Report of the project
“JoJoGAN: One Shot Face Stylization“

Mattia Pannone 1803328

1. Introduction

This project was made by Min Jin Chong and D.A. Forsyth at University of Illinois at Urbana-Champaign and concerns about transfer the style from a reference style image to another input image, in particular for the faces. In their work the authors used the Neural Networks and in particular the GAN (Generative Adversarial Networks) for the style transfer task, in particular to learn a style mapper and apply it on input images to map the style of the reference images, all this with a single style image reference.
With respect to the state of the art for this kind of task, this work produce a paired dataset using GAN Inversion, StyleGAN’s style-mixing property and a pixel-level loss, so it is capable to produce images preserving input pose, expression and identity and tend to to capture distinct style details and diversity in a better way.
This project use 4 steps: a GAN inversion to invert a style reference image; the production of a training set; a fine-tuning step to fine-tune the StyleGan; final the inference step to produce the output.
In my work, I re-organized the functions to load generator and discriminator models, elaborate the images to use, train the GAN on a new reference frame, produce the output. Next I produced the same output of the paper, loading pre-trained models on different styles and plotting the output. Second I trained the network on new reference style images taken from the web. For all the output I used new images different from those used in the paper, in particular the image of an actor, an image of me and a set of image generated randomly thanks to the generator network. I also plotted the loss curve of the training process.
In the google colab used, I used data and classes from original git, I don’t re-writed the classes from scratch for its complexity, but I putted at the end the notebook the classes of generator and discriminator to show their structure.

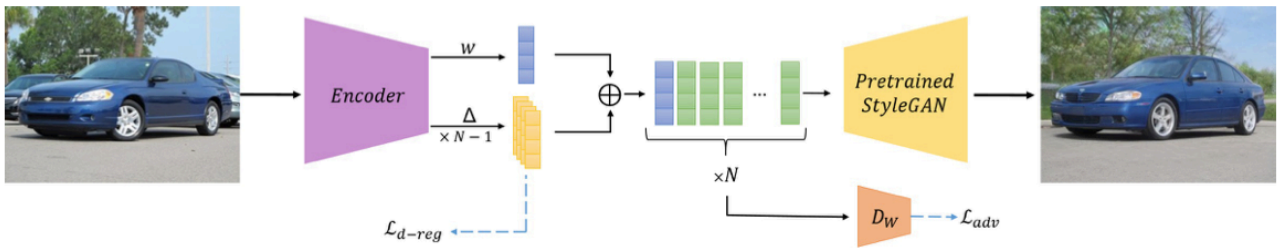

Fig. 1

2. Technologies used

The project is based on the GANs, that stay for Generative Adversarial Networks, introduced by Ian Goodfellow in 2014. The concept behind this kind of networks is to produce new data as much as possible to real data (they can include images, videos, music, natural languages), learning their distribution. In order to do this, GANs train simultaneously two neural nets: a generator G (a decoder) to produce new data (fake data) and a discriminator D that take data in input and judge if they are real or fake (so it is a binary classifier). The name Adversarial Networks is because G and D play opposite games, the first have to minimize the error to produce data as much as possible near to the reality, the second has to maximize the capability of distinguish the real data from fake ones. Generator is good when discriminator is not able to recognize the fake data from the real ones.
In the first step of this work, GANs are used for the GAN inversion, that is invert a given image to obtain it in a latent space i.e. retrieve a latent code from which it is possible reconstruct the original image. The pre-trained net used is “e4e” [Fig. 1], that stay for “Encoder for Editing”, a novel encoder that is specifically designed to allow for the subsequent editing of inverted real images.

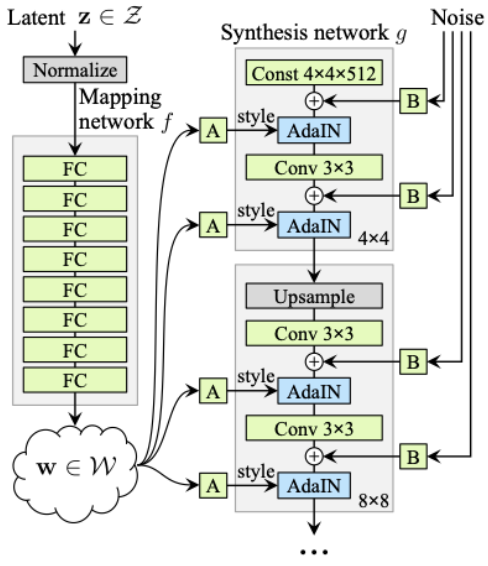

Fig. 2

The other GAN used is StyleGAN [Fig. 2], that is a styled-based GAN architecture, it yields state-of-the-art results in data-driven unconditional generative image modeling and it is capable to affine transforms then produce styles that control the layers of the synthesis network via adaptive instance normalization (AdaIN). In order to do experiments, I have used both pre-trained model and new trained model.

