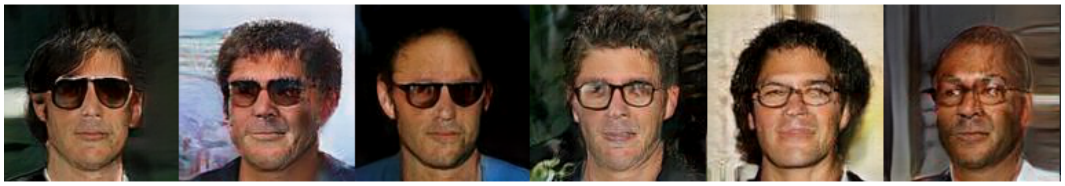
42 epochs to finish the 128×128 resolution step. The cDCGAN was much faster with 2.25 hours for training our current model. While the cPGAN training was limited by our available hardware, the cDCGAN did not appear to improve after being trained for around 18 epochs.

General Quality. As can be seen in Figure 4, the results of the cDCGAN clearly show faces, but very few of them would be mistaken as real. One reason for this is that the foreground and background are often not clearly distinguishable. Also, many of the images show a micro-structure as if they were painted on a canvas or taken with a low resolution camera. Additionally, most of the images are strongly distorted, blurred, or parts of the faces are missing or seem to be merged with the environment, especially the hair and neck region. Finally, a lot of the generated faces look similar. In particular the eyebrows as well as mouths are identical across many samples. After training the model even longer, this effect became stronger, which shows the issue of mode collapse we encountered with this model.

Compared to the cDCGAN, the cPGAN generally produces better results. However, the sample quality varies greatly. There are good results which could be mistaken as real images but mainly the images are mediocre (which can be seen in Figure 5). The bad results are flawed in many ways. Some faces have holes or are strongly distorted. Even in images with a good quality, blurriness and swirls can be detected, especially on the edges of the faces or the hair. Additionally, the people in the images squint extraordinarily often, which is unrealistic. Furthermore, the background of the images is often unconvincing. While most of the pictures would not be mistaken as real, a good variety of facial structures and characteristics can be recognized. The cPGAN generally produces more detailed results, which is especially noticeable in skin and hair texture. For example, while the cDCGAN has


(a) Category: blond and young women


(b) Category: black-haired men


(c) Category: men with glasses

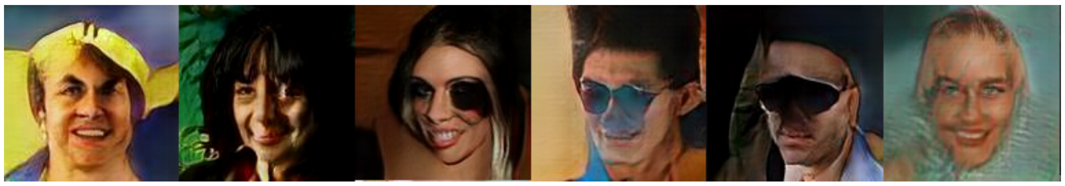

(d) Mediocre results from the categories in figures (a), (b), and (c)

Figure 5: 128×128 images generated with the cPGAN (selection)

been able to learn some amount of texture for blond hair, darker hair tones are often devoid of any texture and merge with the background. Also, results of categories that we suspect might be harder to learn, such as glasses and hats, look much better when generated with the cPGAN. We conclude that these strengths of the cPGAN are in large parts attributable to the employed coarse-to-fine approach.

However, the results achieved by our cPGAN were less impressive than those presented by Karras et al. in their paper on progressive GANs (15). We assume those differences are due to the limited hardware resources and therefore model training time we had at our disposal. As we implemented large parts of the model ourselves, we also cannot rule out implementation-specific deviations that might be partly responsible for the discrepancy. Also, we did not train the model up to the 1024x1024 resolution Karras et al. used due to hardware limitations.

For future work, in order to improve the quality of the results, we could employ the “Truncation Trick” introduced by Brock et al. (25). With this technique the latent space is truncated when generating images, resulting in higher perceived quality at the cost of sample variance. This trade off can then be manually adjusted.

Conditional Generation. Conditional generation is generally functioning for the cDCGAN and the cPGAN model. Individual images might deviate from the specified attributes, but the overall result is satisfying. However, there are combinations of attributes that do not produce satisfying results at all or only in a few cases, for example “bald man with glasses” or “smiling man with hat”. A reason for this might be that this group of people is underrepresented in the training data or that glasses and hats require more complex pixel structures and are therefore harder to learn for the model.

It is interesting to note that the attributes that work best have a “big” visual effect - like hair color, smiling or gender. We assume that these attributes are more obvious and therefore easier to learn.

There is a certain amount of entanglement between the different attributes. For example, it appears that the cDCGAN has learned “Heavy Makeup”, “Wearing Earrings” and “Wearing Necklace” to be proxies for gender, making some of the samples “female-looking” even when we set the gender to male. This effect is less distinctive in the cPGAN, where only “Heavy Makeup” has a similar, albeit weaker, effect.

