Fader Networks : Manipulating Images by Sliding Attributes

Christon Eliot ELIOT.CHRISTON.1@ETU.SORBONNE-UNIVERSITE.FR
Master Systèmes Avancés et Robotiques
Sorbonne Université
Paris, France

Lévêque Robin ROBIN.LEVEQUE.2@ETU.SORBONNE-UNIVERSITE.FR
Master Systèmes Avancés et Robotiques
Sorbonne Université
Paris, France

Magoudi Théo THEO.MAGOUDI.1@ETU.SORBONNE-UNIVERSITE.FR
Master Systèmes Avancés et Robotiques
Sorbonne Université
Paris, France

Pétard Adrien ADRIEN.PETARD@ETU.SORBONNE-UNIVERSITE.FR
Master Systèmes Avancés et Robotiques
Sorbonne Université
Paris, France

Editor: Machine Learning Avancé (2023-2024)

Abstract

This report presents our work on reproducing the results of the paper Lample et al. (2018). The paper presents a way of manipulating images by sliding attributes. We implemented our own version of the algorithm and we were able to reproduce the results of the paper qualitatively. We tried to stick as close as possible to the original paper, even though the implementation details were not always clear. Inspiration was also taken from hardikbansal (2017), Goodfellow et al. (2016) and Huang (2019). We used the CelebA dataset, preprocessed the images and trained a model to encode and decode the images in order to manipulate them. We have been able to produce results that come close to those presented in the article.

Keywords: Fader Networks, CelebA database, Image manipulation, Adversarial Learning

GitHub repository of the project : `https://github.com/RobLevv/MLA_Projet_2023`

1. Introduction

In a world dominated by visual media and rapid technological advancements, the ability to manipulate images with precision and creativity has become an indispensable asset. Enter the realm of "Fader Networks : Manipulating Images by Sliding Attributes," a groundbreaking approach that transcends traditional image editing boundaries.

As we navigate our digitally interconnected lives, the demand for personalized and dynamic content has surged. Imagine the power to effortlessly transform a photograph, not just in terms of color or composition, but by sliding attributes like age, gender, and expression with a fluidity reminiscent of adjusting sliders on an audio mixing console. This is the promise and potential of Fader Networks — a revolutionary leap in image manipulation that aligns seamlessly with the needs and expectations of our contemporary visual landscape.

Our project aims to harness the possibilities unlocked by Fader Networks and explore the direct applications in today's world. From reshaping online identities to enhancing creative expression, the implications are vast and compelling. As we delve into this venture, we are not merely exploring an abstract concept but stepping into the forefront of a technological wave that is reshaping the way we perceive and interact with visual content.

In an era where customization is king and visual storytelling is paramount, Fader Networks emerge as a game-changer. No longer confined by the limitations of traditional editing tools, we are venturing into a space where images can be manipulated effortlessly, empowering users to tell their stories with a level of precision and personalization that was once unimaginable.

As we introduce and implement this cutting-edge technology, we're not just manipulating pixels ; we're shaping the way we perceive and interact with visual narratives in our dynamic, visually-driven world.

2. Algorithm presentation

In this section, we will present various aspects of our algorithm. We will delve into the key techniques we have implemented and highlight the distinctions from the original algorithm. The code will be presented in 5 parts : Data processing, Pre-training, Loss implementation, Training, and Inference.

Data processing : First, we load images from the folder Img_processed, which contains pre-processed images from the CelebA dataset. We ensure that all loaded images have the correct shape (3, 256, 256), and the y vector of attributes containing -1 and 1 values is transformed to 0 and 1 values. We decide to create different preprocessing functions, cropping and resizing the images, as well as the train-test split and the plotting of images.

thaning : We have a `network_architectures.py` file that defines neural network architectures consisting of three main components :

1. **Encoder** : Comprising 7 convolution layers, it transforms an image with a shape of (3, 256, 256) into a latent space with a shape of (512, 2, 2).
2. **Decoder** : Also composed of 7 convolution layers, it transforms a latent space of shape (512+N, 2, 2) into an image with a shape of (3, 256, 256). Here, we use 512 + N filters containing y attributes instead of the originally proposed 512 + $2N$ filters.
3. **Discriminator** : The discriminator consists of a C512 layer followed by a fully-connected neural network with two layers of sizes 512 and n, respectively.

