Machine Learning in Applications
AM1-02 Project Report

Damiano Bonaccorsi, Daniele Rege Cambrin, Giulia D'Ascenzi, Patrizio de Girolamo
Politecnico di Torino

CONTENTS

I Introduction 1

II Background 1
II-A Facial Attribute Editing . . . 1
II-B Conditional Variational Autoencoders . . . 2
II-C Generative Adversarial Network. . . . 2

III AttGAN 3
III-A Architecture components . . . 3
III-B Losses . . . 4
III-C Proposed modifications . . . 4

IV Implementation details 5
IV-A Network . . . 5
IV-B Metrics . . . 5
IV-C Training details . . . 5
IV-D Dataset . . . 5

V Experiments 6
V-A Varying the number of shortcut layers . . . 6
V-B Pretraining the discriminator . . . 6
V-C Choosing the optimal subset . . . 6
V-D Finetuning the network . . . 6
V-E Final comparison . . . 7

VI Conclusions and future works 7

References 9

LIST OF FIGURES

1 Adding eyeglasses to our faces . . . 1
2 Autencoder (AE) . . . 2
3 Variational Auto Encoder (VAE) . . . 2
4 Encoder-Decoder architecture with skip-connections . . . 2
5 AttGAN architecture . . . 3
6 Varying the number of shortcut layers . . . 7
7 Effect of FreezeD on image modification. . . . 7
8 Effect of λ_1 on image modification. . . . 7
9 Effect of *Alternate* training on image modification at different epochs. . . . 7
10 Effect of "Eyeglasses" modification in AttGAN and finetuned AttGAN . . . 8
11 Impact of the different weight of reconstruction loss on generator loss . . . 8
12 Impact of pretraining discriminator . . . 8
13 Impact of different subsets on generator loss . . . 8
14 Impact of FreezeD on generator loss . . . 8
15 Impact of the different weights of reconstruction loss and FreezeD on generator loss using *Alternate* training. . . 8

LIST OF TABLES

I AttGAN architecture . . . 4
II The top-10 attribute in subsets ordered by descending percentage . . . 6

Machine Learning in Applications AM1-02 Project Report

***Abstract*—The goal of this project is to create a social media-like filter that is able to add eyeglasses to the face depicted in an input image. The task is known in literature as "facial attribute editing" and to handle it we chose as a baseline a well-known generative adversarial network called AttGAN [1]. The original network is able to handle arbitrary attributes' changes, therefore, we started by modifying the training procedure to make it focus on the eyeglasses attribute only. Then, we studied the impact that adding skip-connections in the encoder-decoder architecture has on the overall network performance. Finally, due to limited temporal and computational resources, we exploited the pretrained model partially provided by the AttGAN authors and finetuned it for our specific task.**

In this report, we present the modifications applied to the original training procedure and the results we obtained.

The code produced during this project can be found at https://github.com/MLinApp-polito/mla-prj-02-am1.

I. Introduction

Everyday, on various social media apps, millions of users apply filters to their photos modifying facial details. Usually, they do not realize they are using very sophisticated technologies and that even an apparently trivial task such as adding eyeglasses to an input image, hides many challenges.

This type of filter performs what in literature is known as "facial attribute editing task", manipulating an input image depicting a human face by changing one or more facial attributes avoiding undesired modifications to those details that are not directly involved in the process. It is challenging since the human face has a complex structure and the desired attributes are often only described in terms of their presence or absence, without including any more information concerning style and other peculiar features that are unique to a certain face. Moreover, the training cannot be supervised by the ground truth, since it would be unfeasible to collect labelled images of the same person with varying attributes and, therefore, the task falls in the unsupervised learning scenario.

In this project we aim to obtain a neural network able to act as a social media filter, modifying the face depicted on the input image by adding eyeglasses. To tackle the mentioned challenges we used as starting point architecture AttGAN [1], a well-known facial attribute editing network.

Firstly, we studied how AttGAN works, carefully analysing its architecture, the training steps and the losses.

Then, we adapted the training procedure, originally designed to perform well enough in multi-attributes editing, to make it focus on one attribute at the time, therefore simplifying its original task.

Next, as other papers proposed [2] [3], we modified the architecture adding skip-connections between the encoder and decoder layers, checking the benefits and limitations that such change introduces in the overall performances.

Fig. 1: Adding eyeglasses to our faces

Since it was impractical to train the network from scratch due to the limited resources available, we exploited the pretrained model provided by AttGAN's authors and we conducted several experiments to evaluate the goodness of our proposals.

Finally, we present the obtained results, offering a visual comparison of them.

An example of the desired filter can be seen in Figure 1.

II. Background

A. Facial Attribute Editing

The objective of facial attribute editing is to generate a face with a certain target attribute while preserving the attribute-excluding facial details. In the last decade, several methods have been proposed to achieve this goal and can be split into two categories: optimization-based methods and learning-based methods.

The optimization-based methods try to narrow the gap between the features of the input image and the features of images having the target attributes. For example, Upchurch et al. [4], look for a feature space where image editing can be performed by simply applying a linear interpolation between images with a certain attribute and images without it and optimizing the input image to match the moved features. However, this kind of methods is laborious and time-consuming.

The learning-based methods instead are much more popular. The initially-proposed ones, as [5] and [6], are networks that could perform manipulation of a specific combination of attributes only. Later methods proposed single models able to perform arbitrary multi-attribute editing. These methods widely make use of Variational Autoencoders (VAE) [7] and Generative Adversarial Networks (GAN) [8]. IcGAN [9], for example, generates an attribute-independent latent representation, that is then used by a conditional GAN to produce an image with the desired attributes. On the other hand, StarGAN [3] proposes a conditional attribute transfer network and uses a cycle consistency loss to preserve key attributes between the input and the output image. Finally, AttGAN [1] uses a generator, also based on an autoencoder architecture, that creates the output image, and a discriminator that forces the output image to be visually realistic. Additionally, an

attribute classification constraint is applied to the generated image, to guarantee the correct change of the attributes, and a reconstruction loss is used to ensure that the generated image is as visually similar to the input image as possible.

