University of Bologna

Deep Learning Project

CONDITIONAL FACE GENERATION

Presented by:
Marcello Sicbaldi
Pietro Obbiso

Abstract

In this project we investigate several techniques of generative models. Among them, we focus our attention on the ones considered by us the best in terms of results furnished. The goal here is to provide the best generation of images having specific attributes chosen directly by the user. We work with CelebA dataset containing more than 200k images and 40 attributes annotations per image. The quality of the generated images is measured using the Fréchet Inception Distance (FID).

CONTENTS

INTRODUCTION

"There are many interesting recent developments in deep learning…The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion." – **Yann LeCun**

Generative models are widely used in many subfields of AI and Machine Learning. Recent advances in parameterizing these models using deep neural networks have enabled scalable modeling of complex, high-dimensional data including images, text, and speech.
The most widespread generative deep learning models are Generative Adversarial Networks (GANs) Variational Autoencoders (VAEs).
The first consists in a generative model G which captures the data distribution, and in a discriminative model D that discriminates between real and fake samples [1]. While G tries to generate images as realistic as possible to fool D, D progressively improves its discriminative ability to push G to do better. GANs perform really well, but they are also very hard to train, and this is their main drawback.
VAEs consists of an inference network (encoder) and a generative network (decoder) working on a target latent variable model [2]. The main advantages of VAE are that it doesn't need strong assumptions, and its training is fast via backpropagation. However, the generated samples have in general a worse quality than those generated by GANs.

In this project, we explore the generative ability of different generative models, training them on the CelebA dataset [3]. By investigating different models and architectures, our goal is to find the one that provides the best image generation, both unconditional and conditional.

1. RECENT TECHNIQUES

Before starting directly with the personal experiments that we carried out, and their corresponding results, we firstly conducted a survey of the most recent techniques involving generative purposes.

1.1 Variational Autoencoder

If you are beginning to explore the field of Generative Deep Learning, a Variational Autoencoder (VAE) is ideal to kick off your journey.
Before going deeply into a Variational Autoencoder, it is crucial to analyse a simple Autoencoder.
A simple Autoencoder consists of two neural networks: an Encoder and a Decoder. The Encoder is responsible for converting an image into a compact lower dimensional vector, called latent vector, which is a compressed representation of the image. The Encoder, therefore, maps an input from the higher dimensional input space to the lower dimensional latent space. In an Autoencoder, this latent vector is fed into the Decoder. The Decoder is a different neural network that tries to reconstruct the image, thereby mapping from the lower dimensional latent space to the higher dimensional output space.
How an autoencoder can be exploited in the context of image generation? In order to be able to use the decoder of our autoencoder for generative purposes, we have to be sure that the latent space is regular enough. One possible solution to obtain such regularity is to introduce explicit regularisation during the training process. Thus, a variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative processes.
Just as a standard autoencoder, a variational autoencoder is an architecture composed of both an encoder and a decoder and that is trained to minimise the reconstruction error between the encoded-decoded data and the initial data. However, in order to introduce some regularisation of the latent space, we proceed to a slight modification of the encoding-decoding process: instead of encoding an input as a single point, we encode it as a distribution over the latent space. The encoded distributions are chosen to be normal so that the encoder can be trained to return the mean and the covariance matrix that describe these Gaussians. The reason why an input is encoded as a distribution with some variance instead of a single point is that it makes it possible to express very naturally the latent space regularisation: the distributions returned by the encoder are enforced to be close to a standard normal distribution.
The loss function that is minimised when training a VAE is composed of a “reconstruction term” and a “regularisation term” (on the latent layer), that tends to regularise the organisation of the latent space by making the distributions returned by the encoder close to a standard normal distribution. That regularisation term is expressed as the Kulback-Leibler divergence between the returned distribution and a standard Gaussian, which can be directly expressed in terms of the means and the covariance matrices of the two distributions.
Summing up, the full architecture of a VAE results as the following one:

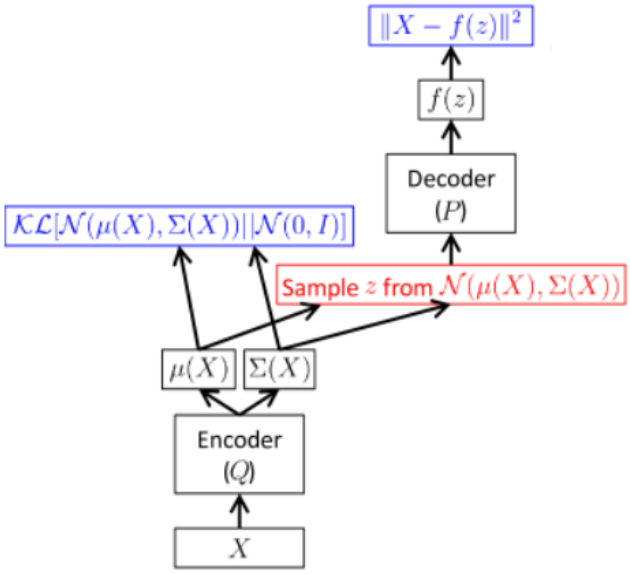


Figure 1: VAE Architecture

1.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) provide an attractive alternative to variational autoencoders as generative models. GANs are composed of two competing neural networks, which are trained together. One is a generator network that takes random noise as input and generates samples; the other is a discriminative model that tries to differentiate between samples from generative source and real training data. So the generator generates a batch of samples, and these, along with real examples from the domain, are provided to the discriminator and classified as real or fake (figure 1).
The discriminator is then updated to get better at discriminating between real and fake samples in the next round, and importantly, the generator is updated based on how well, or not, the generated samples fooled the discriminator.

