Face synthesis with Deep Convolutional Generative Adversarial Networks

Lauri Heikka
University of Oulu
Oulu, Finland
lauri.heikka@student.oulu.fi

Abstract

Generative adversarial networks (GANs) have produced state-of-the-art results in image synthesis, the generation of realistic fake images. I implement and test one seminal GAN-based approach to image synthesis, the deep convolutional generative adversarial network (DCGAN) on a dataset of low-resolution images of celebrity faces. The DCGAN architecture is able to generate realistic images of faces that occasionally fool a human evaluator and score reasonably well on commonly applied performance metrics of synthesized images. Smooth transitions in the generator latent space suggest that the model learns meaningful representations, instead of memorizing training examples and vector arithmetic performed in the latent space reveals that the generator learns a semantically meaningful structure.

1. Introduction

Image synthesis, the generation of realistic fake images has been an active research area in computer vision over the last few years. Image synthesis is an important research area with many applications. One such use case is to generate feature representations that are useful for downstream tasks such as image classification ([19]. Another one is generating synthetic datasets. An important limitation of many deep learning applications is the availability and quality of data. Image synthesis methods can assist in generating entirely new datasets that can be used for downstream learning tasks.

Realistic image synthesis is also an important part of the pipeline in other computer vision tasks. These application areas include translating images from a domain to another, for example translating aerial images to maps ([12, 18, 27], image in-painting [10], upscaling and improving the quality of images ([16, 23]) and generating synthetic images from a text description ([9, 20, 25]).

Remarkable progress was made in the field of image synthesis with the introduction of Generative Adversarial Networks (GANs) [7]. GANs build on the premise of pitting a generative model against a discriminative model that aims to detect whether a sample is a true sample from the data, or created by the generator. Most successful applications of GANs in the field of image generation have been GANS that employ deep convolutional networks ([14, 19, 26]).

A seminal paper in applying a deep convolutional approach was Radford *et al.* [19] who proposed deep convolutional generative adversarial networks (DCGANs). The DCGAN approach consists of a set of improvements for GANs to create stable architectures for image generation. The primary way that DCGANs differ from vanilla GANs with convolutional layers is that any pooling layers are replaced with strided convolutions for the discriminator and fractionally strided convolutions for the generator. There are also no fully connected layers in the network, batch normalization is used between convolutional layers to prevent mode collapse and stabilize training. The maxout activation in the generator, proposed in the original GAN paper ([7]) is replaced with the ReLU activation function.

In this research, my objective is to implement and study the properties of a DCGAN that generates small-resolution images of faces. The training data consists 32x32 images of celebrity faces. I do not do excessive finetuning to optimize model performance for any specific metric. Rather, I iterate on model parameters and architecture until I reach that are subjectively of high enough quality. The rest of the study focuses on performance evaluation and establishing a better understanding of the model. I argue that exploring and testing the capablities of this simple model is a more interesting exercise than exhaustive exploration of hyperparameters or model architectures. My implementation follows quite closely the details described in the original paper with minor adjustments to suit the dataset.

I demonstrate that the DCGAN solves the problem well in multiple ways. First, I evaluate the performance of the DCGAN model on several metrics following prior literature. These metrics include the Fréchet inception distance (FID)[8], the inception score[21], as well as an ad-hoc visual turing test in the spirit of Salimans *et al.* [21], where I myself, and one voluntary participant attempt to distinguish

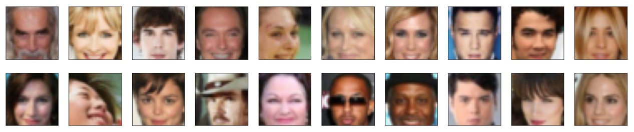
Figure 1. A random sample of images from the training dataset.

between real and generated images.

While, as expected, the trained model does not achieve state-of-the-art level performance, it does perform reasonably well compared to prior literature with respect to all three metrics. The results come with the caveat that the images in our dataset are smaller than commonly used in the literature. This creates difficulties for both the network in learning and human evaluators in distinguishing fake images from real images. Therefore, results are not directly comparable to prior literature.