B. Conditional Variational Autoencoders

The original autoencoder architecture has been presented by Hinton et al [10], and consists of two major components: an encoder to map the input image into a latent space and a decoder that, starting from the latent space, reconstructs the input image as shown in figure (2). The primary goal of this kind of network is mainly to reduce the dimensionality of the feature space.

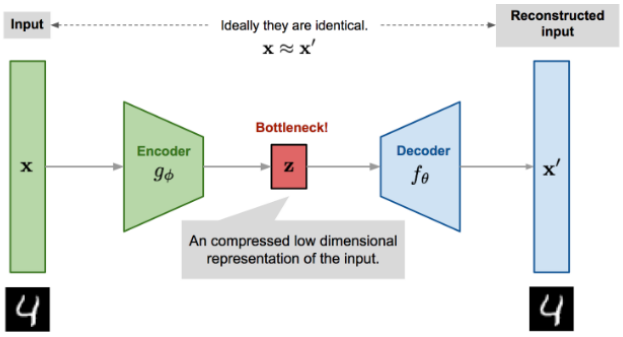

Fig. 2: Autencoder (AE) from [11]

Later, the Variational Autoencoder [7] architecture demonstrated the ability of the encoder-decoder pair to generate unseen data.

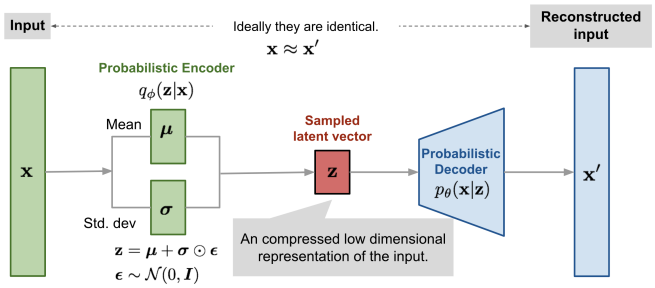

Fig. 3: Variational Auto Encoder (VAE) from [11]

As Figure 3 shows, in variational autoencoders, the input image is not mapped into a fixed vector, but rather it is mapped into a distribution, by having the encoder predict mean and variance for each input image. The decoder then samples a random point from the distribution and forces the encoder to learn a latent space mapping where similar points lead to similar images. The limitation of the VAE model is that it does not allow to control the kind of data that will be generated, since the test-time input will be a sample chosen from the latent space, not supervised by the user. For this reason, in this project, we used another version of VAE, based on the Conditional Variational Autoencoder (CVAE) [12] architecture, that provides a way to control the data generation process, performing structured output predictions. This is done by supplying the network with a control variable as extra input, that conditions the type of generated output.

Finally, experimental results [13] [14]have shown that adding skip connection between the encoder and the decoder layers, usually increases the training stability and also improves the visual quality of the generated images. The changes in the architecture are inspired by UNet [15] and can be seen in Figure 4.

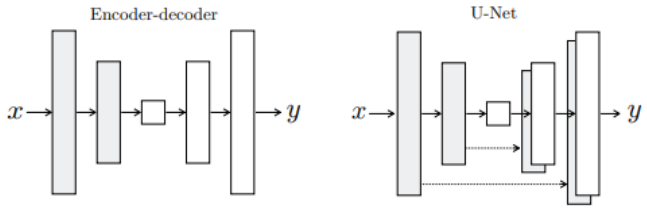

Fig. 4: Illustration from [13] of how a generator with an encoder-decoder architecture changes when adding skip-connection between mirrored layers

C. Generative Adversarial Network.

Generative Adversarial Networks (GANs) are architectures designed to generate images using an adversarial process introduced by Goodfellow et al. [8].

These networks are composed of the generator and the discriminator, which are trained simultaneously while playing a min-max game. The generator takes as input random noise and tries to fool the discriminator by producing real-looking images. On the other hand, the discriminator tries to distinguish between real and fake images, considering as ”real” the images of the training set and as ”fake” the ones created by the generator.

Stable training techniques. This kind of model has had enormous success in the last years, being able to produce astonishing results. Nevertheless, GANs' architecture implies many difficult aspects that need to be considered during training:

- *Non-convergence*: GAN is a zero-sum non-cooperative game, if one wins the other loses. In game theory, GAN converges when the discriminator and the generator reach a Nash equilibrium, namely a point where one player will not change action irrespective of the opponent's one. In practice, reaching this equilibrium is not easy and the training is indeed unstable, with the loss function constantly shifting.
- *Vanishing gradients*: this effect can occur when the quality of the predictions supplied by the discriminator is so good that do not provide enough information for the generator to make progress. In these situations during backpropagation, the gradient gets increasingly smaller, making the network learn very slow or making it stop learning completely.
- *Mode collapse*: the goal of the generator is to produce a wide variety of fake data that mimics the real ones. If during training, the discriminator gets stuck in a

local minimum, the generator will lazily keep generating samples that are similar or identical. This issue is also called the "Helvetica scenario".

To address these difficulties, multiple techniques have been proposed. In DCGAN [16] the authors suggested some constraints on the GAN architectures to improve the stability of the network: replacing spatial pooling layers with strided convolutional, using batch normalization and ReLu and Leaky Relu as activation functions. DCGAN quickly became the de facto standard GAN architecture for image generation problems. Subsequently, to further enhance the training stability and avoid mode collapse, Arjovsky et al [17] suggested using the Wasserstein distance, which measures the similarity between the real data distribution and the generated data distribution, as a robust loss for the adversarial training. Later WGAN-GP [18] made the process even more stable by adding a gradient penalty regularization term on the discriminator loss. In this project, we adopted WGAN-GP for the adversarial learning, as done by AttGAN's authors, too.