3. Work steps

In order to produce new stylized images, the project follow 4 steps: the first is the GAN Inversion, where we obtain a style code “w” from a reference style image thanks to e4e net and from this style code, given in input to a StyleGAN net, we can obtain a real image; the second step is to produce a training set from the previous style code, in fact using the StyleGAN’s style mixing mechanic we find a set of style codes W that are “close” to w; the third step is the training procedure where we fine-tune the StyleGAN to obtain the optimized weights for the network capable to produce new real stylized output; the last step is to do inference on an input and produce the output. This last step is done in two ways: first is given as input one or more real face images (of real person), the the style is applied on random stylized samples by sampling random noise and generating with our fine-tuned StyleGAN.
To deal with the output, the original work use two techniques to control the style: one is to use a function $f = (1 - \alpha)f_i^A + \alpha f_i^B$ where α is the interpolation factor, applied to the latent space to control the style intensity, increasing α results in stronger style intensity. The other method is for controlling aspect of style, this is made by applying a mask to the input to choose if preserve colors of the input or ensure that color is driven by the style example. [Fig. 3] show this steps.

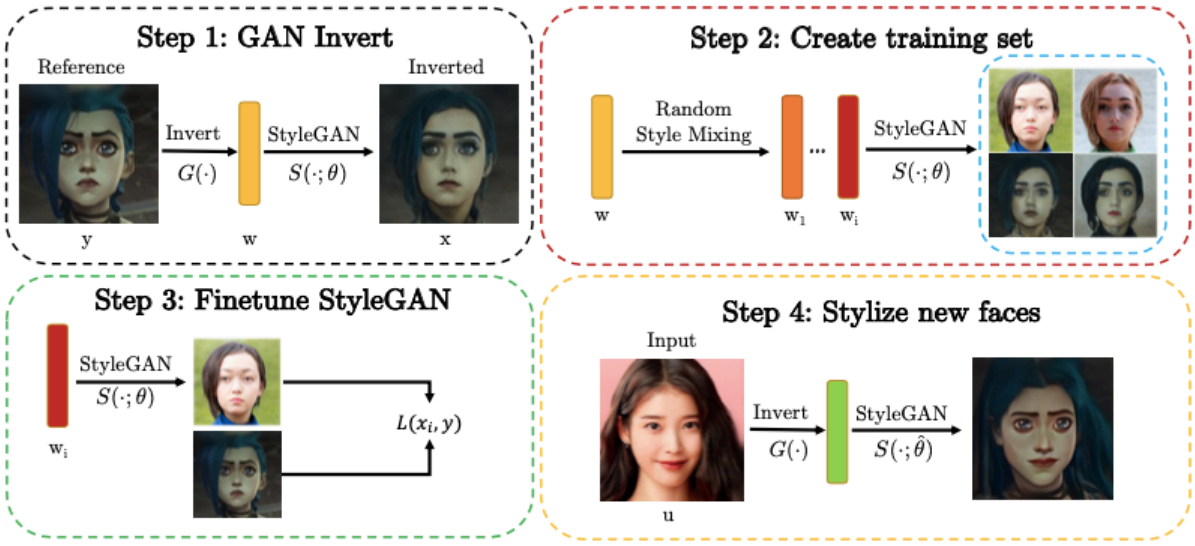

Fig. 3

4. My personal work

Most of the original work is available on git hub from where I downloaded major part of material to execute and reproduce the output. In particular I re-organized all the functions needed and I reproduced the output first with pre-trained model and then training the model on new reference style images, using as input both real faces and random generated ones.
The functions I re-writed using the available code are: *load_generator* and *load_discriminator* to load generator and discriminator from the pre-trained stylegan2 (a new version of the original StyleGAN) model; *elaborate_image t*o load an image (both for reference and input images) to localize the face and produce a latent code form the GAN inversion with the e4e net; *compute_output_on_given_image* and *compute_output_on_random_images* both for produce the output stylized image, the first one loading an existing image, the second one using first the generator of StyleGAN to generate a set of random images and then applying the generator fine-tuned of the StyleGAN with the reference image; *load_new_reference* to load a new (or a set of news) reference style image and obtain the latent code to use for a new training; *train_on_new_reference* to train a new model on new reference style image; *plot_train_trend* to plot the curve of the training process.
To reproduce the output I first plotted the output of all available pre-trained models on two given image [Fig. 4-5], (Robert Downey Jr. and myself) and on three random generated images, output shown in [Fig 8-9]. Next I trained the model on two new reference images [Fig. 6-7] using 300 epochs and alpha parameter (to control style intensity) set to 1, each training taken about 3,5 hours, obtaining good results [Fig. 10-11-12-13-14-15]. I also tried a training with alpha set to 0 and obtained an output with less style intensity [Fig. 16].