In addition to these quantitative tests of model performance, I also explore the latent space of the generator. Radford *et al*. [19] demonstrate that simple arithmetic operations and linear interpolations reveal structure and smoothness in the latent space of the generated images. I conduct a similar experiment on our dataset. The results suggest that the generator learns smooth transitions between generated images. Furthermore, I show that vector arithmetic in the latent space can be used to manipulate the semantic content of generated images.

2. Data

I work with a subset of the CelebA dataset[17]. The dataset consists of 89931 images of (mostly western) celebrities. The images have been pre-cropped to remove parts of the images that do not contain faces, and resized to 64x64 images. As specified in the project instructions, I resize the images further down to 32x32 images, which facilitates faster model training while still making it possible to create realistic images of faces. The pixel values are rescaled to range (-1, 1), which corresponds to the range of the tanh output used in the last layer of the generator of our model. Aside from these transformations, no preprocessing is performed on the data aside from transforming the images to Torch tensors for training in PyTorch.

A random draw of examples from the dataset is shown in figure 1. The images mostly consist of caucasian people. Face orientation and positioning, as well as expressions and facial features are varied. The are accessories like glasses and hats visible as well. The images contain only very minimal background and are taken in a controlled lighting. Karras *et al*. [14] note that the CelebA dataset images contain visual artefacts like aliasing, compression and blur. The visibility of these effects are diminished by the fact that the images here are focused on the faces and have a smaller resolution. Evidence of these effects is still visible to a minor extent, however.

3. Methods

3.1. Generative Adversarial Networks

The model I train is a generative adversarial network, introduced by Goodfellow *et al*. [7]. GANs are essentially a minimax game between two networks: the discriminator and the generator. The discriminator's task is to distinguish real images from fake ones output by the generator. The loss that is minimized for training the discriminator resembles the binary cross entropy loss for a binary classifier.

$$\begin{aligned} J^{(D)}(\theta^{(D)}, \theta^{(G)}) = &-\frac{1}{2}\mathbb{E}_{x\sim p_{data}} log(D(x) \\ &-\frac{1}{2}\mathbb{E}_z log(1 - D(G(Z))). \end{aligned} \tag{1}$$

The discriminator is trained on two minibatches every iteration. The first is a minibatch of true examples, labeled with ones. This is represented by the first part of the equation. The second minibatch consists of fake samples generated by the generator and are all labeled with zeros. This is represented by the second part of the equation above.

The generators task is to create realistic fake images, which the discriminator cannot distinguish from real ones. The simplest form of a minimax objective would be to just take the negative of the training loss for the discriminator. In practice, however, generators trained this way suffer from vanishing gradients. Instead, Goodfellow *et al*. [7] propose to maximize the log-probability of the discriminator mistaking the generated samples for real ones. In practice, this means flipping the labels for the generated examples to ones. The generators loss is then:

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_z log(D(G(Z))). \tag{2}$$

Goodfellow *et al*. [7] empirically demonstrate that this GAN architecture, has great potential for image synthesis tasks, when combined with a convolutional architecture.

3.2. DCGAN

The GAN implemented in this study is based on the Deep Convolutional GAN (DCGAN[19]), which significant part of later literature for image synthesis builds upon. The DCGAN architecture consists of a few key insights that improve the performance of GANs with convolutional architectures on image synthesis tasks:

- Using the Adam[15] optimization algorithm
- An all-convolutional architecture with no pooling layers or fully connected layers(following Springenberg *et al*. [24]). Instead of pooling layers, dimensions are

controlled with strided convolutions for the discriminator and fractionally strided convolutions (i.e. transposed convolutions) for the generator with stride > 1.

- Batch normalization[11] between all convolutional layers.
- ReLU activation function for the generator in all layers, except Tanh for the last layer. LeakyReLU activation function for the discriminator in all layers, except sigmoid for the output layer.

While the DCGAN architecture does not represent the current state-of-the-art for image synthesis, it is a seminal approach on which much of the later literature builds upon and an important benchmark to study. Furthermore, much of the later literature focuses on improving performance on high-quality images, while our dataset consists of 32x32 images. The simpler DCGAN architecture is likely well suited for this particular problem.