Finally, a carefully tuned learning rate may mitigate some of the cited GAN's problems. In our experiment, to induce stability, especially in the initial phase of training, we tested a warmup learning rate schedule [19]: it starts training using a very low (sometimes zero) learning rate and steadily increases it until it reaches the desired value, usually in 1 or 2 epochs. This prevents the model from taking drastic decisions during the very first steps, that would then need to be corrected as training goes on, resulting in a slightly slower, but much stabler, beginning.

Fine-tuning techniques in GANs. The classical way of training GANs is from scratch, requiring a large amount of training data and heavy computational resources. However, when the resources at disposal are not sufficient, it would be useful to exploit fine-tuning of a model pretrained on a bigger dataset. Even if this is a classical solution in discriminative models, it can be difficult to use for generative networks.

Mo et al [20] suggested using the "FreezeD" strategy, showing that freezing lower layers of the discriminator helps the model to achieve better results during the fine-tuning operation. This is done, since, intuitively, the lower layers of the discriminator learn generic features of images while the upper layers learn to classify whether the image is real or fake based on the extracted features.

In [21] the authors point out that a pretrained discriminator is more critical than the generator. Nevertheless, during this project, we explored how much the results can be improved with only a pretrained generator provided.

III. AttGAN

As previously mentioned, our starting point for this project is AttGAN. As described in [1], AttGAN is composed of two main components: the generator and the discriminator. The details on each layer of the two components are reported in Table I, while the overall structure of the network is shown in Figure 5. In the following section, we describe the design of AttGAN, the role of each component and how the training of the network works. Finally, we outline the approach we followed to adapt AttGAN to achieve our goal: performing single-attribute editing to add realistic eyeglasses to the input images efficiently in terms of training data and resources required.

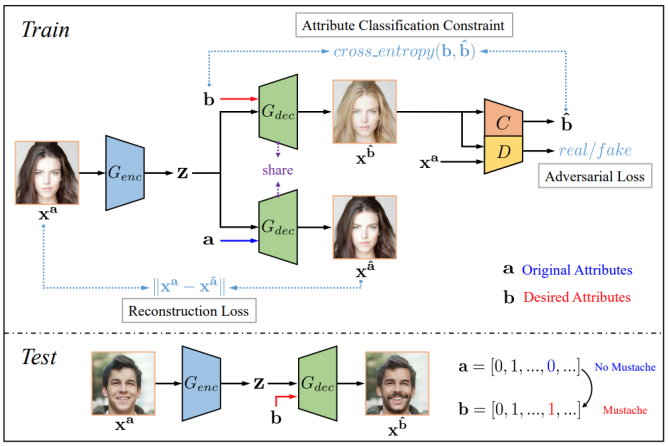

Fig. 5: AttGAN [1] architecture. In our experiments, the desired attribute is "eyeglasses".

A. Architecture components

The generic task we want to solve can be summarised as follow: given an input image x_a labeled with n binary attributes a=[a_1, a_2, ..., a_n], we want to generate an output image x_b with n binary attributes b=[b_1, b_2, ..., b_n].

The generator network (G) is composed by an encoder (G_{enc}) and a decoder (G_{dec}). The encoder gets as input x^a and encodes it into the latent representation $\mathbf{z}$:

$$\mathbf{z} = G_{enc}\left(\mathbf{x^a}\right) \tag{1}$$

Then, to get the output image, z is decoded conditioned on b by the decoder.

$$\mathbf{x}^{\hat{b}} = G_{dec}(\mathbf{z}, \mathbf{b}) \tag{2}$$

Thus the whole editing process is formulated as

$$\mathbf{x}^{\hat{b}} = G_{dec}\left(G_{enc}\left(\mathbf{x^a}\right), \mathbf{b}\right) \tag{3}$$

The work of the generator is unsupervised since the ground truth is not available, but we expect the output will comply with the following constraints:

1) being visually realistic
2) having the desired attributes b
3) keeping intact the other details of the input image x_a

To force the first requirement during training, an adversarial learning process is adopted between the generator and a discriminator (D). D predicts whether the output image of the generator is real or fake. In this way, the generator will be forced to produce more and more real-looking images aiming to fool the discriminator.

To check the correctness of the output image attributes, an attribute classifier (C) is employed, whose role is to predict the presence or absence of each of the n desired attributes.

GENERATOR		DISCRIMINATOR	
Encoder (G_{Enc})	**Decoder** (G_{Dec})	**Discriminator** (D)	**Classifier** (C)
Conv(64,4,2), BN, Leaky ReLU	DeConv(1024,4,2), BN, Leaky ReLU	Conv(64,4,2), LN/IN, Leaky ReLU	
Conv(128,4,2), BN, Leaky ReLU	DeConv(512,4,2), BN, Leaky ReLU	Conv(128,4,2), LN/IN, Leaky ReLU	
Conv(256,4,2), BN, Leaky ReLU	DeConv(256,4,2), BN, Leaky ReLU	Conv(256,4,2), LN/IN, Leaky ReLU	
Conv(512,4,2), BN, Leaky ReLU	DeConv(128,4,2), BN, Leaky ReLU	Conv(512,4,2), LN/IN, Leaky ReLU	
Conv(1024,4,2), BN, Leaky ReLU	DeConv(3,4,2), Tanh	Conv(1024,4,2), LN/IN, Leaky ReLU	
		FC(1024), LN/IN, Leaky ReLU	FC(1024), LN/IN, Leaky ReLU
		FC(1)	FC(13), Sigmoid
43.4 M parameters		44.7 M parameters	

TABLE I: AttGAN [1] architecture for 128x128 images. Conv(d,k,s) and DeConv(d,k,s) denote convolutional layer and transposed convolutional layer with d as dimension, k as kernel size and s as stride. BN stands for batch normalization, LN for layer normalization and IN for instance normalization. In the last row of the table are reported the total number of parameters for the generator and for the discriminator.