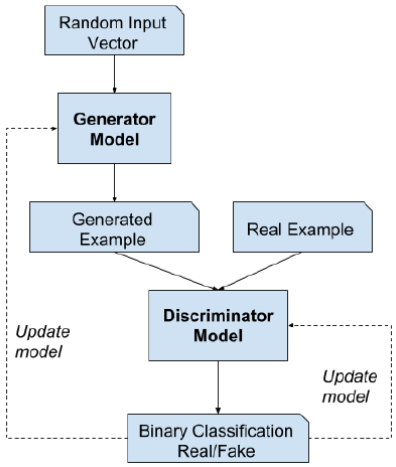


Figure 2: GAN architecture

More formally, considering the function $v(\theta_g, \theta_d)$, where θ_g and θ_d are the parameters of the generator G and discriminator D respectively, we can formulate GAN training as optimizing:

$$\min_g \max_d v(\theta_g, \theta_d) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))], \quad (1)$$

where z is a vector noise sampled from a known simple distribution p_z (e.g. normal).

1.3 Conditional GANs

Although GAN models are capable of generating new random plausible examples for a given dataset, there is no way to control the types of images that are generated.
However, by conditioning the model on additional information, such as a class label, it is possible to direct the data generation process. There are two motivations for making use of the class label information in a GAN model: improve the GAN and allow for targeted images generation. The improvement may come in the form of more stable training, faster training, and/or generated images that have better quality.
This method was first introduced in the paper *Conditional Generative Adversarial Nets* [4] where the authors point out the possibility to perform the conditioning by feeding **y,** the auxiliary extra information, into both the discriminator and generator as an additional input layer. Specifically, the authors performed the conditioning by feeding **y** into both the discriminator and generator as additional input layer. Both D and G were multi-layer perceptrons (MLPs) in this case.
But what if D and G are Convolutional Neural Networks?
Since the generator always takes as input the noie vector, $\mathbf{y} \sim \mathbf{p}_{\text{data}}$ and $\mathbf{z} \sim \mathbf{p}_z$ (where $\mathbf{p}_z = N(0, 1)$) can be simply concatenated at the input level [5-7].
On the other hand, the discriminator takes images as input: in this case the natural solution is to repeat the label vector for each pixel and concatenate it to the input image along the channel dimension (**vectorization)**.
Another important design decision concerning the discriminator is where (in which layer) to insert **y.** Different authors concatenate **y** in different parts of the discriminator: in the last fully connected layer [6], after the first convolutional layer [5], or in multiple layers [7]. The literature results show that the earlier **y** is positioned in the model the better: this might be due to the fact that the model is allowed to have more learning interactions with the label vector.

1.4 Auxiliary Classifier GANs

A more novel implementation of the class specified generative model is the Auxiliary Classifier GAN (ACGAN) [8] wherein by adding a classification term to the generator and discriminator loss, the generator is forced to generate a specific class of data for a given input.
ACGAN is similar in principle to the Conditional GAN (cGAN) that we discussed in the previous chapter. For both cGAN and ACGAN, indeed, the generator model is provided both with a point in the latent space and the class label as input, e.g. the image generation process is conditional. What differs the two models is in the discriminator. While for the cGAN, the input to the discriminator is an image

(fake or real) and its label, the discriminator model in the ACGAN is only provided with the image as input. It must then predict whether the given image is real or fake as before, and must also predict the class label of the image.
The plot below summarizes the inputs and outputs of the two GAN models described so far:

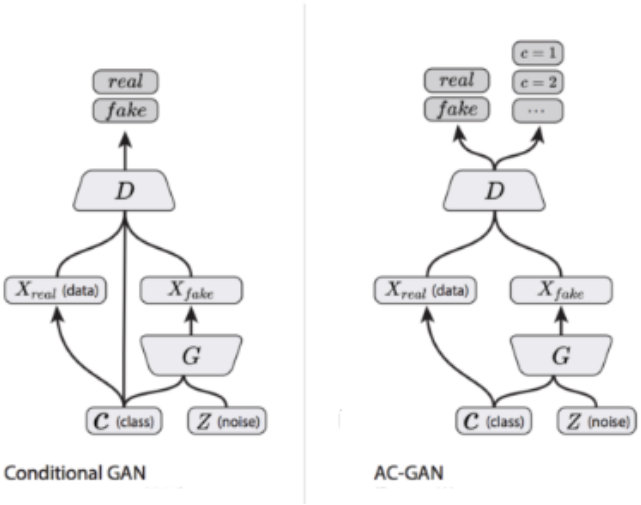


Figure 3: cGAN vs ACGAN

The architecture is described in such a way that the discriminator and auxiliary classifier may be considered separate models that share model weights. In practice, the discriminator and auxiliary classifier can be implemented as a single neural network model with two outputs.

Two years later, Bazrafkan et al [9] presented a slight variation to the ACGAN, introducing the Versatile Auxiliary Classifier with Generative Adversarial Network (VAC + GAN). They added a classifier network which is trained separately with respect to the discriminator: the two models do not share any weights. So the classification term (in ACGAN) is taken out of the discriminator's loss function by adding a classifier network that back-propagates through the generator.