Establishing a reliable metric for image synthesis tasks is difficult. Due to this, it is not necessarily meaningful to focus on exhaustively tuning the model architecture and hyperparameters. Instead, I iterate on model architecture and hyperparameters based on a qualitative evaluation of a random sample of generated images after each training run and stop after reaching a qualitatively satisfying performance. The different choices that were explored are discussed below together with discussion of the final model architecture.

The convolutional layer architecture follows Radford *et al.* [19]. The discriminator and generator architectures are depicted in figure 2. Every training iteration, the discriminator takes in two sets of inputs, a minibatch of RGB images from the training data, and a minibatch of fake images created by the generator. Both have the shape (batch size x 3 x 32 x 32). The discriminator downsamples the input with strided convolutions to an eventual output of size (batch size x 1 x 1 x 1), which is then reshaped to the column vector (batch size x 1) of outputs.

The generator takes as input a (batch size x 100) noise matrix Z, which consist of noise vectors sampled from a 100-dimensional standard normal distribution. Z is then cast into a 4-dimensional tensor of size (batch size x 100 x 1 x 1). I also experimented with sampling the noise from a uniform distribution, as is done by Radford *et al.* [19]. I found slightly better results when sampling from a normal distribution. The noise is then upsampled by the generator to the eventual minibatch of RGB images with the shape (batch size x 3 x 32 x 32).

I use kernel size 4, stride 2 and padding 1 for the convolutional layers following the official torch implementation by Radford *et al.*[1]. The only exception to the usage of this convolution specification is the use of kernel size 2, stride 1 and no padding in the last convolutional layer for the discriminator and the first transposed convolution for the generator.

I use 128 filters for the first convolutional layer in the discriminator and second to last layer in the generator. As is common for CNN architectures the amount of filters is doubled for each convolution as we move to further layers in the discriminator, increasing to the $2^3 * 128 = 1024$ filters for the second to last layer. The transposed convolutions in the generator follow the same setup in reversed order, starting from 1024 filters and winding down to the eventual 128 filters in the second to last layer. I also experimented with a larger amount of filters but found no noticeable improvement in results, only added computational cost. Less filters resulted in qualitatively worse performance.

Out of interest, I also experimented with two alternative setups. First, I tried one fewer convolutional layers, so that all kernel sizes were 4, with the last convolution having a stride of 1 and 512 channels for the first (last) layer of the generator(discriminator). Second, I tried a kernel size 2 for all convolutional layers with no padding across all layers, based on the intuition that the images I work with are smaller. I observed no improvement in the quality of generated images with these architectures.

I use ReLU activations between the convolutional layers in the generator except for the final layer, for which I use tanh to produce outputs in the pixel range (-1, 1). For the discriminator, I use a LeakyReLU activation between all convolutional layers, except for the final layer, which uses the sigmoid activation to produce probability-like outputs in the range (0, 1). These choices directly follow Radford *et al.* [19].

I use batch normalization[11] between all convolutional layers. Batch normalization scales the input to each unit to zero mean and unit variance. Radford *et al.* [19] argue that batch normalization is critical in stabilizing the learning process and particularly in avoiding mode collapse. Mode collapse, i.e. when the generator starts to produce only one or a few different outputs is a common form of instability with GAN architectures.

3.3. Training details

I train both the discriminator and the generator using the binary cross-entropy loss setup discussed in subsection 3.1. For the final training runs, I train the networks for 100 epochs using minibatch gradient descent. Based on generated images, this might be longer than necessary. However, the runtime still falls short of training times reported for GANs in most of the literature. I use a batch size of 128, although I experimented with smaller batch sizes as well.

[1]Radford *et al.* [19] report a kernel size of 5, which requires force padding the images to correct size. According to one of the co-authors, this was done because of the way transposed convolutions are implemented Theano. See: https://github.com/soumith/dcgan.torch/issues/11