Finally, to ensure that the decoder outputs an image that changes the desired attributes, but also preserves other details, a reconstruction learning constraint is used. Thus, we enforce that, given x_a as input to the encoder, it produces a latent representation **z**, that when conditioned by the original attributes **a**, allows the decoder to generate an image that can be approximated by x_a itself. This can be summarised as:

$$\mathbf{x}^{\widehat{\mathbf{a}}} = G_{dec}(\mathbf{z}, \mathbf{b}), \mathbf{x}^{\widehat{\mathbf{a}}} \rightarrow \mathbf{x}^{\mathbf{a}} \tag{4}$$

In this way, we ask the encoder to preserve, in the latent representation **z**, enough information of the attribute-excluding details, enabling the decoder to actually restore them. The network will then transfer the detail preservation ability from the face reconstruction task to the attribute editing task.

In Table I, it can be noticed that the classifier and the discriminator share most of the architecture. The only difference is on the last layer: the discriminator produces one only output (real/fake), while the classifier needs to predict one binary value for each attribute (13 in our experiments).

B. Losses

Now we consider the loss functions used during the training of the generator and the discriminator.

Attribute classification Loss. To enforce the attribute classification constraint we use the classifier C. C gets as input the produced output image of the generator $x^{\widehat{b}}$ and predicts its attributes $\widehat{b}$. The generator needs to produce an image with attributes b so we apply a cross-entropy loss comparing the expected attributes b and the actual attributes $\widehat{b}$:

$$\min_{G_{enc}, G_{dec}} \mathcal{L}_{cls_g} = \mathbb{E}_{\mathbf{x}^{\mathbf{a}} \sim p_{data}, \mathbf{b} \sim p_{attr}} \left[\ell_g(\mathbf{x}^{\mathbf{a}}, \mathbf{b}) \right] \tag{5}$$

$$\ell_g(\mathbf{x}^{\mathbf{a}}, \mathbf{b}) = \sum_{i=1}^{N} -b_i \log C_i(\mathbf{x}^{\widehat{\mathbf{b}}}) - (\mathbf{1} - \mathbf{b_i}) \log(\mathbf{1} - \mathbf{C_i}(\mathbf{x}^{\widehat{\mathbf{b}}})) \tag{6}$$

Where p_{data} indicates the distribution of the real images, p_{attr} the distribution of the attributes, $C_i(\mathbf{x}^{\widehat{\mathbf{b}}})$ the prediction of the classifier C on the i_{th} attribute. Then, to train the classifier itself we feed the C with the input images x_a and we compare its prediction with the original attributes a.

$$\min_{C} \mathcal{L}_{cls_c} = \mathbb{E}_{\mathbf{x}^{\mathbf{a}} \sim p_{data}} \left[\ell_r \left(\mathbf{x}^{\mathbf{a}}, \mathbf{a} \right) \right] \tag{7}$$

$$\ell_r(\mathbf{x}^{\mathbf{a}}, \mathbf{a}) = \sum_{i=1}^{N} -a_i \log C_i(\mathbf{x}^{\mathbf{a}}) - (1 - a_i) \log(1 - C_i(\mathbf{x}^{\mathbf{a}})) \tag{8}$$

Reconstruction Loss. Furthermore, the reconstruction loss is introduced to preserve attribute-excluding details. The goal is to make the decoder learn how to reconstruct the original image x^a, having as input the latent space z and the original attributes a. The learning objective is formulated as

$$\min_{G_{enc}, G_{dec}} \mathcal{L}_{rec} = \mathbb{E}_{\mathbf{x}^{\mathbf{a}} \sim p_{data}} \left[\left\| \mathbf{x}^{\mathbf{a}} - \mathbf{x}^{\hat{a}} \right\|_1 \right] \tag{9}$$

Adversarial Loss. The adversarial learning process between the generator and the discriminator is used to enforce a visual realistic constraint on the generated images. In our experiments we used, following WGAN [17], as adversarial losses:

$$\min_{\|D\|_{\llcorner} \leq 1} \mathcal{L}_{adv_d} = -\mathbb{E}_{\mathbf{x}^{\mathbf{a}} \sim p_{data}} D\left(\mathbf{x}^{\mathbf{a}}\right) + \mathbb{E}_{\mathbf{x}^{\mathbf{a}} \sim p_{data}, \mathbf{b} \sim p_{attr}} D(\mathbf{x}^{\widehat{\mathbf{b}}}) \tag{10}$$

$$\min_{G_{enc}, G_{dec}} \mathcal{L}_{adv_g} = -\mathbb{E}_{\mathbf{x}^{\mathbf{a}} \sim p_{data}, \mathbf{b} \sim p_{attr}} [D(\mathbf{x}^{\widehat{\mathbf{b}}})] \tag{11}$$

optimized via WGAN-GP [18].

Overall Objective. The resulting loss function is obtained by combining the attribute classification constraint, the reconstruction loss and the adversarial loss, as can be seen in the following:

$$\min_{G_{enc}, G_{dec}} \mathcal{L}_{enc,dec} = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{cls_g} + \mathcal{L}_{adv_g} \tag{12}$$

C. Proposed modifications

In order to finetune AttGAN to our needs, we propose two training approaches that we call *Inverse* and *Alternate*.

Inverse Training. In the original training approach, the target attributes are obtained by randomly permuting the attributes' vectors of images belonging to the same batch. This approach guarantees a good level of generalization but makes the task really complex for the generator.

Since we are only interested in changing one attribute, we propose a training approach where the presence or absence of the target attribute is simply inverted, hence the name, while other attributes are kept the same. In other words, the generator simultaneously learns to add and remove the target attribute only, depending on its original value. In our case, the generator is asked to add eyeglasses to faces without them and to remove

eyeglasses from faces with them. This is the approach used in all of our experiments.