In our implementation, we incorporate two dropout layers with a probability of 0.20, deviating from the recommended 0.30 in the paper, as it yielded better results. Additionally, we use the sigmoid activation function instead of tanh to ensure that the results are bounded between 0 and 1.

Then we define two Python files : `autoencoder.py` and `discriminator.py`. The former introduces an `Autoencoder` class that utilizes the encoder and decoder layers specified in

`network_architectures.py`. The `forward` method processes an image (`x`) and attributes (`y`). It traverses the encoder layers to obtain the latent representation (`x -> latent`). Subsequently, the latent space is concatenated with the attributes, and the result undergoes the decoder layers to produce the decoded image (`latent, y -> decoded`). To concatenate the latent space and attributes, we unsqueeze twice and expand the values to match the shape of the latent space, resulting in 512 + N filters for the input of the decoder.

The latter file defines a `Discriminator` class for adversarial objectives using the discriminator layers from `network_architectures.py`. The `forward` method takes the latent space as input and outputs a prediction of the attributes of an image (`latent -> y_pred`).

Loss Implementation : This module contains functions responsible for computing the losses associated with the three primary objectives of the project. The reconstruction loss (`x, decoded -> reconstruction_objective`) is calculated using the classic Mean Squared Error (MSE).

$$L_{\text{AE}}(\theta_{\text{enc}}, \theta_{\text{dec}}) = \frac{1}{m} \sum_{(x,y)\in D} \|D_{\theta_{\text{dec}}}(E_{\theta_{\text{enc}}}(x), y) - x\|_2^2 \tag{1}$$

Next, the discriminator loss (y, y_pred -> discriminator_objective) is computed using binary cross-entropy, a substitution for logarithmic losses due to their similarity. Cross-entropy is favored for its ease of implementation, aligning with the paper implementation.

$$L_{\text{dis}}(\theta_{\text{dis}}|\theta_{\text{enc}}) = -\frac{1}{m} \sum_{(x,y)\in D} \log P_{\theta_{\text{dis}}}(y|E_{\theta_{\text{enc}}}(x)) \tag{2}$$

Lastly, the adversarial loss is the difference between the reconstruction loss and the opposite of the discriminator output.

$$L(\theta_{\text{enc}}, \theta_{\text{dec}}|\theta_{\text{dis}}) = \frac{1}{m} \sum_{(x,y)\in D} \|D_{\theta_{\text{dec}}}(E_{\theta_{\text{enc}}}(x), y) - x\|_2^2 - \lambda_e * \log P_{\theta_{\text{dis}}}(1 - y|E_{\theta_{\text{enc}}}(x)) \tag{3}$$

Training : We first create a `Logs` folder to generate logs in a specified directory for each new training, in order to display and save results of our training. We create an `add` method to log various information during training. This method creates a file detailing the parameters and the architecture of the model, three other files containing the values of the losses for each training step, and finally, the decoded image at the end of each epoch to visualize the results of the current model training.

Then we create a `train_loop.py` file which contains the main loop to train our model. It defines a function `train_model` that will iterate over the specified number of epochs. Every epoch, we iterate over the batches. It loads the batch of images with the dataloader. It ensures that the images have the correct shape for training and sends them to the GPU if available. Going through the autoencoder and the discriminator, it generates the latent space, the decoded image, and the y predictions. Finally, we compute the model losses and update the autoencoder and discriminator weights based on adversarial and discriminator objectives. This part was optimized by preprocessing the images before the training, and it can be further optimized by saving the images in a format that does not need processing after loading.

Inference : Lastly, we create an `inference.py` file to make predictions on a new image using a trained model, an original image for modification, and newly modified attributes. This function returns the decoded image along with the discriminator's predictions in the latent space.