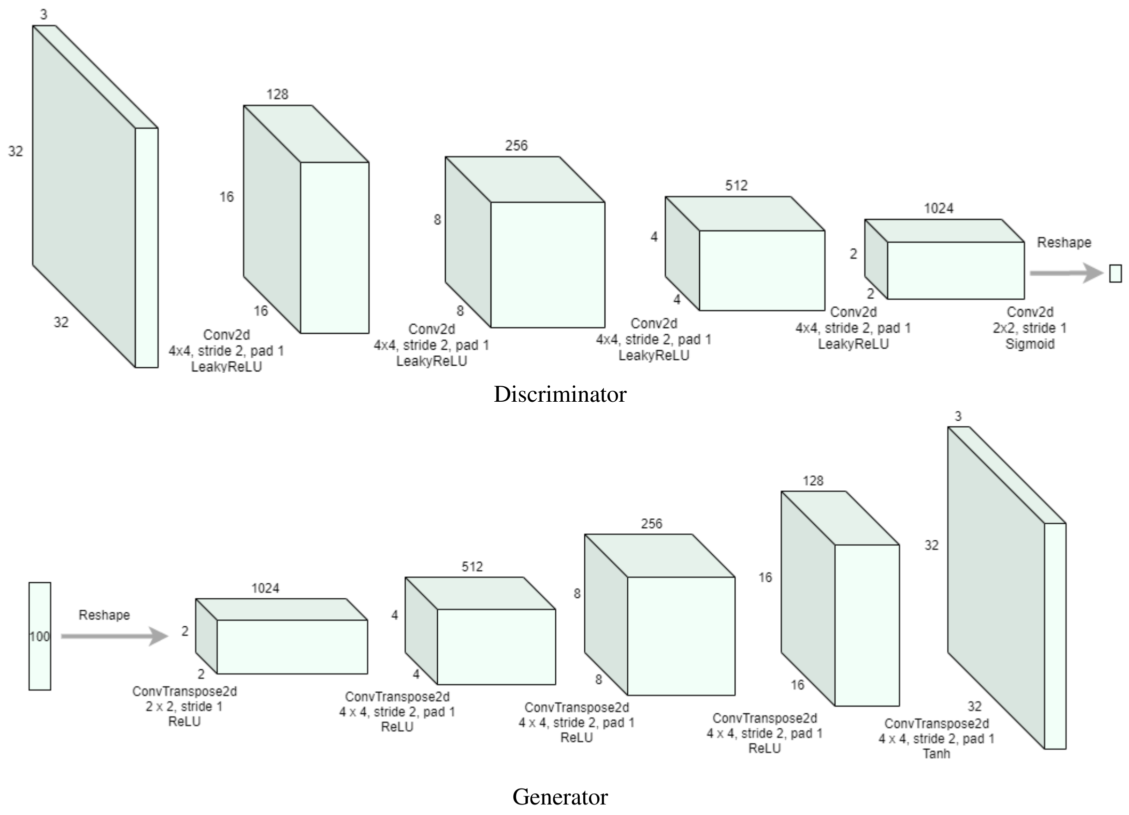

Figure 2. Architectures of the two component networks of our DCGAN model.

More recent papers (e.g. [14, 26]) tend to use use smaller batch sizes with better results. A qualitative comparison revealed no notable improvements in image quality.

For optimization, I use Adam[15]. Adam controls the learning rates for parameters with based on an exponential moving average of the gradient and the squared gradient. The hyperparameters β_1 and β_2 control the decay rates of the moving averages. Kingma & Ba[15] propose $\beta_1 = 0.9$ and $\beta_2 = 0.999$. I find that training is unstable with $\beta_1 = 0.9$. The same problem is reported by Radford *et al*. [19]. I follow their advice and set β_1 at 0.5, and find more stable training performance. The decay rate of the squared gradient average, β_2 is left at 0.999. I set the initial learning rate α to $2e^{-4}$. A more greedily set α of $1e^{-3}$ resulted in unstable training. My implementation mainly utilizes PyTorch and NumPy. I train the networks in a Google Colab environment on an NVIDIA Tesla V100 GPU.

4. Experiments

4.1. Qualitative evaluation of generated images

Figure 3 shows 16 curated examples from a random sample of 100 images created by the generator. Comparing to previous literature is difficult, because the literature mostly focuses on larger resolution images. Images shown in Radford *et al*. [19], trained on a different dataset of faces appear less realistic than the ones generated by this model. As a more recent example, Aneja *et al*. [3] generate images using 64x64 CelebA images. Their images appear more realistic and high fidelity than the ones produced by the DCGAN here.