Alternate training. The second technique, which we called *Alternate* consists in:

- Generator training: the network receives only images without the target attribute to make the generator learn to add it.
- Discriminator training: the network receive only images with the target attribute to make the discriminator learn the characteristics of real images.

This technique has been used in some experiments explained in the next sections.

IV. Implementation details

A. Network

For the purpose of our analysis, we referred to the PyTorch implementation of AttGAN [22], integrating it within the PyTorch Lightning framework [23].

No major modifications were made to the original architecture.

We first tested the effects of varying the number of shortcut layers by training from scratch. Then, we promptly switched our focus to the finetuning part of the project, where we took advantage of the pretrained generator weights, provided by the AttGAN's authors, as our starting point.

We also explored the possibility of pretraining the discriminator (or rather, aligning it), because of the lack of a pretrained one. We achieved this by freezing the pretrained generator and evaluating the performance of the discriminator only. Since this is a supervised task (because we know which images are real and which ones are fake) we exploit the classical training-validation method using discrimination accuracy as a metric. Also, to stabilize the fine-tuning we used the FreezeD [20] strategy, which consists in simply freezing some of the lowest layers of the discriminator.

B. Metrics

Because of the nature of the task at hand, we preferred evaluating the results qualitatively by means of visual inspection, as mainly done also by the original AttGAN authors in [1].

Quantitative feedback can still be obtained by looking at the losses and their trends, with particular attention to the reconstruction loss (for overall image quality and details preservation) and to the generator loss (for the correct generation and classification of the desired attributes).

C. Training details

The batch size was set to 128 and the starting learning rate was set to 0.0001. The maximum number of epochs was 30 and we used early stopping on the generator loss with patience of 20 steps.

The Adam optimizer was used, with $\beta_1 = 0.5, \beta_2 = 0.999$. The default hyper-parameters for the loss are: $\lambda_1 = 100$, $\lambda_2 = 10$, $\lambda_3 = 1$, as per the original implementation.

The discriminator-to-generator ratio was set to 3, which means that the generator is only trained for one step every 3 steps of the discriminator. The learning rate warmup was set such that the target initial learning rate would be reached at the end of the first epoch for the discriminator, and at the end of the second epoch for the generator. These settings were chosen in accordance with the idea that the discriminator should be trained a little bit more, and that the generator had already been pretrained, so we try to avoid drastic modifications to it.

At the end of each training epoch, we qualitatively evaluate the results on a small set of images.

D. Dataset

The dataset used in this project is CelebFaces Attributes (CelebA) [24]. The original dataset contains more than two hundred thousand images annotated with forty binary attributes that specify the presence (value 1) or the absence (value -1) of a certain facial characteristic.

Due to the hardware limitations and the simplified task (learning only to modify a single attribute), we reduced the total size of the training set by formulating two different subsets:

- *Eyeglasses only*, composed only by images with the presence of eyeglasses, around 11000 samples;
- *Eyeglasses extended*, composed by 1/3 of images with eyeglasses and 2/3 randomly sampled from within the remaining images, for a total of around 33000 samples.

A quick look at the attributes' distribution tells us that the original dataset is not balanced (Table II): some attributes, such as "No Beard" and "Young", belong to more than 70% of the images, while other attributes are associated with at most 40% of the images. As a matter of fact, a consistent number of attributes ("Mustache", "Bald" and "Goatee" for example) are present in less than 10% of the total faces.

From Table II it is possible to see the different distributions of attributes obtained for the different subsets. The images with eyeglasses are mostly males without beard, but by simply introducing some random samples from the rest of the dataset, the "Male" attribute is better re-balanced and the "No_Beard" attribute increase its relevancy.

As originally done by He et. al. [1], the model was trained on a subset of the original attributes, choosing 13 characteristics with strong visual impact: "Bald", "Bangs", "Black Hair", "Blond Hair", "Brown Hair", "Bushy Eyebrows", "Eyeglasses", "Male", "Mouth Slightly Open", "Mustache", "No Beard", "Pale Skin" and "Young".

Moreover, during training, each image is preprocessed adopting the same transformations applied in the original AttGAN implementation [1]: centre cropping with output size 170×170, thus applying padding on the border of the images, then resizing to 128×128, and finally normalizing with mean 0.5 and standard deviation 0.5 for each RGB channel.

	Complete		Eyeglasses extended		Eyeglasses only	
	attribute	percentage↓	attribute	percentage↓	attribute	percentage↓
1	No_Beard	0.83	No_Beard	0.79	Eyeglasses	1.00
2	Young	0.77	Young	0.67	Male	0.79
3	Attractive	0.51	Male	0.52	No_Beard	0.68
4	Mouth_Slightly_Open	0.48	Mouth_Slightly_Open	0.48	Mouth_Slightly_Open	0.47
5	Smiling	0.48	Smiling	0.46	Big_Nose	0.46
6	Wearing_Lipstick	0.47	High_Cheekbones	0.41	Young	0.42
7	High_Cheekbones	0.46	Attractive	0.39	Smiling	0.40
8	Male	0.42	Wearing_Lipstick	0.36	High_Cheekbones	0.28
9	Heavy_Makeup	0.39	Eyeglasses	0.33	Black_Hair	0.21
10	Wavy_Hair	0.32	Big_Nose	0.30	Chubby	0.21

TABLE II: The top-10 attribute in subsets, ordered by descending *percentage*. *percentage* indicates the percentage of images having the attribute set to 1 in the subset.

V. Experiments

A. Varying the number of shortcut layers

First, we evaluate the effect of using a different number of shortcut layers (skip connections) inside the generator, as shown in Figure 6, by training the network from scratch. In accordance with [2], a larger number of shortcut layers guarantees a higher reconstruction quality, because the decoder layers receive more information from the encoder layers, but the attributes-modifying capability decreases, as the influence of the attributes' vector is less prominent.