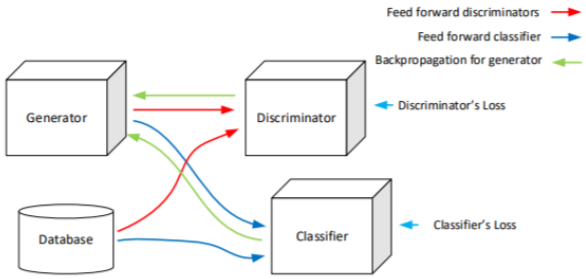


Figure 4. VAC + GAN structure

The authors show that this framework is able to increase the Jensen Shannon Divergence (JSD) between classes generated by the generator. i.e., the generator can produce samples drawn from a desired class. When compared to cGANs for gender specific face generation, the proposed method is able to generate a higher variation of samples for each class and also between classes.

2. DATASET

All the above-mentioned techniques, experiments and results, extracted from some relevant papers, helped us to first have a general idea about these topics and then, most crucial, to guide us in the development of our personal experiments.

Our model is trained on the CelebFaces Attributes (CelebA) Dataset. CelebA is a large-scale face attribute dataset with 202,599 face images and 40 binary attributes annotations per image (figure.
CelebA is characterized by a wide variety of face expressions, views and backgrounds.

CelebA Facial Attributes:

0: 5_o_Clock_Shadow	20: Male
1: Arched_Eyebrows	21: Mouth_Slightly_Open
2: Attractive	22: Mustache
3: Bags_Under_Eyes	23: Narrow_Eyes
4: Bald	24: No_Beard
5: Bangs	25: Oval_Face
6: Big_Lips	26: Pale_Skin
7: Big_Nose	27: Pointy_Nose
8: Black_Hair	28: Receding_Hairline
9: Blond_Hair	29: Rosy_Cheeks
10: Blurry	30: Sideburns
11: Brown_Hair	31: Smiling
12: Bushy_Eyebrows	32: Straight_Hair
13: Chubby	33: Wavy_Hair
14: Double_Chin	34: Wearing_Earrings
15: Eyeglasses	35: Wearing_Hat
16: Goatee	36: Wearing_Lipstick
17: Gray_Hair	37: Wearing_Necklace
18: Heavy_Makeup	38: Wearing_Necktie
19: High_Cheekbones	39: Young

Figure 5. CelebA attributes

2.1 Data pre-processing

Before focusing on the implementation of the models and their respective architectures, it was needed a short but essential phase of pre-processing.

Right after loading the data, we cropped CelebA faces at position [45:173,25:153], in order to concentrate more on the region of interest of the faces and not on background details. Then we reduced the image size to 64x64. Eventually, we scaled the pixel values of each image to the range [0,1] or [-1,1], depending on whether we were using VAEs or GANs.

3. EXPERIMENTAL RESULTS

3.1 Conditional VAE

As beginners we first chose the VAE model, which we thought being the best approach to start, as VAEs are known to be easier to train than GANs.

We decided to implement the encoder and decoder of our VAE using a convolutional architecture. Here are the details on our model and training configurations:

Encoder: 4 x
- *Conv2D* layer
- *BatchNormalization* layer
- *LeakyReLU* as activation layer (*alpha = 0.2)*

Decoder: 4 x
- *Conv2DTranspose* layer
- *BatchNormalization* layer
- *LeakyReLU* as activation layer (*alpha = 0.2)*

Latent space dimension: 64

Optimizer: *Adam, learning rate = 0.0005*

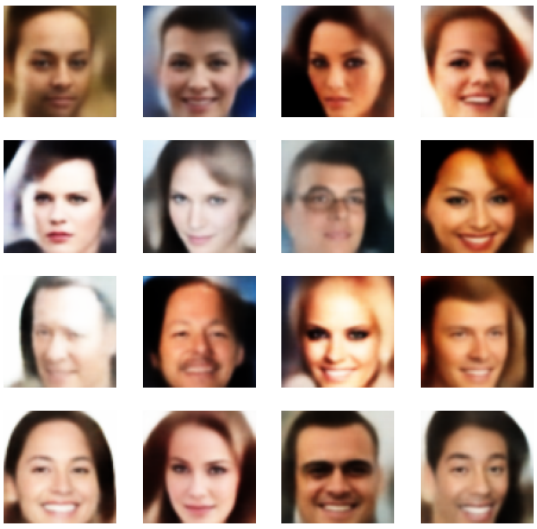
Loss: weighted sum between regularization term (KL divergence) and reconstruction term (Mean Squared Error) → $KL + \beta * MSE$, $\beta = 10000$

VAE Input: Batches of 128 images

Epochs: 100

In the following figures we display the reconstructed and generated images obtained with this first experiment.

Reconstruction



Generation

Although the visual quality is not bad and the images clearly represent human faces, they are a bit blurry. Moreover, the FID for generated images is 74.30, which is quite far from the state-of-the-art FID of generative models trained on CelebA.
Because of this we decided to leave the VAE model behind and enter the more promising world of GANs.