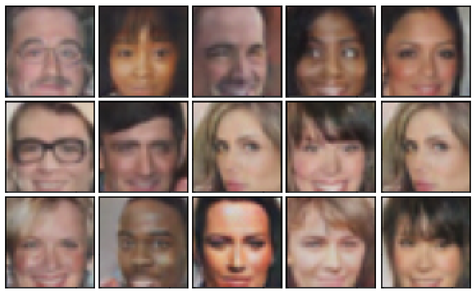

Figure 3. Curated images created by the generator after 100 epochs of training.

The model has difficulties with certain features. Glasses often appear unrealistic, or are very thin and difficult to distinguish. Eyes in general can appear indistinguishable and blurred in generated images. The generator also often has

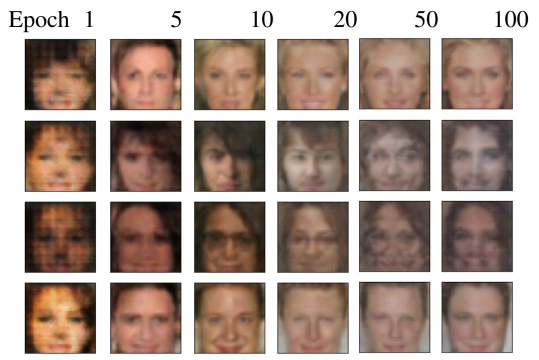

Figure 4. Generated images by epoch, with a static z initiation.

difficulties in generating realistic hair. When generated images contain backgrounds, they appear as blurred and a single color. All of these shortcomings are occasionally notable enough to provide a signal to a human evaluator that the image is fake.

I do not observe mode collapse in any of the many training runs conducted. The generated images are, in general quite varying. A noticeable aspect of the results, however, is that the majority of the generated faces are of caucasian people. This is mainly due to the dataset being dominated by pictures of caucasian people; an unfortunate attribute of the dataset that the generator faithfully learns to represent. This ethnic bias is difficult to resolve with a model, and is likely best addressed by a more balanced dataset (see e.g.[13]).

Figure 4 shows the evolution of images with a fixed random initialization for the noise vector z at different epochs. I see a distinct improvement in training performance until epoch 20. At epoch 50 the generator seems to have had a setback, from which signs are still visible at epoch 100. It is difficult to determine qualitatively, whether the images generated at epoch 100 represent an improvement over the ones generated at epoch 20. Notably, while the images in figure 4 are from one training run, the same setback-effect after a long period of training also occurred in some other training runs. Plotting training losses by iteration revealed that the generator´s loss also becomes noisier as training progresses, suggesting that the generator has difficulties converging. Reconsidering the optimization strategy and for example considering a lower learning rate would likely help the issue. I dont explore this due to the limited time on the project.

4.2. Quantitative evaluation of model performance

4.2.1 Approach

It is difficult to benchmark the performance of generative models. One metric proposed in the literature for evaluating the quality of output images of generative models is the inception score[21]. The inception score uses the conditional label distribution from a pretrained Inception v3 Network to measure the models ability to generate meaningful but varied new images. More precisely, it compares the entropy of the conditional label distribution of each image, $p(y|x)$ to the entropy of the entire distribution of generated images $\int p(y|x = G(z))dz$. The formula for the inception score is

$$InceptionScore = exp(\mathbb{E}_x D_{KL}(p(y|x)||p(y))) \quad (3)$$

where D_{KL} denotes the Kullback–Leibler divergence. Salimans *et al.* [21] report that the score correlates well with human scoring of the realism of generated images. I adapt a PyTorch implementation of the inception score, from[4].

Barrat and Sharma[4] point out that the inception score is not an optimal metric to measure the performance and can be easily manipulated. Furthermore this dataset of celebrity faces is very homogenous compared to the imagenet dataset that the inception model was trained on. Thus, it is not an ideal metric in our setting, because the label distribution for generated images is likely to have very low unconditional entropy by design. A more recent and commonly used metric is the Fréchet inception distance (FID)[8]:

$$FID = ||m - m_w||_2^2 + Tr(C + C_w - 2(CC_w)^{1/2}). \quad (4)$$

where m and C denote the mean and covariance of the activations from the last layer for generator outputs, and m_w, C_w represent the same statistics for training samples. Tr denotes the matrix trace operation. The FID score directly compares the statistics of generated images and training samples. Thus, results are less dependent on what kind of images were used to train the networks. I restructure a Numpy/Keras implementation of the FID score from [5] to fit the PyTorch implementation.

