Project 3
Report

Generate Faces with Specific Attributes

Students:

Simone Gayed Said

Pierpasquale Colagrande

Teacher:

Andrea Asperti

Tutor:

Erika Gardini

Course:

Deep Learning

Contents

Objectives

The aim of this project is to generate new face images similar to training ones (the CelebA dataset) according to user specified attributes (e.g. with eyeglasses, blond hair etc.).

We have tried to experiment different approaches in order to find the best possible solution with reference to the resources available to us.

The quality of generated images has been measured using the requested metric, Frechet Inception Distance (FID). We have also proposed possible metrics to assess conditional generation.

Tools

Several tools have been used for the neural network development and training procedure. We decided to use free tools or, at most, products which gave free credits to students.

Git and GitHub

The entire application was developed using git and GitHub. A GitHub organization has been created in order to open a shared repository for the project and to work on it.

Jupyter Notebook

Since developing a neural network implies doing research and trying different approaches, we decided to use Jupyter Notebook, which allows us to edit different parts of the code and trying different approaches without having to re-run the entire script.

Colab

While creating a neural network, the training procedure can be very time consuming. Therefore, using one of our personal machines to train the network was not a clever option. We then evaluated different online services, like AWS, Google Cloud, Colab and so on. We evaluated this services on different factors like price, delivered computational power and integration with other technologies. We then decided for the system which offered the most with less compromises.

The list that follows resumes our evaluations:

AWS

Students can have 100$ of free credit to use on AWS, however this amount of money is enough to get a standard virtual machine. We could purchase a virtual machine with a GPU but this was not the best choice compared to other solutions.
AWS also includes Amazon SageMaker, a framework designed to develop, train and distribute deep learning systems, however, we thought that this was a little overkilled for our needs, considering the prices.

Microsoft Azure

Like AWS, Azure gives 100$ of free credit to students. Azure also has a framework to develop, train and distribute deep learning systems too. Same evaluations we did for AWS were done for Azure.

Google Cloud

Google Cloud is different from AWS and Azure. It gives 300$ of free credit for anyone who decides to have a trial period. Google Cloud also allows to purchase a VM with a TPU instead of a GPU, which is a device that is like a GPU, but it is designed specifically to work with Tensorflow. It does not have an integrated deep learning framework but, as we said before, it was unnecessary. Sadly, this amount of free credit was enough to buy a medium powered virtual machine with a GPU. TPUs were expensive and so out of our buying power.

Kaggle

Kaggle, a subsidiary of Google, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment. It has many datasets you can easily import among which there is also the CelebA one. Kaggle Kernels also offer the possibility to use hardware accelerator such as GPUs and TPUs but for a limited amount of time (30 hours per week). The drawbacks are that Kaggle Kernels seem to be a little laggy and in general less powerful compared to Colab ones, moreover it does not provide any integration with GitHub.

Colab

Colab is a free tool from Google that can be used for research purposes. It gives enough training power to train a network and it is well integrated with GitHub. It also has the possibility of using TPUs instead of GPUs, completely for free. Unfortunately, Google Colab has limited storing capacity, so it must be used, for big datasets, together with Google Drive. Colab also does not maintain in memory the files uploaded, so not using Google Drive means reuploading the dataset manually at each run.

After we evaluated pros and cons of each system, we decided to use Colab to develop our deep learning program.

Microsoft OneDrive

Finally, we used OneDrive to store this report as it permits to share the document between different users and to edit it at the same time together.

Recent techniques

Standard techniques

Conditional VAE

Variational autoencoders (VAEs) are generative models which make use of deep neural networks to describe the distribution of observed and latent (unobserved) variables. They can be used for generative purposes (e.g. generating faces). VAEs are composed of an encoder, which learns the distribution of observed variables, and a decoder, which generate images using the learnt distribution.

Standard VAEs, however, cannot be used to generate conditioned images, so, conditional VAEs have been introduced, which allow to control the generation of images.