Fig. 4

Fig. 5

Fig. 6

Fig. 7

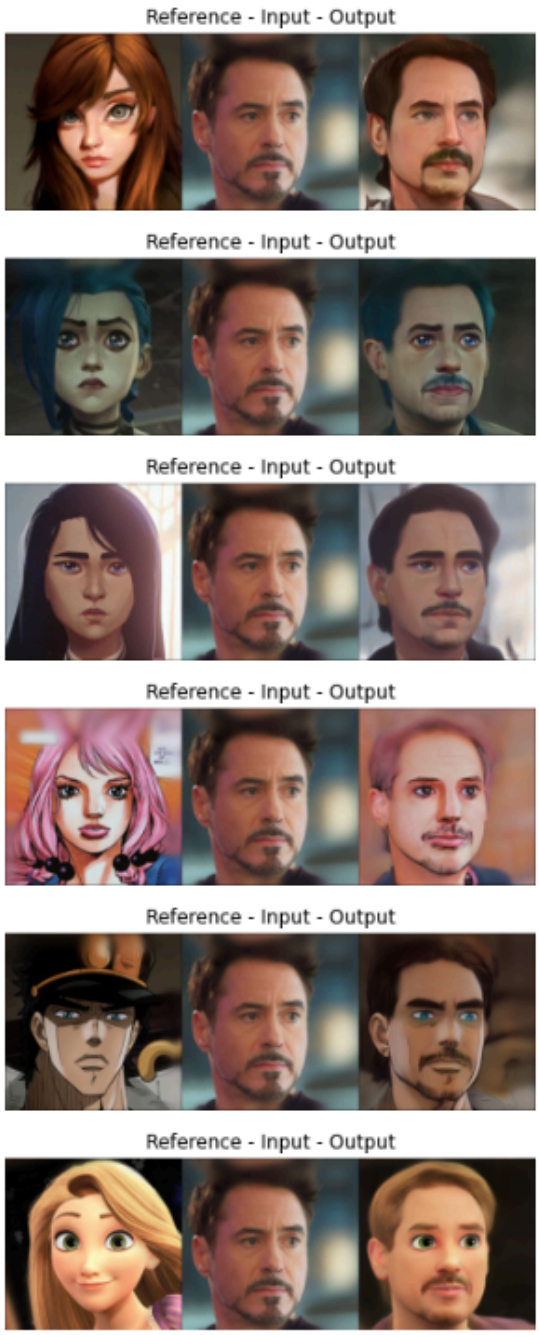

Fig. 8

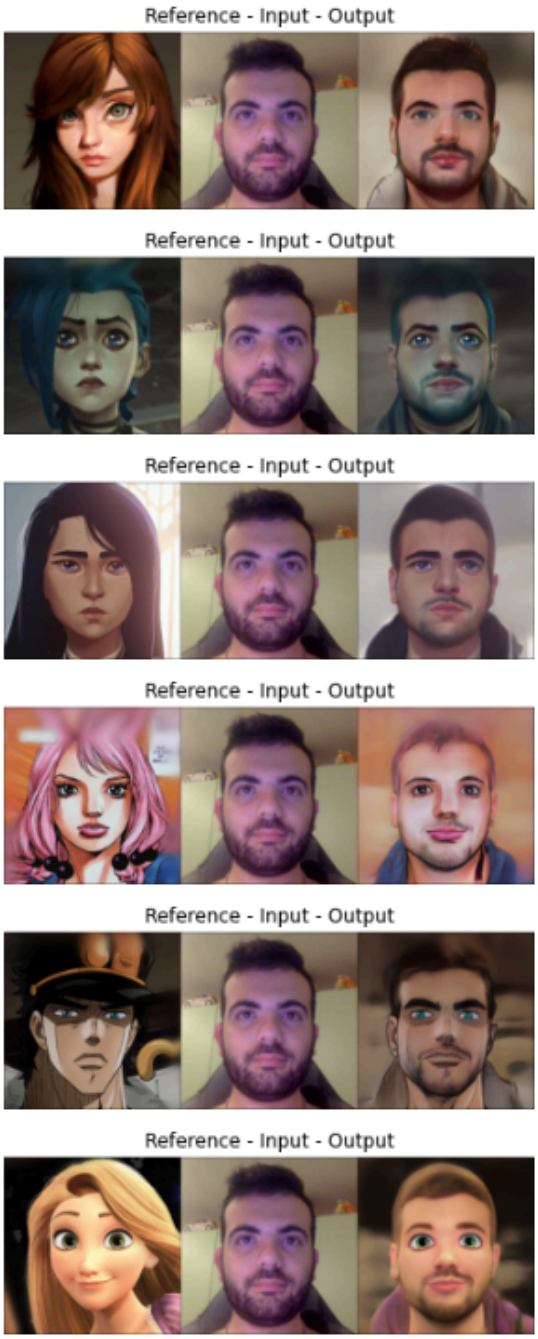

Fig.9

Fig. 10

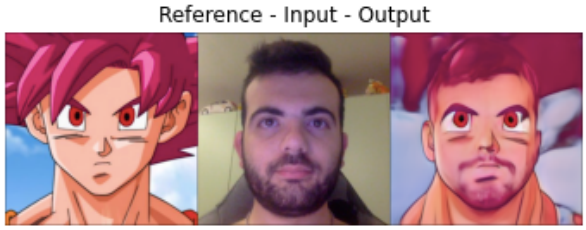