As a third quantitative metric of the capability of the generator to create realistic images, I conduct a small ad-hoc experiment in the spirit of the turing test. The test procedure follows Salimans *et al.* [21], but at a smaller scale: We take a random draw of 50 samples of fake and 50 samples of true images from the training data. Two human participants (the author and a voluntary participant) then individually label these images as real or fake. The accuracy and the rate of fake images mislabeled as real (False Positive Rate, FPR) are recorded for both participants. This is not a scientifically rigorous method of assessing the performance of the generator, but provides some guidance to the quality of the images produced without an excessive use of resources.

4.2.2 Results

Quantitative evaluation results are reported in table 1. For inception score and Fréchet inception distance, the results are similar in magnitude to ones reported in prior literature. I identify one recent paper in the literature (U-Net

Metric	DCGAN	Benchm.	Paper/Dataset
Inception score	2.08	3.43	U-Net GAN[22] CelebA 128x128
FID	9.46	5.25	NCP-VAE[3] CelebA 64x64
Human accuracy	0.745	0.787	Salimans et. al[21] CIFAR-10
Human FPR	0.340	-	

Table 1. Results from quantitative evaluation of the DCGAN performance, together with the most suitable identified comparison in the literature.

GAN[22]) that uses the inception score for another version of the CelebA dataset with larger (128x128) images. The DCGAN model achieves a notably worse inception score of 2.08, compared to the 3.43 of the U-Net GAN. FID for the DCGAN approach here is 9.46. The lowest (lower is better) FID reported for the CelebA 64x64, at PapersWithCode[2], the NCP-VAE[3], is 5.25, which is over nearly twice lower than the DCGAN. The best reported FID for a GAN approach using CelebA 64x64 that I find, is HDCGAN[6], with a more comparable FID of 8.44. It is important to note that the comparisons are reported on datasets of larger images. In identifying comparisons it seems that, in general, higher resolution images tend to result in worse performance in terms of both the inception score and FID.

The results from the human evaluation are more impressive. Across two attempts, both human participants can only achieve an accuracy below 0.8, with an average of 0.745. The only benchmark I could identify was Salimans *et al.* [21], who report a higher average human accuracy of 0.78. However, their results are on the CIFAR-10, which consists of 32x32 images of different types of vehicles and animals, and is potentially an easier dataset for human evaluators.

On average, 34% of fake images are mislabeled as true. The small image resolution makes the task difficult for a human evaluator and real images are mislabeled as fake almost as often as fake images are mislabaled as real. The human prediction accuracy also varies across draws[2]. As this is a small scale experiment, I do not seek to establish statistical significance or make any strong claim about the scientific rigor of the human evaluation. The results, however, provide some anecdotal evidence that the DCGAN approach is capable of generating realistic images of faces.

I also tested selecting a sample of top 1% of generated images based on their respective inception scores, following Salimans *et al.* [21]. There was no improvement in the generated samples compared to random draws in terms of a qualitative evaluation and human accuracy. This result supports the intuition that the label distribution of the Inception model might not be particularly meaningful for our dataset.

5. Visualizations of model characteristics

Finally, I conducted three visualization experiments, to further understand the characteristics of the model. Figure 5 shows images generated along linear interpolation paths between two noise vectors z in the generator latent space[3]. Hair color, skin tone, face orientation and facial expressions transition smoothly along the latent space. The smooth transitions suggest that the generator learns a meaningful representations of faces instead of memorizing training examples.