For these reasons, we choose to use one shortcut layer, considering it a good trade-off. This was also the final decision made by the original PyTorch AttGAN implementation.

Thus, all the following experiments adopt 1 shortcut layer inside the generator.

B. Pretraining the discriminator

While keeping the pretrained generator frozen, we trained the discriminator such that it would "align" to the current state of the generator, until it finally reached a low loss value and good accuracy. To avoid overfitting, we adopted the original, generic training approach, where target attributes are obtained by means of random permutation, and we used a larger subset of the original dataset (around 70000 samples) chosen by means of random sampling.

Successive tests show that the generator benefits from being paired with a pretrained discriminator (figure 12), in terms of both quality and training stability.

C. Choosing the optimal subset

Due to the huge amount of images in the original dataset, we compared the effect of using the *Eyeglasses only* and *Eyeglasses extended* subsets: the results seem to favour the use of the more various dataset. This was foreseeable considering that we are using the Inverse training approach, so when having as input only images with eyeglasses the network never learns how to apply them to the faces.

In Figure 13 the generator loss is generally higher for the restricted dataset, while the discriminator loss is lower, so the discriminator is probably learning faster to distinguish fake from real leading the generator to worse results.

Given these results, all the following experiments make use of the *Eyeglasses extended* subset.

D. Finetuning the network

Finally, we compare the results obtained from the combination of the various finetuning techniques introduced so far.

FreezeD requires freezing some of the lowest layers of the discriminator, so we study how many layers should be frozen to achieve the best convergence. We can see from Figure 14 that freezing a high number of discriminator layers improves the general trend of the generator loss. Also, we can notice the overfitting is clear after training for some epochs. On the other hand, freezing 4 layers of the discriminator prevents the loss from increasing any more. Without FreezeD or with only a few layers frozen, the overfitting is more evident. Visually, FreezeD leaves at least some traces of the desired modification in later epochs. Without freezing any layer the output image seems to be a mere reconstruction of the original one (Figure 7) and there is no presence of the required eyeglasses. For these reasons, we decided to always freeze four layers of the discriminator.

Observing the trend of the other losses in the previously mentioned cases, we also noticed that the reconstruction loss seems to be the only one which is constantly decreasing, while the discriminator and the generator losses only reach a plateau. This happens while the process of manipulating the image attributes tends to degenerate, having no eyeglasses at all in some cases. Therefore, to address this problem we tried to reduce the weight of the reconstruction task, decreasing the value of λ_1. From figure 11, it can be seen how the drastic variation of λ_1 leads the model to great divergence in the first few epochs, but then starts to converge after epoch 5. High values of λ_1 (as 50), tends to converge and overfit faster than lower values (as 10 or 1). From Figure 8 it is possible to see that the modification appears more evident with $\lambda = 1$, which in turn affects a bit the reconstruction capability of the network.

Finally, we tested the effect of the *Alternate* method in combination with previous techniques, and the results in Figure 15 show a strong tendency of diverging. Visually (Figure 9), the images display a strong presence of eyeglasses, even for the most difficult faces, but they are clearly noisy, and the reconstruction capability of the network is also negatively affected as training goes on, leading the model to create very spoiled images.

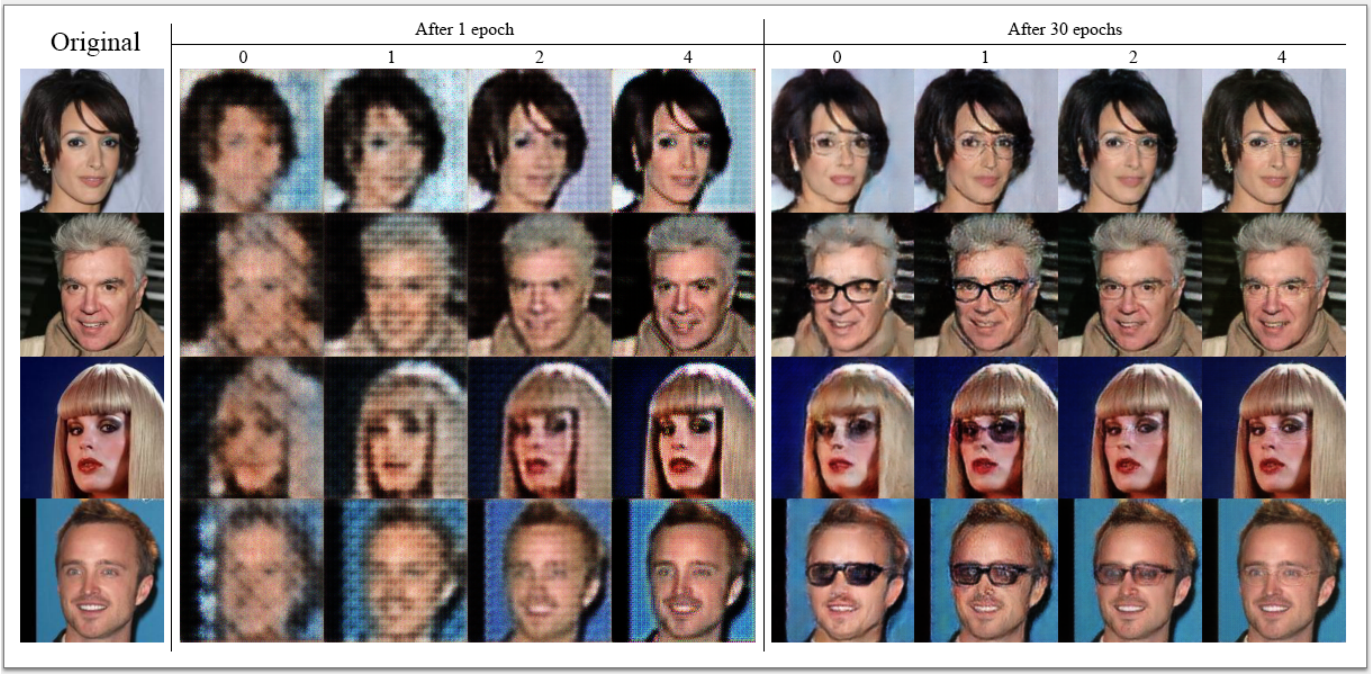

Fig. 6: Varying the number of shortcut layers. Here, 0, 1, 2 and 4 refer to the number of shortcut layers used inside the generator. From the images, it can be seen that increasing the number of skip connections improves image quality at the cost of weakened attribute manipulation ability.