Conditional VAE are autoencoder architectures that condition the generated data using another description of the data, y. In these models, we can generate samples from the conditional distribution p(x|y) by changing the value of y.

Conditional GAN

Generative Adversarial Networks, or GANs, are deep neural networks use for training generative models, for example to generate faces. GANs are composed by a generator, which learns to generate images starting from noise, and a discriminator, which learns to distinguish between fake images (generated by the generator) and real images. GANs are based on a min-max game between the generator and the discriminator, in which the generator learns to fool the discriminator in order to produce realistic images.

However, there is no way to condition images that are generated, using standard GANs.

To do this, we can use conditional generative adversarial networks, or cGANs for short, which are a type of GAN that allow to condition the generated images. Image generation

can be conditional on a class label, if available, allowing to generate images of a given type.

State-of-the-art techniques

MHingeGAN: cGANs with Multi-Hinge Loss

This work uses a new algorithm to incorporate class conditional information into the discriminator of GANs via a multi-class generalization of the commonly used Hinge loss. This approach contrasts most GAN frameworks in which a single classifier is trained for K+1 class with one loss function, instead of a real/fake discriminator, or a discriminator classifier pair. MHingeGAN can perform well in both fully supervised and semi-supervised settings and learns an accurate classifier concurrently with a high-quality generator [1].

LOGAN: Latent Optimisation for Generative Adversarial Networks

This work improves CS-GAN (Cyclic Synthesized GANs) with natural gradient-based latent optimization and shows that it improves adversarial dynamics by enhancing interactions between the discriminator and the generator. Experiments demonstrate that latent optimization can significantly improve GAN training, obtaining state-of-the-art performance [2].

FQ-GAN: Feature Quantization Improves GAN Training

In this work, it is proposed Feature Quantization (FQ) for the discriminator, to embed both true and fake data samples into a shared discrete space. The quantized values of FQ are constructed as an evolving dictionary, which is consistent with feature statistics of the recent distribution history. Hence, FQ implicitly enables robust feature matching in a compact space. This method can be easily plugged into existing GAN models, with little computational overhead in training. Extensive experimental results show that the proposed FQ-GAN can improve the FID scores of baseline methods by a large margin on a variety of tasks, achieving new state-of-the-art performance [3].

Our model

Tested approaches

To obtain the best possible results, we tested several different models and techniques and then we ended up using GANs in combination with a Versatile Auxiliary Classifier. Among all the different approaches we used, we found out that, for us, this was the

technique producing the best results, considering the image quality, the FID score, and its capacity to deal with conditioning. We started experimenting on basic GANs network, trying to improve basic architectures, and experimenting on them. Then we found an interesting model of a VAC-GAN described by *Sophie Searcy* in one of her blog posts [4] and we tried to improve the results by adding to it the improvements we explored in the previous GANs we worked on.

Versatile Auxiliary Classifier with Generative Adversarial Network

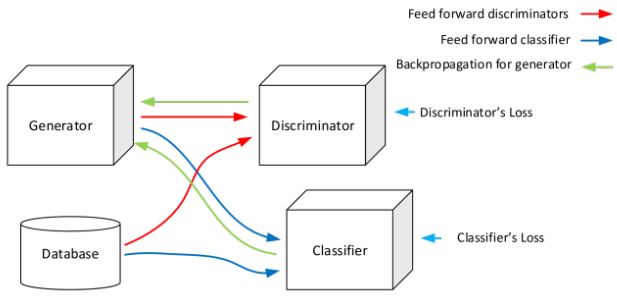

Figure 1. Model for training conditional deep generators.

In this kind of approach, the Generative Adversarial Network generator is turned into a conditional generator by placing a multi-class classifier in parallel with the discriminator network and backpropagate the classification error through the generator. This technique is versatile enough to be applied to any GAN [5].