3.2 cGAN

The goal of this experiment is to test the impact of adding the labels vector in different layers of our cGAN model.
For all experiments we used the same DCGAN (Deep Convolutional Neural Networks) [10] architecture, structured in the following way:

GENERATOR

Operation	**Kernel**	**Stride**	**Filters**	**BN**	**Activation**
Concatenation	*Concatenate a random label sampled from the label distribution with latent vector*				
Transpose Convolution	4 x 4	2 x 2	128	Yes	Leaky Relu
Transpose Convolution	4 x 4	2 x 2	128	Yes	Leaky Relu
Transpose Convolution	4 x 4	2 x 2	128	Yes	Leaky Relu

Transpose Convolution	4 x 4	2 x 2	128	Yes	Leaky Relu
Convolution	5 x 5	1 x 1	3	No	Tanh

DISCRIMINATOR

Operation	**Kernel**	**Stride**	**Filters**	**BN**	**Activation**
Convolution	5 x 5	2 x 2	128	No	Leaky Relu
Convolution	5 x 5	2 x 2	128	No	Leaky Relu
Convolution	5 x 5	2 x 2	128	No	Leaky Relu
Convolution	5 x 5	2 x 2	128	No	Leaky Relu

After the last convolution, we flattened the 4 x 4 feature map and fed it into a Dense Layer with 1 output neuron, followed by a sigmoid activation.

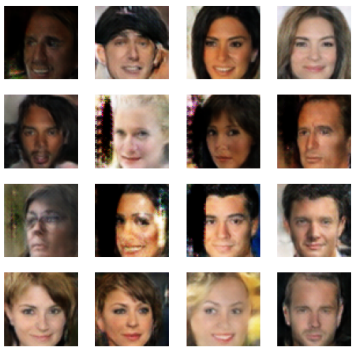
We use a latent dimension of 100.
For what concerns the generator model, we always concatenated the random labels vector and the latent vector at input level. All models are trained with the Adam optimizer ($\beta_1 = 0.5$) with a learning rate of 0.0002 (as suggested in the DCGAN paper) and a mini-batch size of 64 samples during 100 epochs. We used Binary Cross-Entropy as the objective function.
Now we present various experiments regarding the optimal position of **y** in the discriminator: cGAN#1, cGAN#2, cGAN#3, cGAN#4, cGAN#5. As specified in section 1.3, we concatenate **y** by means of vectorization: when the dimension of a layer in D is $n \times n \times d$, we replicate the vector c to match the size $n \times n$ of the feature map and perform a depth concatenation.

- *cGAN#1*: Here we concatenated **y** with the input image, before the first convolution. The results were not satisfying, as we obtained a FID of 37.2.
- *cGAN#2*: In this model we inserted **y** after the first convolution. The quality of generated images improved significantly: this is in fact reflected by a FID score of 11.9. The model is also able to generate images according to specific attributes (section 3.5)
- *cGAN#3*: This model is the fusion of the two models above, as we concatenated the condition both at input level and after the first convolution. The generative ability of this model falls between cGAN#1 and cGAN#2, as the FID score is 30.9. By looking at this model and at cGAN#1 we can deduce that conditioning the discriminator at input level is not optimal in terms of the quality of generated images.
- *cGAN#4*: We then tried to concatenate **y** with the flattened 4 x 4 feature maps obtained after the last convolution, which can be thought as a more natural solution, since it is a simple concatenation between two vectors and does not require vectorization. Although the quality of

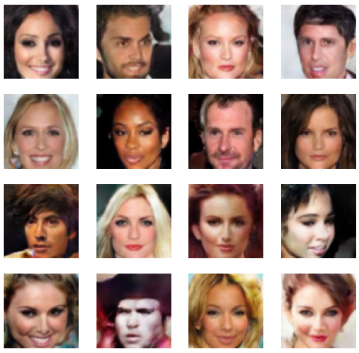
the generated images was satisfying (FID score = 18.8), this model was not able to generate images with specific attributes. We believe that inserting **y** in the last layer causes the model to have insufficient learning interactions with it, and so the network is not able to learn the additional information.

- *cGAN#5*: This time we concatenated **y** after the second convolution, and we obtained a FID score of 12.4.
- *cGAN#6*: We inserted **y** after the first and second convolutions, hoping to improve our cGAN#2 and cGAN#5 models by providing the network more information regarding the labels. We did not observe any improvements as we obtained a FID score of 14.5.
- cGAN#7: In our last experiment we inserted y in every layer except for the last. The quality of generated images did not improve (FID = 14.6).

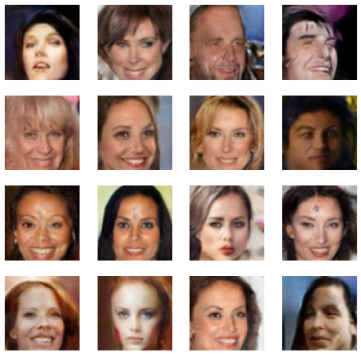
All these experiments show that, in terms of FID score for generated images with random labels, the position in which we insert **y** in the discriminator is not crucial, as long as it is not at the input level.
The comparison between the conditional abilities of these networks will be better explored in section 3.5.



cGAN#1. FID = 37.2



cGAN#2. FID = 11.9



cGAN#3. FID = 30.9

Batch normalization

During our experiments we observed that applying Batch Normalization (BN) in the generator after every convolution except for the last improved the FID score for generated images (Figure 6). These improvements are not seen when BN is applied also to the discriminator.