Radford *et al.* [19] demonstrate a semantic structure in the latent space of their generator, that allows for the semantic content of generated images to be manipulated by vector arithmetic in the latent space. Figure 6 illustrates two of these vector arithmetic experiments performed on our generator. For the first one, I identify twelve latent space vectors: four for which the generator produces a man with no hair, four that produce a man with hair and four that produce a woman with hair. I produce an aggregate neutral man vector and neutral woman vector by taking the average of the four vectors. I then deduct the aggregate neutral man vector and add the aggregate neutral woman vector on each of the vectors that produce a man with no hair. The outcome is four noise vectors that each produce an image of a woman with no hair. Each produced image also has similar face positioning and facial features than the starting image.

In the second vector arithmetic exercise, I produce noise vectors corresponding to smiling men out of vectors that produce smiling women with a similar procedure. The results from the vector arithmetic suggest that the latent space learned by the generator could be useful in downstream applications and conditional generative modeling. They also further support the notion, that the generator learns a semantically meaningful representation of the images.

To be complete in my examination, I also experimented with guided backpropagation visualization of the discriminator activations. The implementation is adapted from [1]. Four examples of guided backprop activations are shown in figure 7. The activations show that different filters clearly activate on distinct parts of faces such as eyes and nose. However, I did not find the results to be very interpretable or interesting in general.

[2]Accuracies achieved across two draws of 100 images for the two participants were 0.72, 0.77, 0.76, 0.73.

[3]A more correct method, would be to perform spherical interpolation rather than linear interpolation in order to avoid the interpolation points being outside of the expected distribution of the latent space. The topic is discussed here: https://github.com/soumith/dcgan.torch/issues/14. However, l interpolation generates smooth enough transitions in the latent space in this scenario in our setting, so I retain the simpler implementation.

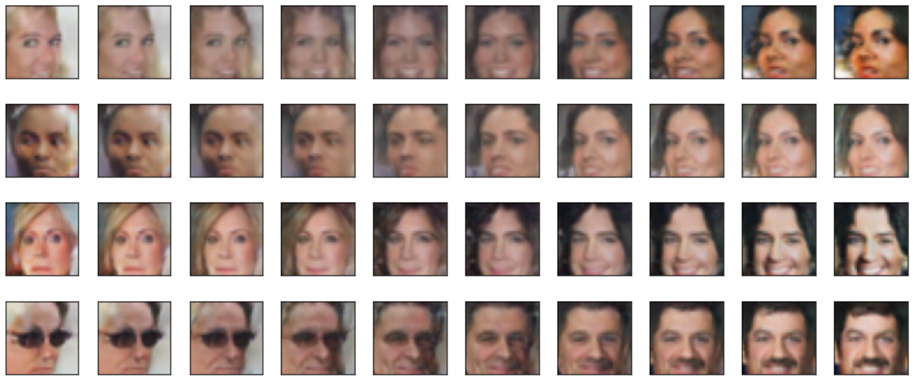

Figure 5. Generated images along a linear interpolation path in the latent space between different images.

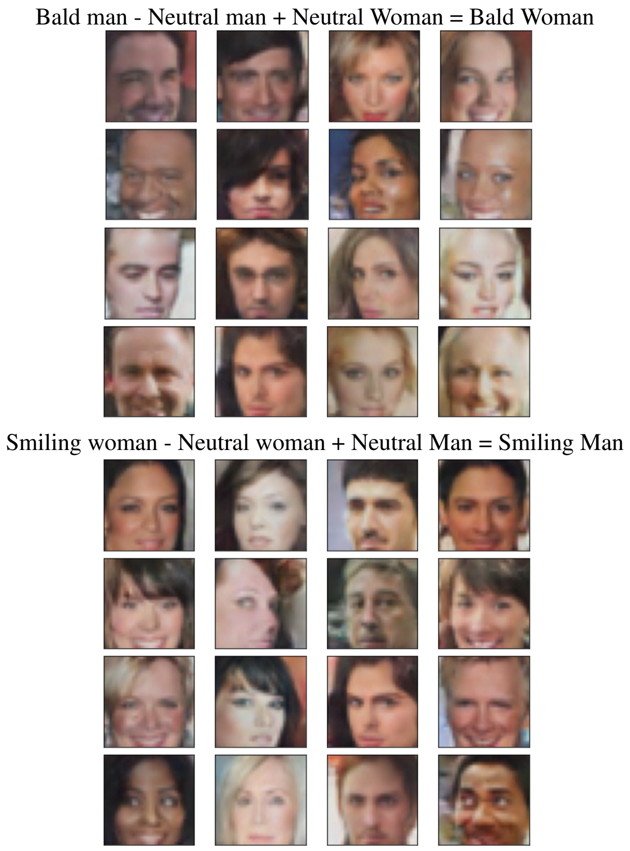
Bald man - Neutral man + Neutral Woman = Bald Woman
Smiling woman - Neutral woman + Neutral Man = Smiling Man

Figure 6. Two examples of vector arithmetic in the latent space.