Architecture

Generator

The primary goal of the generator is to generate RGB images of dimension 64x64. It takes a 100-dimensional latent vector and some extra information, the facial attributes, and tries to generate realistic images. The generator network is a deep convolutional neural network. It is made up of dense, upsampling, and convolutional layers. The model takes as input the concatenation of a noise vector (the latent vector) and a conditioning vector. The conditioning vector is the additional information provided to the network. In our case, of course, it is the list of 40 attributes provided by the CelebA dataset.

Discriminator

The primary goal of the discriminator network is to identify whether the provided image is fake or real. It does this by passing the image through a series of downsampling layers and some classification layers. In other words, it predicts whether the image is real or fake. Like the generator network, the discriminator network is a deep convolutional network that contains several convolutional blocks. Each of them is composed of a convolutional layer, and an activation layer.

Classifier

The primary goal of the classifier network is the multi-label classification. The classifier learns to classify images extracting labels from them. It does this by passing the image through a series of downsampling layers and some classification layers. Like the other networks, the classifier network is another deep convolutional network that contains several convolutional blocks. Each of them is composed of a convolutional layer, a batch normalization layer and an activation layer.

Techniques used to improve the results

TTUR

Two Time-Scale Update Rule, or TTUR, consists in choosing different learning rates for the generator and discriminator. In the paper where TTUR was first introduced, the authors provided a mathematical proof of convergence to Nash equilibrium and showed that implementing famous GANs (DCGAN, WGAN-GP) using different learning rates, for the discriminator and the generator, achieved state-of-the-art results. Generally, it is recommended to choose a higher learning rate for the discriminator and a lower one for the generator. In this way, the generator must make smaller steps to fool the discriminator and so it does not choose fast, unprecise and unrealistic solutions just to win the adversarial game [6].

Spectral Normalization

Spectral Normalization is a weight normalization that stabilizes the training of a model. It controls the Lipschitz constant to mitigate the exploding gradient problem and the mode collapse problem. Conceptually, it restricts the weight changes in each iteration, allowing the discriminator to not over depend on a small set of features in distinguishing images. This approach has proven to be computationally light compared to WGANs and achieve good mode coverage [7]. We have implemented it in our model by the definition a new Conv2D Layer that extends the classical Conv2D Layer of Keras by the introduction of a Spectral Normalization parameter that if set to True enables the weight normalization.

Balance between discriminator and generator

The discriminator and generator are always competing to undercut each other. Mode collapse is often explained as an imbalance between the discriminator and the generator. To make the training more stable we decided to do two training iterations on the discriminator for each training iteration on the generator.

Label smoothing

With GANs, overconfidence is an issue. To avoid this problem, we penalize the discriminator by setting our target label value to 0.9 instead of 1.0 when an attribute is present. This trick helped a lot in decreasing the FID score. In fact, this technique reduced the score of about 10 points.

Results

Generated Images

Figure2. Top-Left: Young = 1, Attractive = 1, Black_Hair = 1, Smiling = 1, Heavy_Makeup = 1, No_Bread = 1
Top-Right: Young = 1, Attractive = 1, Brown_Hair = 1, Smiling = 1, Heavy_Makeup = 1, No_Bread = 1
Bottom-Left: Young = 1, Attractive = 1, Blond_Hair = 1, Smiling = 1, Heavy_Makeup = 1, No_Bread = 1
Bottom-Right: Young = 1, Attractive = 1, Brown_Hair = 1, Smiling = 1, Heavy_Makeup = 1, No_Bread = 1, Bans = 1