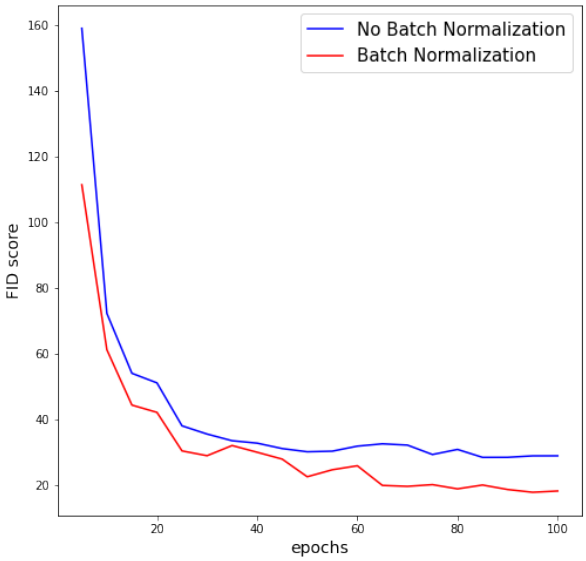


Figure 6. How batch normalization influences FID score during training

3.3 ACGAN

After spending a lot of time experimenting with cGANs, we decided to explore a more novel architecture, the ACGAN.
We used the same architecture shown in the previous section for both the generator and the discriminator. Since ACGAN does not require the concatenation of **y** in the discriminator, it has a clear advantage w.r.t. cGANs: less hyperparameters to optimize.
A slight architectural change was made in the discriminator, that needs to have two output layers (Figure 7).

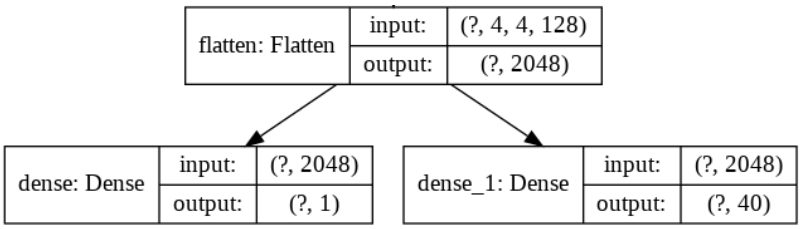


Figure 7.

The first is a single node with the sigmoid activation for predicting the realness of the image. The second is multiple nodes, one for each class. One could think to apply softmax activation after this layer, since we are dealing with multi-class classification; however, the CelebA labels are not mutually exclusive, while the assumption of mutual exclusivity is an important component of softmax. On the other hand, the sigmoid activation function converts each score of the final node between 0 and 1 independent of what the other scores are: for this reason we chose sigmoid as activation after our multiple nodes layer.

The training configurations are the same as the one described in section 2.2.
The top FID score we obtained was 14.2, similar to our best cGANs models.
The next figure displays a set of generated faces with random attributes:

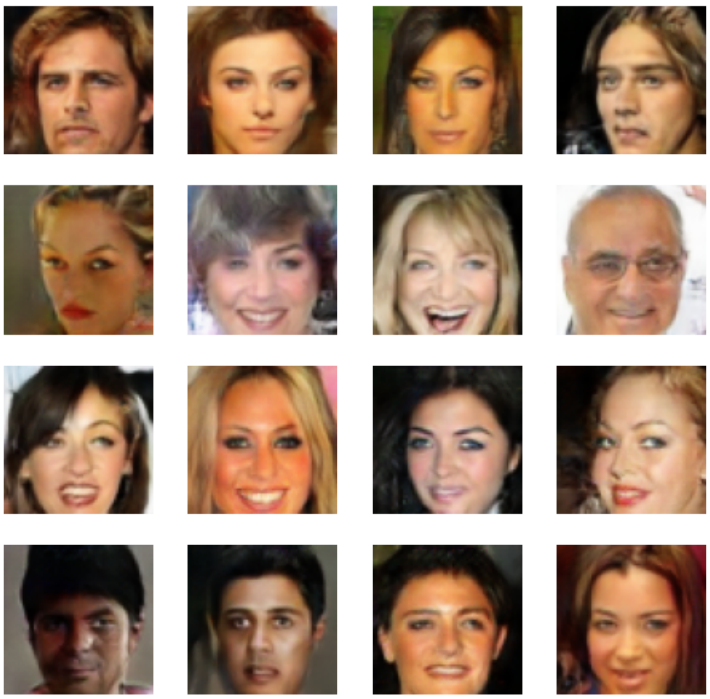


Figure 8. FID score = 14.2

3.4 VAC - GAN

Inspired by [11] we decided to carry out our last experiment by trying the VAC + GAN architecture. The most relevant changes w.r.t to our previous experiments are the use of Spectral Normalization [12] in both D and G and hinge loss [13] instead of binary crossentropy.

Spectral normalization

One of the most relevant challenges when applying generative adversarial networks is the easy instability of its training. Paper *Spectral Normalization for Generative Adversarial Networks* [12] illustrates how to stabilize the training process of the discriminator by introducing a new weight normalization technique called Spectral Normalization, where its implementation is quite simple, and the additional computational cost is small.

Spectral normalization imposes a global regularization on the discriminator, allowing the use of as many features as possible, leaving more freedom in choosing the number of singular components (features) to feed to the next layer of the discriminator.

When the authors applied spectral normalization to the GANs on image generation tasks, the generated examples were more diverse than the conventional weight normalization and achieved better or comparative inception scores (FID score for example) relative to other studies obtained by using different normalization techniques.