3. Dataset presentation

3.1 CelebA Dataset

In our implementation, we employed the CelebA dataset to train and evaluate the Fader Networks, consistent with the methodology outlined in the original paper. The CelebA dataset is a widely utilized resource in the domain of facial attribute recognition, comprising over 200,000 celebrity images annotated with 40 different attribute labels, including "Smiling," "Male," "Young," etc.

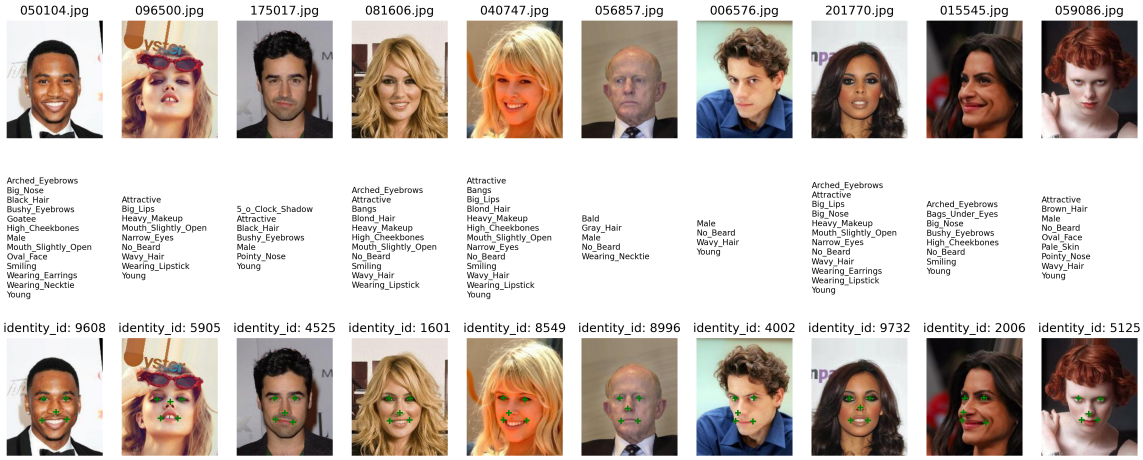

FIGURE 1 – Example images from the CelebA dataset

The dataset is partitioned into a training set of 162,770 images, a validation set of 19,867 images, and a test set of 19,962 images. All images are of size 178x218 pixels, with faces centered and aligned. For our experiments, we preprocessed the images into a separate dataset with square dimensions of 256x256 pixels, ensuring no additional transformation is necessary before feeding them into the network.

Depending on the experiment, we utilized different subsets of the CelebA dataset to reduce training time. A comprehensive data exploration notebook is available in the repository, providing detailed insights into the dataset. Here, we highlight some key aspects.

It's crucial to note that the CelebA dataset is unbalanced (see Figure 2), with certain attributes being more represented than others. This inherent bias may influence the training of the Fader Networks.

3.2 Etu Dataset

For additional evaluation, we constructed a dataset featuring the faces of contributors to this project. This dataset was exclusively used for inference to assess the Fader Networks on faces outside the CelebA dataset. All images in this dataset are of size 256x256 pixels, with centered and aligned faces.

3.3 Other Datasets

Our framework is adaptable to various datasets for training and evaluation, the only requirement being that the images are of size 256x256 pixels. The Fader Networks are not confined to facial images and can be applied to diverse image types. In the original paper, the authors demonstrated