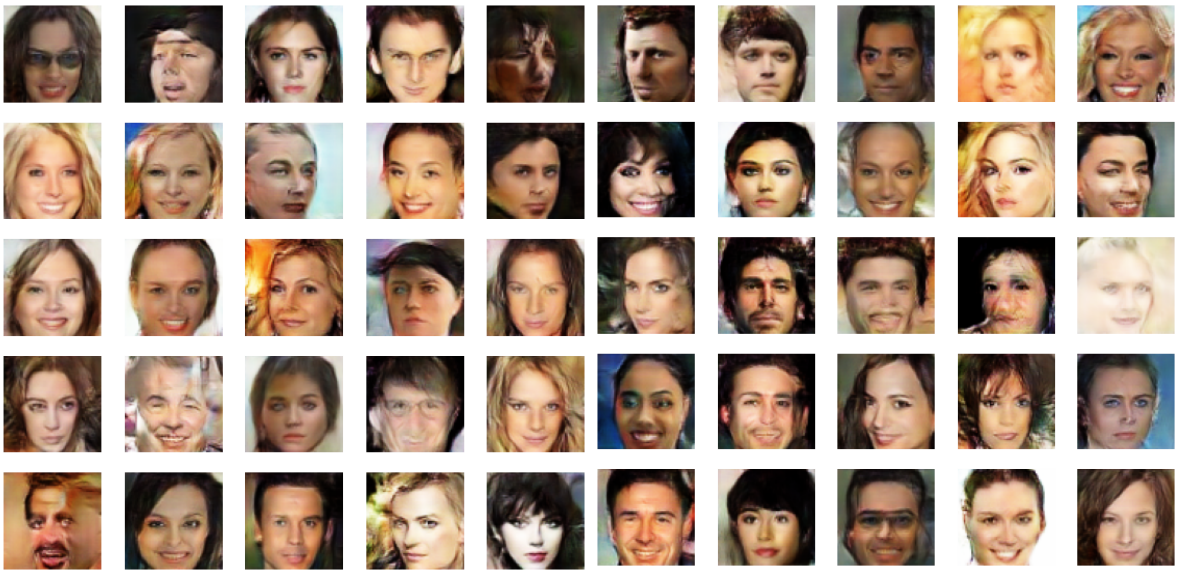

Figure 3. Images generated from random labels of the dataset

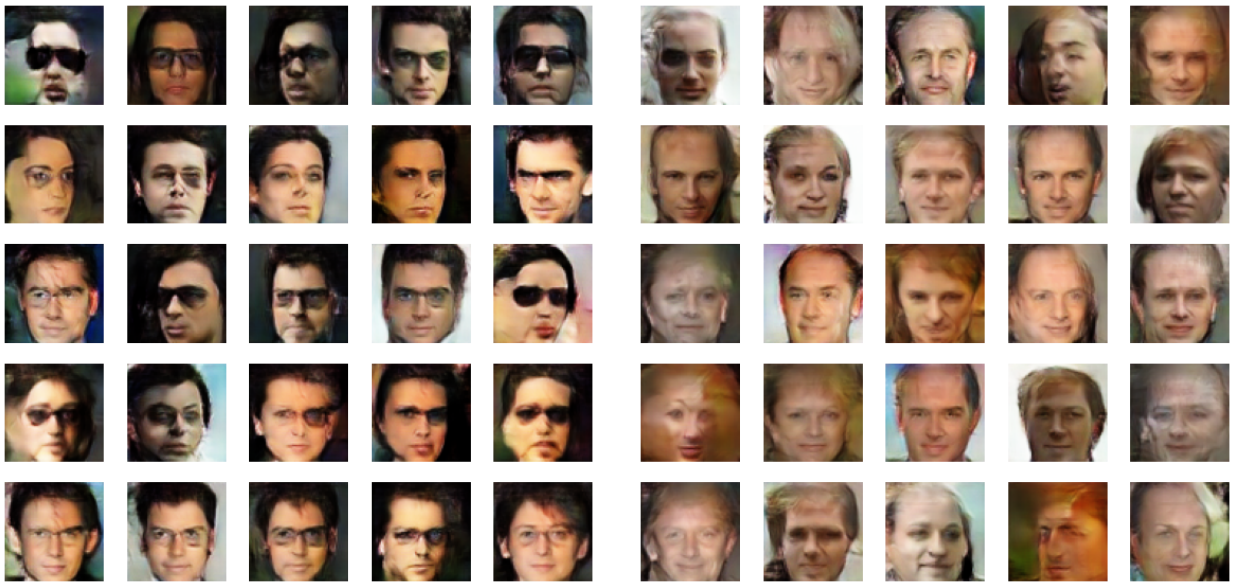

Figure 4. Left: Male = 1, Young = 1, Attractive = 1, Black_Hair = 1, Eyeglasses = 1, No_Bread = 1
Right: Male = 1, Attractive = 1, Bald = 1, Big_Nose = 1, No_Bread = 1

FID

The Frechet Inception Distance (FID) score is a metric that calculates the distance between feature vectors calculated for real and generated images.