Hinge loss

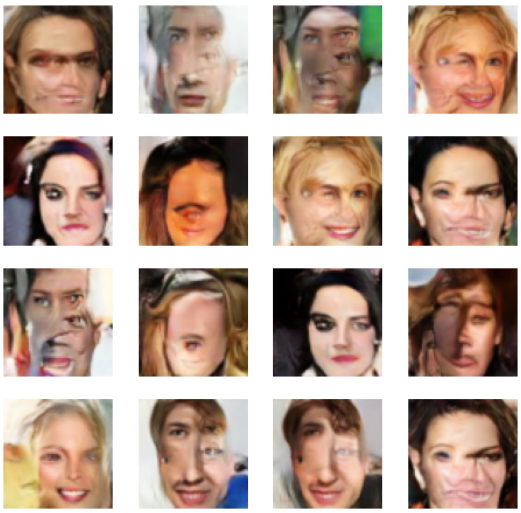
When combined with spectral normalization of weights in the discriminator, the hinge loss greatly improves performance, and has become a mainstay in recent state of the art GANs [13].

$$L_G = -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0, -1 - D(G(z),y))]$$

$$L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z),y)$$

Going back to our experiment, we used the architecture shown in section 2.2 for the generator, the discriminator and the classifier.

Although we expected to better stabilize the training process with this experiment, the results proved us wrong, as we experienced almost all the common problems with GANs we have seen during the lessons. In the early stages of training we had problems with global structure and counting, while as the training went on the model experienced mode collapse.



Problems with counting and structure

Mode collapse

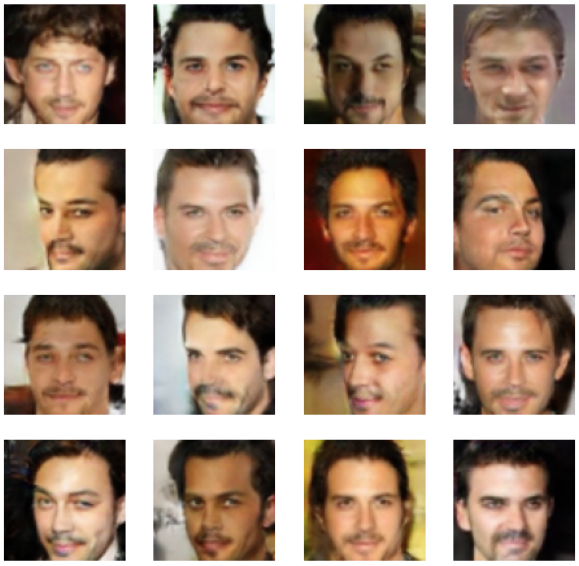
3.5 Conditional Image Generation

In this section we explore how well our generator models can generate faces with specific attributes.

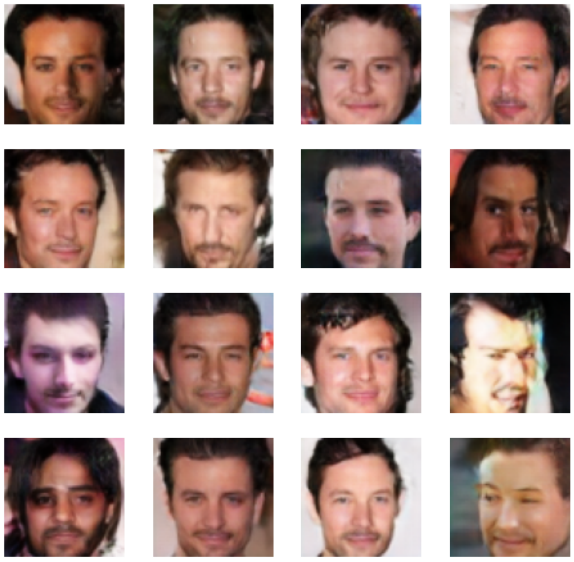
We visually compared the ability of our best (in terms of FID) cGAN models (*cGAN#2, cGAN#4, cGAN#5, cGAN#6 and cGAN#7*) and *ACGAN* model to conditionally generate faces.

Apart from *cGAN#4*, in which we injected **y** in the fully connected layer of D, all other models are able to generate faces with specific attributes. We did not observe significant differences between the models, as we can see in the following figure.

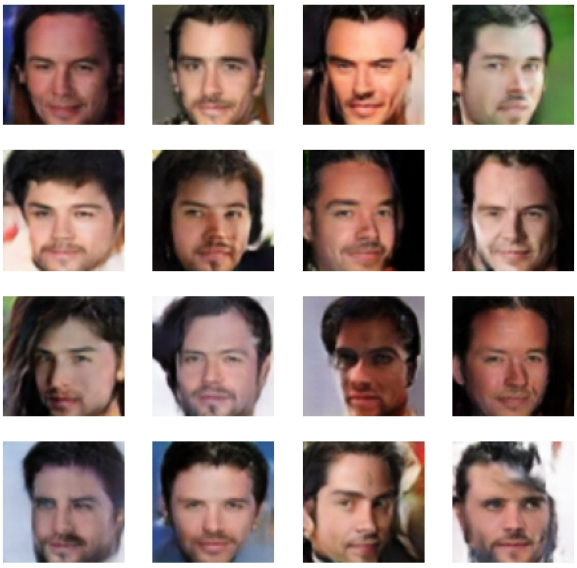
Male, Young, Attractive, Black Hair, Smiling, Goatee