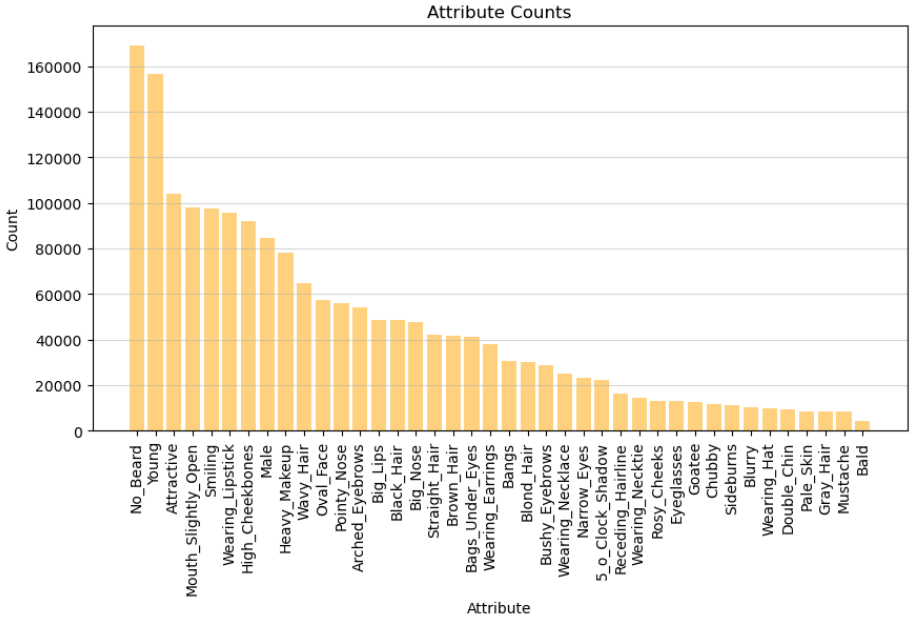

FIGURE 2 – Distribution of attributes in the CelebA dataset

this versatility by using a flower dataset and a building dataset. Given more time, we could explore training the Fader Networks on these alternative datasets.

4. Experimental evaluation

4.1 Experiment description

Our initial step involved training our model on a subset of 100 images from the CelebA dataset. This preliminary phase ensured the generation of analyzable results. We began by training only the autoencoder and the discriminator neural networks separately, which allowed us to verify the architecture's ability to reconstruct images from the latent space produced by the encoder and ensure the disriminator capability to learn.

The next step was to incorporate the adversarial component, which created a competitive dynamic between the autoencoder and the discriminator. The goal of this architecture is to enable the encoder to transform an image into a latent space that is invariant to the attributes present in the original image. This invariance allows the manipulation of the decoder by feeding it modified attributes, altering the decoded images at the output.

The adversarial component consists of the discriminator attempting to extract the attributes from the latent space. The more errors the discriminator makes, the more invariant the latent space becomes to the attributes. This process ensures the effective manipulation of images by sliding attributes.

We assessed the effectiveness of our architecture by visualizing the decoded images and monitoring the discriminator's loss. As soon as the images began to resemble human faces and the discriminator's loss approached zero, we used a larger dataset (approximately 160,000 images)

4.2 Results

We were able to obtain good results on the attribute sliding. The best results are on few attributes mainly, the most respresented in the dataset. For example, the mouth opening attribute has good results on those examples. In those two following sets, we get similar results, the original image is reproduced properly and the mouth opens gradually with the attribute.

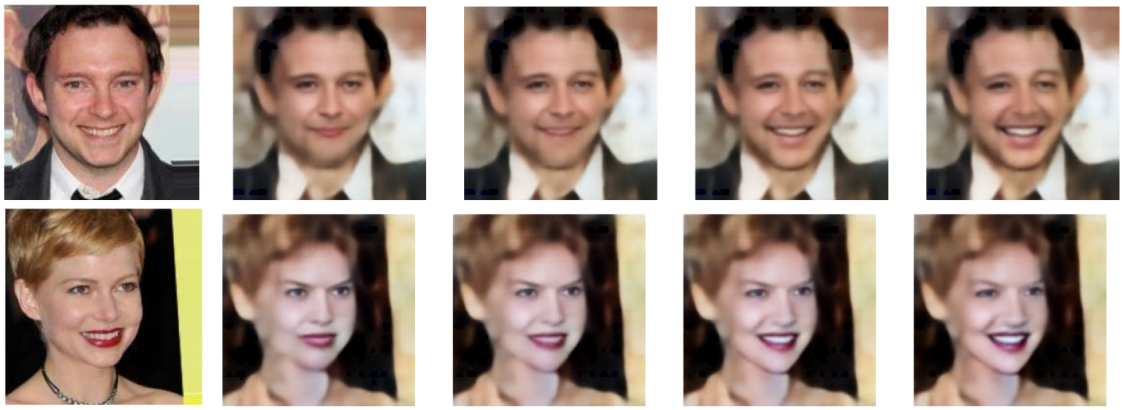

FIGURE 3 – Sliding attribute mouth opening on CelebA datset with our model

We can compare our results with the ones given in the article on the male/female transformation.