The score summarizes how similar the two groups are in terms of statistics on computer vision features of the raw images calculated using the inception v3 model used for image classification. Lower scores indicate the two groups of images are more similar, or have more similar statistics, with a perfect score being 0.0 indicating that the two groups of images are identical.

The FID score is used to evaluate the quality of images generated by GANs. Low scores are correlated with higher quality images [8].

The best fid score we achieved, calculated over a set of 10000 images, is 27.4. The labels used to generate the conditioned images were taken directly from the dataset labels in order to generate random images that looked like the original dataset images. We chose this approach because generating random labels will also mean generating possible paradoxes (e.g. a blonde bald, a woman with mustache).

Proposed Metric

Unconditional image generation models have seen rapid improvement both in terms of generation quality and diversity. These generative models are successful, if the generated images are indistinguishable from the real images that are sampled from the training distribution. This property can be evaluated in many ways: the most popular are the Inception Score (IS), which considers the output of a pretrained classifier, and the Fréchet Inception Distance (FID), which measures the distance between the distribution of extracted features of the real and the generated data.

While unconditional generative models take as input a random vector, conditional generation allows to control the class or other properties of the synthesized image.

In order to provide valuable metrics to evaluate and compare conditional models, we propose two metrics, presented by *Yaniv Benny, Tomer Galanti, Sagie Benaim, Lior Wolf* in their paper *Evaluation Metrics for Conditional Image Generation* [9], called Conditional Inception Score (CIS) and Conditional Fréchet Inception Distance (CFID). This two metrics contain two components each:

- the within-class component (WCIS/WCFID), which measures the quality and diversity for each of the conditional classes in the generated data. In other words, it measures the ability to replicate the distribution of each class in the true samples
- the between-class component (BCIS/BCFID), which measures how close the representation of classes in the generated distribution is to the representation in the real data distribution

Conditional Inception Score (CIS)

The conditional analysis of the Inception Score addresses both aspects of conditional generation: the need to create realistic and diverse images and the need to have each generated image match its condition. To do that two scores are defined: the between-class (BCIS) and the within-class (WCIS).

BCIS evaluates the IS on the class averages. It is a measurement of the mutual information between the conditioned classes and the real classes. The prediction probabilities for all the samples in each conditioned class are averaged to produce the average prediction probability of the entire class, then the IS is computed on these averages.

WCIS evaluates the IS within each category. It is a measurement of the mutual information between the real classes conditioned on the samples and the real classes conditioned on the conditioned classes. The final score is the geometric average score over all the classes, which is equivalent to the exponent on the arithmetic average of the mutual information over all the classes.

In general, the best thing would be to have the BCIS as high as possible and the WCIS as low as possible. High BCIS indicates a distinct class representation for each conditioned class and a wide coverage across the conditioned classes, which is a desired property. High WCIS indicates a wide coverage of real classes within the conditioned classes, which is an undesired property since each conditioned class should represent only a single real class. In this way, one obtains consistent prediction within each class and has high variability between classes.

Conditional Fréchet Inception Distance (CFID)

For conditional FID, we want to measure the distance between different distributions, according to the feature vector $f(x)$, produced by the pre-trained feature extractor f on sample x.

Analogous to the conditional IS metrics, it is measured the between-class distance between averages of conditioned class features and averages of real class features, as well as the average within-class distance for each matching pair of real and conditioned classes.