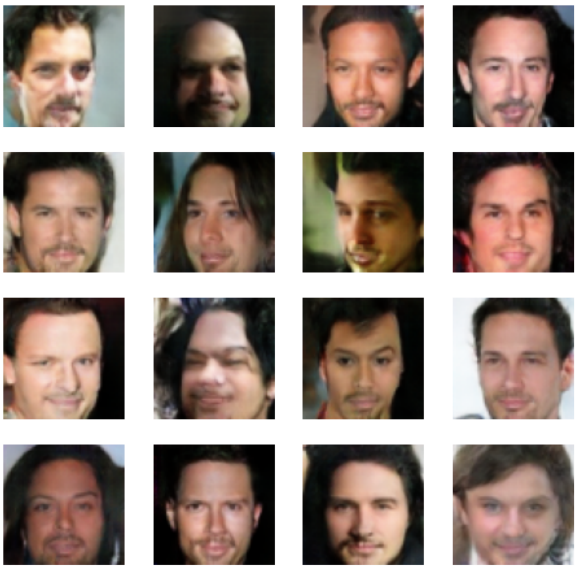
cGAN#2



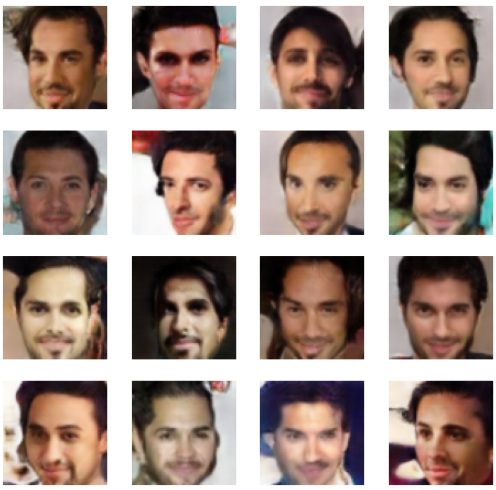
cGAN#5



cGAN#6

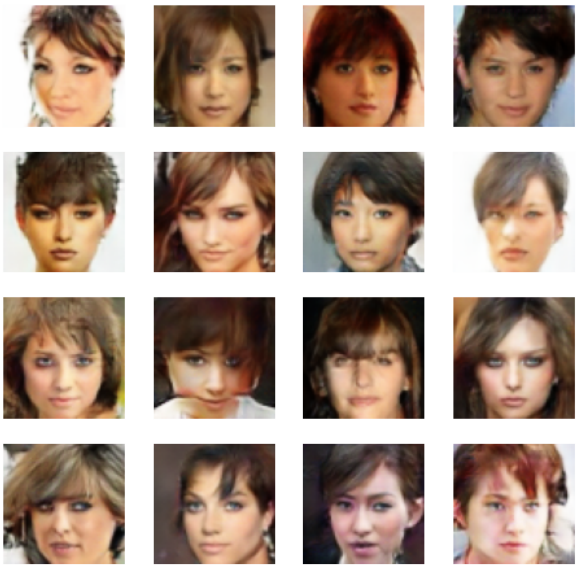


cGAN#7



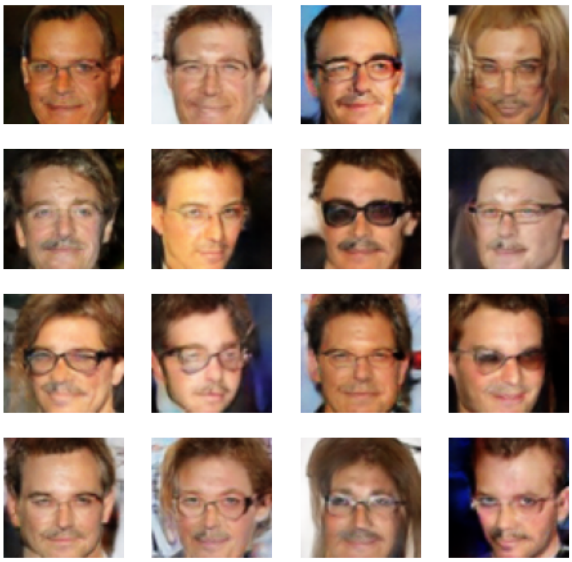
ACGAN

Since we did not visually observe major differences in the conditional abilities of our models, we decided to stick with the one which provided the best FID, namely *cGAN#2* (**y** inserted after the first convolution). Below we display some generated images with different attributes.

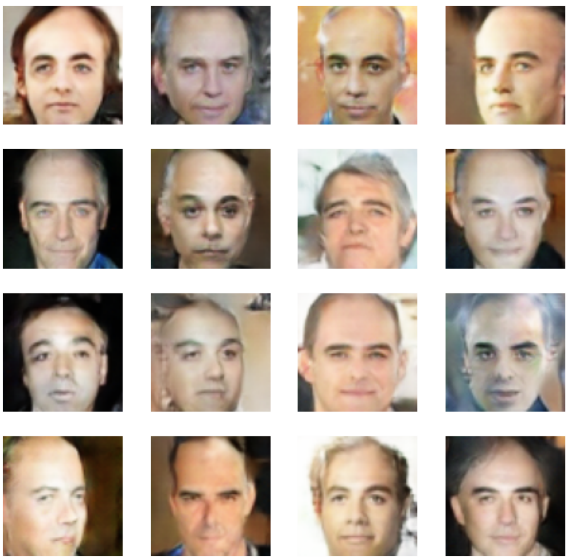


Female, young, attractive, brown_hair, bangs, heavy_make_up, no_beard

Female, young, attractive, blond_hair, smiling, eyeglasses, no_beard



Male, attractive, brown_hair, straight_hair, smiling, eyeglasses, mustache



Male, pale_skin, bald, bushy_eyebrows, arched_eyebrows, no_beard

4. Evaluation Metrics

A problem with generative models is that there is no objective way to evaluate the quality of the generated images. As such, it is common to periodically generate and save images during the model training process and use subjective human evaluation of the generated images in order to both evaluate the quality of the generated images and to select a final generator model. Many attempts have been made to establish an objective measure of generated image quality. An early and somewhat widely adopted example of an objective evaluation method for generated images is the Inception Score, that

Fréchet Inception Distance, or FID, was developed starting from the IS and it gives a more reliable standard to measure the performance of Generative Models, because it compares the statistics of synthetic images with those of real images, instead of only evaluating the quality of the generated samples as Inception Score does.