FIGURE 4 – Comparison between article and our model on the male/female attribute

We can see that our model was trained correctly, it can reconstruct an image with a shifted attribute quite correctly when compared to the article results. We also tried to make our model predict on other images to verify that it was not overfitting the images. We used our own faces and it predicted quite good results too.

FIGURE 5 – Eyeglasses and mouth opening attributes on our faces

4.3 Discussion

We got convincing results for some attributes on some original images. However, for some original images, depending on the quality, the light, the angle or the attribute sliding we encountered multiple bad results. The folowing image is an example of poor light and a lot of dark colors in the original image resulting on creating visual artefacts for some values of the attributes.

FIGURE 6 – Under exposure and complex background

Sometimes, the original image is not perfectly centered and the face is not in the angle mostly present in the dataset so the reconstruction gives a not very satisfying result.

FIGURE 7 – Inacurrate placement of the face

The dataset conditions the results we can obtain. We saw in the dataset presentation that the celebA dataset is unbalanced between the attributes making the generated images on the low represented attributes such as bald and mustache not very convincing. Furthermore, some attributes are often correlated. For example, a younger version of a man will have feminine traits appearing as well as a bearded version of a woman will tend to obtain traits of a man. This correlation can be explained by our architecture because we insert the attributes at the input of the decoder instead of at each layer.

5. Conclusion

In the pursuit of harnessing the capabilities of Fader Networks for image manipulation, our exploration has unveiled several noteworthy challenges and limitations.

During the training phase, a recurring issue surfaced in the form of decoded images exhibiting blurriness, with color discrepancies and pixel inaccuracies compared to the input images. This phenomenon indicates a potential limitation in the fidelity of the reconstruction process, raising questions about the algorithm's capacity to faithfully reproduce intricate details and nuances present in the original images.

Furthermore, the sensitivity of Fader Networks to the positioning and alignment of input images poses a significant constraint. Images that deviate from a centered composition or lack proper alignment of facial features, particularly the eyes, result in compromised training and decoding outcomes. This dependency on a specific input configuration may limit the practical applicability of the model in scenarios where obtaining perfectly aligned images is challenging.

Despite utilizing a substantial database for training, the obtained results still fall short of achieving complete satisfaction. The inadequacies in the generated images suggest potential limitations in the model's ability to generalize and capture the full complexity of attribute manipulations, even with a diverse and extensive dataset.

In the pursuit of our initial goal to compare results with the benchmark established in the referenced paper, it becomes evident that our outcomes did not fully meet expectations. The discrepancy in performance raises questions about the factors influencing the divergence, shedding light on the nuances involved in replicating and achieving comparable results with state-of-the-art models.

As we reflect on the challenges encountered, it is crucial to acknowledge the considerable time investment required for the training process. The protracted duration, notably influenced by the transformation of integer images to floats during data loading, highlights an operational bottleneck that warrants further exploration and optimization.

In conclusion, the project journey has illuminated critical aspects that warrant refinement and further investigation. Addressing challenges related to image fidelity, input sensitivity, dataset sufficiency, and training efficiency will be pivotal in advancing the effectiveness of Fader Networks for image manipulation. These insights provide valuable considerations for future research and development in this dynamic field.

Références

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

hardikbansal. Fader networks tensorflow implementation. https://github.com/hardikbansal/Fader-Networks-Tensorflow, 2017.

Eddie Huang. An intuitive understanding of fader networks. *Towards Data Science*, 2019. http://www.xavierdupre.fr/app/ensae_teaching_dl/helpsphinx/chapters/deep_generative_adversarial_network_gan.html.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader networks : Manipulating images by sliding attributes, 2018. https://arxiv.org/abs/1706.00409.