BCFID measures the FID between the distribution of the average feature vector of conditioned classes in the generated data and the distribution of the average feature vector of real classes in the class real data. It evaluates the coverage of the conditioned classes over the real classes.

WCFID measures the FID between the distribution of the generated data and the real data within each one of the classes. It evaluates how similar each conditioned class is to its respective real class. The total score is the mean FID within the classes.

Conclusions

Working improvements

After all the experiments we did, we noticed that some techniques adopted led to improvements in the quality of images, in the FID score or in the conditionality of the network.

First, we noticed that using upsampling layers in combination of a convolutional layer (for each upsampling layer) in the generator, instead of deconvolution layers, produced images with a higher quality: we had smoother images, while decovolution layers produced more pixeled images.

The use of dropout inside the generator helped us dealing with overfitting, so it also improved the FID because it helped generating images different from the dataset images. Also, dropout helped the network in avoiding mode collapse.

As said in "Techniques used to improve the results" section, label smoothing technique improved FID reducing it significantly.

The type of GAN adopted improved consistently the conditionality of the network in comparison with a standard CGAN.

A key point in improving GANs is balancing both the discriminator and the generator. By improving this aspect of the network, we can obtain a much higher training stability, avoiding mode collapse, as well as better results in general. To balance this aspect, we used the techniques listed in "Techniques used to improve the results" section.

Limitations

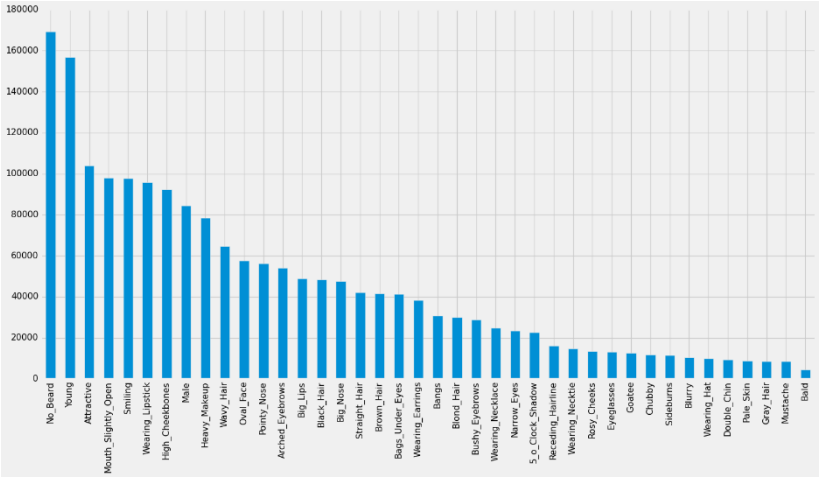

Figure 5. Number of occurrences of each attribute in the CelabA dataset

We also noticed however that our network does not perform very well in certain cases.

For example, some attributes are not contributing excellently to the conditioning of the generated images. We think that this could be an issue related to a lower number of occurrences of these attributes in the dataset. This means that some combinations of attributes are conditioning the images much better than other combination of attributes. For example, we noticed that attributes like “Goatee” or “Mustache” are not conditioning that much and the quality of the images decreases a lot.