(a) No FreezeD (b) FreezeD on 4 layers

Fig. 7: Effect of FreezeD on image modification.

(a) $\lambda_1 = 100$ (b) $\lambda_1 = 50$ (c) $\lambda_1 = 10$ (d) $\lambda_1 = 1$

Fig. 8: Effect of λ_1 on image modification.

E. Final comparison

From figure 10, it is possible to see that our finetuning approach generates an output image that is more coherent to the original one, making fewer modifications where not requested. The eyeglasses are well defined in some cases and less marked in other cases, but they overall seem more consistent in terms of visual appeal. These results were obtained using $\lambda_1 = 50$ and FreezeD applied to the first 4 layers.

(a) Early epoch (b) Late epoch

Fig. 9: Effect of *Alternate* training on image modification at different epochs.

VI. Conclusions and future works

In this project, we managed to slightly improve the results of a GAN, originally trained with a large and sparse dataset, in solving a more specific task, by exploiting the combination of finetuning techniques and hyper-parameters optimization.

The usage of our Inverse training approach and FreezeD proved to be effective despite the simplicity of their formulation. Overall, the goal of obtaining a neural network capable of mimicking the behaviour of a social media filter can be considered achieved.

The next steps could be to find better ways to pretrain the discriminator, in order to improve results during the following finetuning. Moreover, a training strategy similar to the *Alternate* method could be proposed in a revised formulation that prevents the generator from diverging as training goes on.

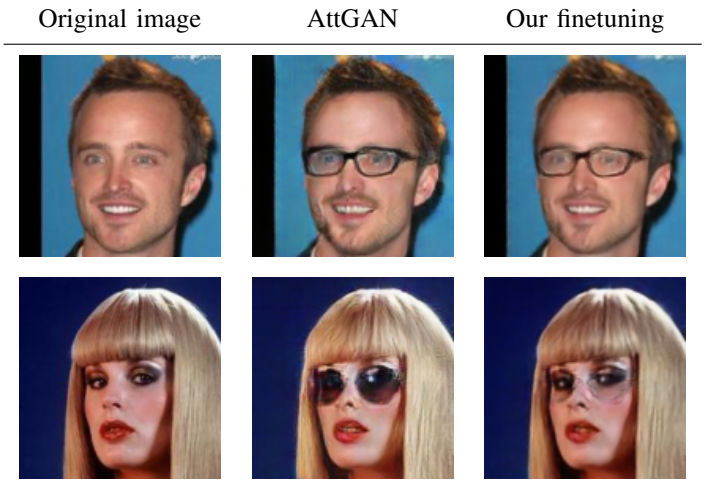

Fig. 10: Effect of "Eyeglasses" modification in AttGAN and finetuned AttGAN

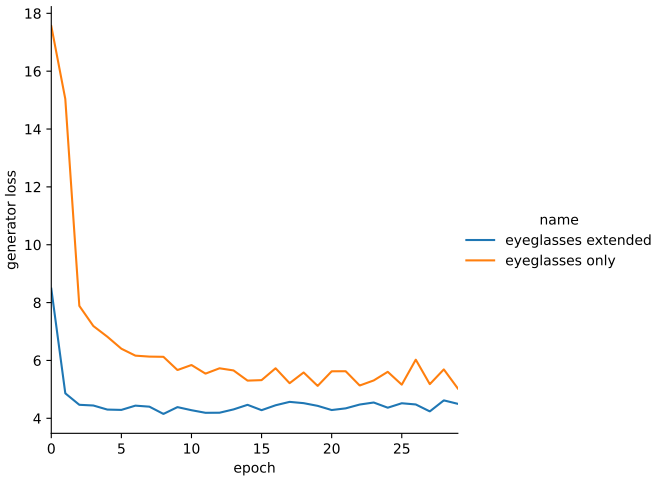

Fig. 13: Impact of different subsets on generator loss

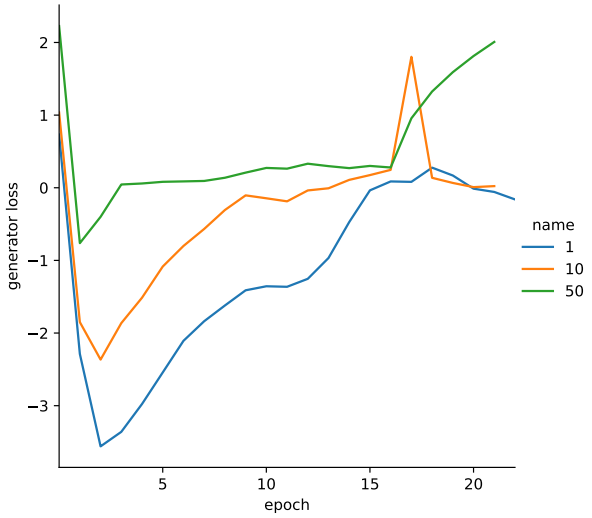

Fig. 11: Impact of the different weight of reconstruction loss on generator loss. *name* indicates the value of λ_1.