4.1 FID

The Fréchet Inception Distance score, or FID for short, 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 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.
FID score is used to evaluate the quality of images generated by generative models, and lower scores have been shown to correlate well with higher quality images.
FID score is calculated by first loading a pre-trained Inception v3 model. The output layer of the model is removed and the output is taken as the activations from the last pooling layer, a global spatial pooling layer. This output layer has 2048 activations; therefore, each image is predicted as 2,048 activation features.
A 2048 feature vector is then predicted for a collection of real images from the problem domain to provide a reference for how real images are represented. Feature vectors can then be calculated for synthetic images. The result will be two collections of 2,048 feature vectors for real and generated images. The FID score is then calculated using the following equation taken from the paper:

$$d^2 = ||mu_1 - mu_2||^2 + Tr(C_1 + C_2 - 2 \times \sqrt{C_1 \times C_2})$$

The mu_1 and mu_2 refer to the feature-wise mean of the real and generated images, e.g. 2,048 element vectors where each element is the mean feature observed across the images. The C_1 and C_2 are the covariance matrix for the real and generated feature vectors.

In our work we've used the pre-trained Inception v3 model of Keras to extract the features vectors (activations) necessary to compute mu_1, mu_2, C_1, C_2. We used a sample of 10000 images to compute the FID.

4.2 Conditional evaluation

Fréchet Inception Distance (FID) is very useful to evaluate unconditional generative models, since it measures the distance between the real and the generated images distribution, without taking into account the conditional aspect. However, employing FID in order to evaluate conditional image generation fails to take into account whether the generated images satisfy the required condition.
In this section we propose some metrics to assess conditional image generation, and we discuss the advantages and disadvantages of each.

The first evaluation metric we propose are classification metrics, such as accuracy and precision.
To exploit these metrics, we could train a classifier on CelebA in order to obtain an attribute prediction network (Anet); subsequently we can use the generator to create images conditioned on specific attribute vectors and make the Anet predict them. If the predicted Anet attributes are closer to the original attributes used to generate an image, it should mean that the generator has successfully learned the capability to generate new images considering the semantic meaning of the attributes.
The main downside of this metric is that it does not take into account image quality and diversity.
One could now think to combine FID score and classification metrics to produce a valuable measurement for conditional generation. Nevertheless, these two metrics are of different scales, so the trade-off between the two is unclear: understanding how to evaluate conditional performance by combining them is not a straightforward task.

The second metric we would like to propose is an extension of FID to the class-conditional setting.
If FID measures the distance between generated images and real images, one could think to measure the FID between generated images with specific attributes and real images with the same attributes.
In this way we can evaluate how similar each conditioned class is to its respective real class. The total score would be the mean FID within the classes.
However, due to the peculiar nature of CelebA, we think that this approach has some major downsides. This is because most of the labels are not mutually exclusive: the number of possible attribute combinations (namely classes) is up to 2^{40} (in reality a bit less because there are some attributes that are mutually exclusive).
For this reason, computing the conditional FID for each class is infeasible; one could instead select a subset of attributes from which to compute the FID, but in this way all the other attributes would not be taken into account.
On the grounds of these considerations we do not think that this method is comprehensive enough to evaluate the conditional generation. This approach could instead work well with other datasets for which the number of different classes is clearly defined, i.e. Minst.

5. CONCLUSION AND FUTURE WORK

Face generation conditioned on facial attributes is a complex task that requires a model to learn not only the general data distribution for face images, but also the additional information about the labels. In this project we developed an extension of the generative adversarial net framework, by adding the ability to condition on arbitrary external information to both the generator and discriminator component.

We have demonstrated that well-designed conditional GANs (CGANs) can perform admirably on this very complex task. We evaluated the model on the CelebA dataset and demonstrated that the conditional information **y** can be used to deterministically control the output of the generator.

It must be noted that we trained all our model using the free GPU offered by the Google Colab environment: if we had access to more powerful computational resources we could have experimented with more complex models and architectures, obtaining hopefully best results.

6. REFERENCES

[1] Goodfellow Ian et al. “Generative adversarial nets.” *Advances in neural information processing systems*. 2014.

[2] Diederik P Kingma, Max Welling. “Auto-Encoding Variational Bayes” 2014

[3] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[4] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets.", 2014.

[5] Guim Perarnau et. al. “Invertible Conditional GANs for image editing”, 2016.

[6] Jon Gauthier et. al. “Conditional generative adversarial nets for convolutional face generation”, 2016.

[7] Xuwen Cao et. al., “Face Generation with Conditional Generative Adversarial Networks”, 2017.

[8] Augustus Odena et. al., “Conditional Image Synthesis With Auxiliary Classifier GANs”, 2016.

[9] Bazrafkan et. al. "Versatile auxiliary classifier with generative adversarial network (vac+ gan).", 2018.

[10] Radford, Alec, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with Deep convolutional generative adversarial networks.” 2015.

[11] https://soph.info/slides/faces.pdf

[12] Miyato, Takeru, et al. "Spectral normalization for generative adversarial networks.", 2018.

[13] Ilya Kavalerov et. al., “cGANs with Multi-Hinge Loss”, 2019