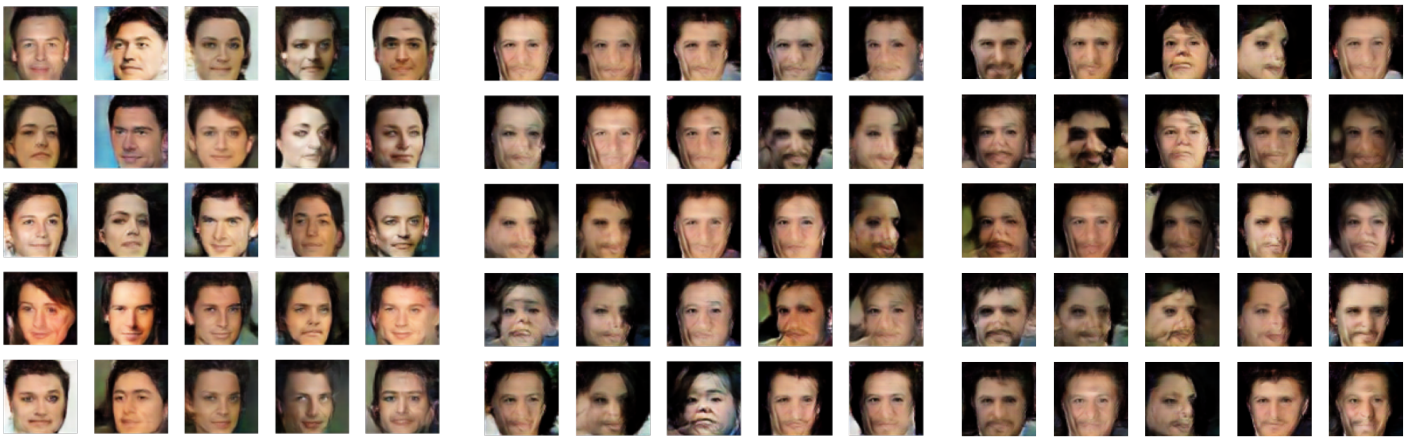

Figure 6. (From left to right): Male = 1, Young = 1, Attractive = 1, Black_Hair = 1, Smiling = 1, No_Beard = 1
Male = 1, Young = 1, Attractive = 1, Black_Hair = 1, Smiling = 1, Goatee = 1
Male = 1, Young = 1, Attractive = 1, Black_Hair = 1, Smiling = 1, Mustache = 1

Also, in order to have better images, we need a certain number of attributes. For example, using just “Blond_Hair” as attribute produces worse results than using a combination of “Blond_Hair”, “Young”, “Attractive”, “No_Beard” and “Smiling”. We noticed that the network is still able to condition using just “Blonde”, but the images have a lower quality in comparison to the images generated using the combination of more attributes. This could be an overfitting issue related to the fact that the network is

learning better the distribution of combinations of attributes instead of learning the distribution of single attributes.

Figure 7.Left: Blond_Hair = 1
Right: Blond_Hair = 1, Young = 1, Attractive = 1, No_Beard = 1, Smiling = 1

Future developments

We notice that, working with GANs, training the network for longer times was reducing the quality of the generated images. This could be avoided by introducing an early stopping mechanism to stop the training when the network reaches the best results.

We also think that introducing the improvements we listed before to a VAE-GAN could be a step forward. In fact, use a VAE-GAN could be a solution to the "overfitting" problem on labels we described before, while all the before mentioned techniques will help the VAE-GAN to be stable while it trains.

References

[1] Ilya Kavalerov, Wojciech Czaja, Rama Chellappa. cGANs with Multi-Hinge Loss. arXiv:1912.04216 [cs.LG],], Dec. 2019

[2] Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, Timothy Lillicrap. LOGAN: Latent Optimisation for Generative Adversarial Networks. arXiv:1912.00953 [cs.LG], Dec 2019

[3] Yang Zhao, Chunyuan Li, Ping Yu, Jianfeng Gao, Changyou Chen. Feature Quantization Improves GAN Training. arXiv:2004.02088 [cs.LG], Apr 2020

[4] Sophie Searcy, https://soph.info/odsc2019, May 2019

[5] Shabab Bazrafkan, Peter Corcoran. Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN), Multi Class Scenarios. arXiv:1806.07751 [cs.LG], Jun 2018

[6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. arXiv:1706.08500 [cs.LG], Jun 2017

[7] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. arXiv:1802.05957 [cs.LG], Feb 2018

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. arXiv:1706.08500 [cs.LG], Jun 2017

[9] Yaniv Benny, Tomer Galanti, Sagie Benaim, Lior Wolf. Evaluation Metrics for Conditional Image Generation. arXiv:2004.12361, Apr 2020