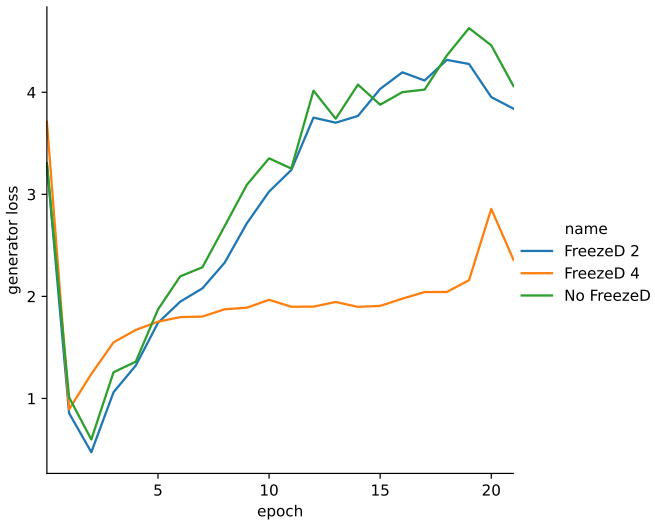

Fig. 14: Impact of FreezeD on generator loss. *FreezeD N* indicates the first N layers of the discriminator are frozen.

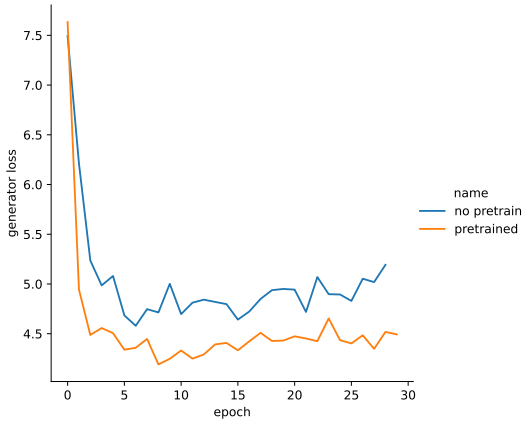

Fig. 12: Impact of pretraining discriminator on generator loss. It can be noticed the presence of an initial loss fall. This happens due to the arrangement of the learning rate from the Adam optimizer.

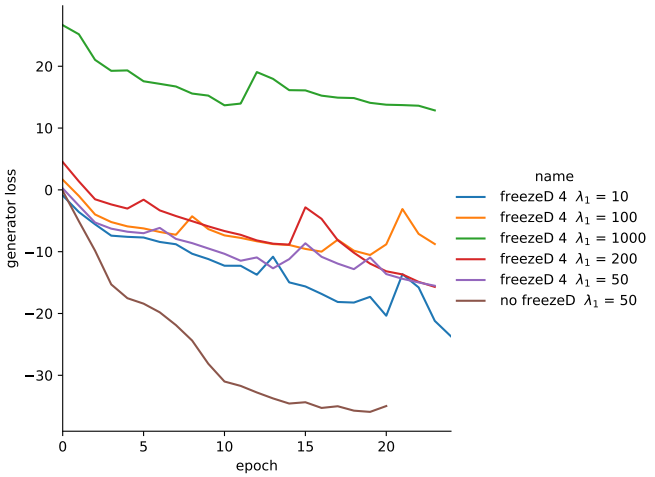

Fig. 15: Impact of the different weights of reconstruction loss and FreezeD on generator loss using *Alternate* training.

REFERENCES

[1] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Attgan: Facial attribute editing by only changing what you want," 2017. [Online]. Available: https://arxiv.org/abs/1711.10678

[2] M. Liu, Y. Ding, M. Xia, X. Liu, E. Ding, W. Zuo, and S. Wen, "STGAN: A unified selective transfer network for arbitrary image attribute editing," *CoRR*, vol. abs/1904.09709, 2019.

[3] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8789–8797.

[4] P. Upchurch, J. R. Gardner, K. Bala, R. Pless, N. Snavely, and K. Q. Weinberger, "Deep feature interpolation for image content changes," *CoRR*, vol. abs/1611.05507, 2016.

[5] S. Zhou, T. Xiao, Y. Yang, D. Feng, Q. He, and W. He, "Genegan: Learning object transfiguration and attribute subspace from unpaired data," *CoRR*, vol. abs/1705.04932, 2017.

[6] W. Shen and R. Liu, "Learning residual images for face attribute manipulation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1225–1233.

[7] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013. [Online]. Available: https://arxiv.org/abs/1312.6114

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[9] G. Perarnau, J. van de Weijer, B. C. Raducanu, and J. M. Álvarez, "Invertible conditional gans for image editing," *CoRR*, vol. abs/1611.06355, 2016.

[10] G. E. Hinton and R. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in Neural Information Processing Systems*, J. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan-Kaufmann, 1993. [Online]. Available: https://proceedings.neurips.cc/paper/1993/file/9e3cfc48eccf81a0d57663e129aef3cb-Paper.pdf

[11] "From autoencoder to beta-vae," https://lilianweng.github.io/posts/2018-08-12-vae/.

[12] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf

[13] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[14] K. Zhang, Y. Su, X. Guo, L. Qi, and Z. Zhao, "MU-GAN: facial attribute editing based on multi-attention mechanism," *CoRR*, vol. abs/2009.04177, 2020.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[16] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.06434

[17] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf

[19] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: training imagenet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017.

[20] S. Mo, M. Cho, and J. Shin, "Freeze the discriminator: a simple baseline for fine-tuning gans," 2020. [Online]. Available: https://arxiv.org/abs/2002.10964

[21] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu, "Transferring gans: generating images from limited data," 2018. [Online]. Available: https://arxiv.org/abs/1805.01677

[22] E. Y.-J. Lin, "https://github.com/elvisyjlin/attgan-pytorch."

[23] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," 2019. [Online]. Available: https://github.com/Lightning-AI/lightning

[24] 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)*, 12 2015.