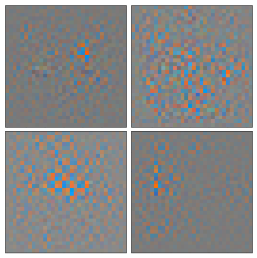

Figure 7. Examples of guided backpropagation filters

6. Conclusion

In this work I have implement and tested a DCGAN approach for generating synthetic images of faces on a dataset of small-resolution images of celebrity faces. I demonstrate that the DCGAN approach and the architectural choices made solve the problem reasonably well. Generated images are realistic enough to occasionally cause an untrained evaluator to mistake them for real images. The images also score reasonably in terms of the inception score and the Fréchet inception distance, although the results fall significantly short of the current state-of-the-art. These metrics are not reported in the original DCGAN paper. Exploration of the latent space uncovers evidence suggesting that the generator learns a semantically meaningful representational structure instead of memorizing training examples.

Although I report quantitative metrics of model performance, I do not do extensive exploration of tuning model performance. This would be an interesting area to pursue, to see how good of a performance the DCGAN approach can reach. Furthermore, the literature on GANs has increased exponentially over the last few years, and new state-of-the-art has been uncovered many times since the DCGAN architecture was proposed. Considering e.g. CelebA dataset versions used in prior literature, or testing more novel architectures on our dataset would be another intuitive avenue to pursue.

The dataset used here suffers from racial bias, resulting in racial bias in the generator. Reconducting the experiments on a more balanced datasets is an intuitive option. Approaches that seek to improve further the heterogeneity of images, for instance, thourgh objective function specification would be an interesting area to further pursue. However, I am not familiar enough with the quickly expanding

GAN literature to provide more than speculation on the subject.

This project is above all a learning exercise. Although, the implementation was heavily guided, the project provided some valuable experience in executing a research project in machine learning on image data, an area that I am not very familiar with. To name some highlights, implementing and experimenting on one provided a much more comprehensive understanding than I had earlier. Experimenting with the convolutional architecture also provided a nice learning opportunity. For example, I got a better intuitive feeling on how (strided) convolutions manipulate output dimensios beyond just applying a formula. Reviewing prior literature, implementing the experiments and writing the report also added to this experience.

References

[1] Grad-cam with pytorch on github. `https://github.com/kazuto1011/grad-cam-pytorch`. Accessed: 2020-12-15.

[2] Papers with code, image generation benchmarks. `https://paperswithcode.com/task/image-generation`. Accessed: 2020-12-15.

[3] J. Aneja, A. Schwing, J. Kautz, and A. Vahdat. Ncp-vae: Variational autoencoders with noise contrastive priors. *arXiv preprint arXiv:2010.02917*, 2020.

[4] S. Barratt and R. Sharma. A note on the inception score, 2018.

[5] J. Brownlee. How to implement the frechet inception distance (fid) for evaluating gans, Aug 2019.

[6] J. D. Curtó, I. C. Zarza, F. de la Torre, I. King, and M. R. Lyu. High-resolution deep convolutional generative adversarial networks, 2020.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30:6626–6637, 2017.

[9] S. Hong, D. Yang, J. Choi, and H. Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7986–7994, 2018.

[10] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[13] K. Kärkkäinen and J. Joo. Fairface: Face attribute dataset for balanced race, gender, and age. *CoRR*, abs/1908.04913, 2019.

[14] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[17] 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.

[18] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[19] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[20] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[21] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016.

[22] E. Schonfeld, B. Schiele, and A. Khoreva. A u-net based discriminator for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8207–8216, 2020.

[23] C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

[24] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[25] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[26] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018.

[27] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.