Objective Evaluation Methods. The adversaries are trained together to maintain an equilibrium; the losses of both adversaries should be balanced without one overpowering the other. Therefore, the losses of the generator and discriminator are not a good indicator for the quality of the generated images. “As an alternative to human annotators”, Salimans et al. (26) propose an automatic method to evaluate the quality of generated images by GAN models objectively which correlates well with the subjective human evaluation - known as the Inception Score. The method captures two properties, the image quality and image diversity, by computing a score between 1.0 and the number of supported classes.

The idea behind this score is that images which are classified strongly in one direction, one class over all others, indicate a high quality. As a result, the overall conditional probability should have a low entropy. The final inception score is computed by summing the Kullback-Leibler (KL) divergence. Therefore the marginal and conditional probability distributions and the relative entropy are combined, over all images. The score is then averaged over all classes before the exponent of the result is taken (26).

In the future, methods like the Inception Score could be used to automatically select the best samples from a generated batch, reducing the need for human selection efforts and improving the perceived quality of the output even though the model itself has stayed the same. We have however paid little attention to the trained discriminator. In the case of the cPGAN with the ACGAN conditioning, we have trained a classifier for facial attributes and image authenticity - we just did not use it for purposes other than training the generator. With high-quality GANs becoming more common place nowadays, good discriminators might be invaluable to differentiate authentic material from fakes. For example, this topic could play an important role in future elections to help slow the spread of fake news in the form of DeepFakes (27). The usage of discriminators should be explored further.

5 Conclusion

We were able to train two separate models that are capable of generating human faces. Both models are also able of conditional generation based on supplied facial attributes without being trained on categorical subsets of the data. The quality of the results is satisfactory, but there is room for improvement. Many of the generated faces are unlikely to pass a “Turing test”, i.e. they would not be able to fool a human. However, the cPGAN in particular was able to produce some very realistic results that are hard to distinguish from the actual CelebA images.

Long training times were a big challenge as the possibility for experimentation was severely limited. Another limitation was our available hardware, as we only had one computer with a GPU at our disposal. Resorting to online platforms, such as Google Colab, partly helped alleviate the problem since we saw a performance decrease by factor six compared to our designated GPU. We frequently hit usage limits and were temporarily suspended from continuing the training process, however. Improvements to the training speed, such as training data formats with higher I/O performance (e.g. HDF5) or better utilization of GPU hardware acceleration, could be highly beneficial to enable more tweaking of the model architecture and hyper-parameters.

In conclusion, we find that generative adversarial networks are an interesting, dynamic, and challenging domain. The rewards are already promising and we expect even further improvements in the coming years in this research field.

References

[1] U. Mutlu and E. Alpaydin, “Training bidirectional generative adversarial networks with hints,” 2020.

[2] K. Slot, P. Kapusta, and J. Kucharski, “Autoencoder-based image processing framework for object appearance modifications,” 2020.

[3] J. Gauthier, “Conditional generative adversarial nets for convolutional face generation,” 2015.

[4] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” 2018.

[5] D. Ma, B. Liu, Z. Kang, J. Zhou, J. Zhu, and Z. Xu, “Two birds with one stone: Transforming and generating facial images with iterative gan,” 2018.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014.

[7] X. Song, M. Shao, W. Zuo, and C. Li, “Face attribute editing based on generative adversarial networks,” 2020.

[8] J. Hui, “Gan - why it is so hard to train generative adversarial networks!.” `http://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b`, June 2018.

[9] A. Gharakhanian, “Gan architecture diagram.” `https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html`, 2017.

[10] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” 2014.

[11] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” *ArXiv*, vol. abs/1610.09585, 2017.

[12] E. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep generative image models using a laplacian pyramid of adversarial networks,” 2015.

[13] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” 2015.

[14] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-resolution image synthesis and semantic manipulation with conditional gans,” 2017.

[15] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *CoRR*, vol. abs/1710.10196, 2017.

[16] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” *arXiv preprint arXiv:1701.07875*, 2017.

[17] C. Wang, C. Xu, X. Yao, and D. Tao, “Evolutionary generative adversarial networks,” 2018.

[18] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” 2019.

[19] M.-C. Sagong, Y.-G. Shin, Y.-J. Yeo, S. Park, and S.-J. Ko, “cgans with conditional convolution layer,” 2019.

[20] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[21] LynnHo, “High-resolution cropping of celeba.” `https://github.com/LynnHo/HD-CelebA-Cropper`, 2020.

[22] S. Chintala, “Gan training tips collection.” `https://github.com/soumith/ganhacks`, 2020.

[23] E. Lindernoren, “Gan implementation collection.” `https://github.com/eriklindernoren/PyTorch-GAN`, 2020.

[24] I. Goodfellow, “Nips 2016 tutorial: Generative adversarial networks,” 12 2016.

[25] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” 2018.

[26] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” 2016.

[27] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, and S. Nahavandi, “Deep learning for deepfakes creation and detection: A survey,” 2019.