Fig. 11

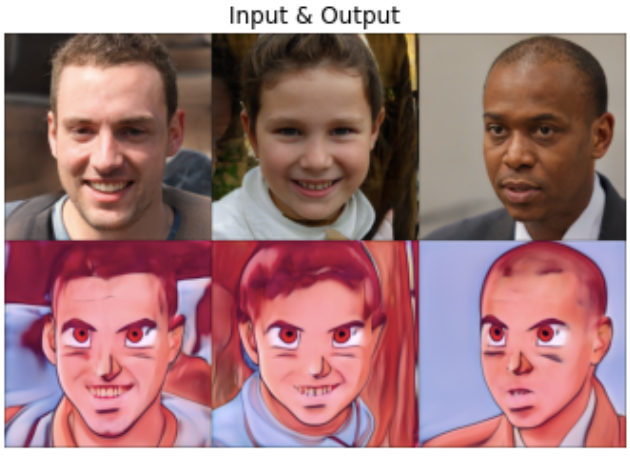

Fig. 12

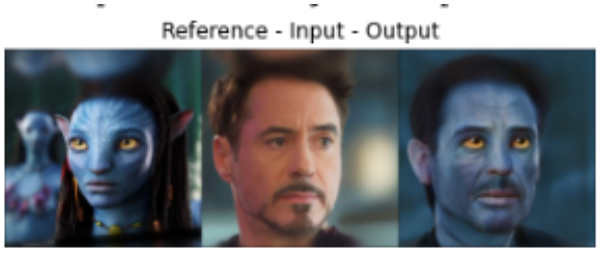

Fig. 13

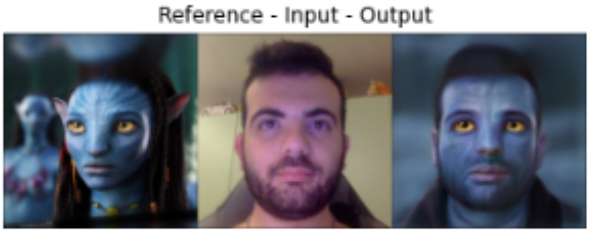

Fig. 14

Fig. 15

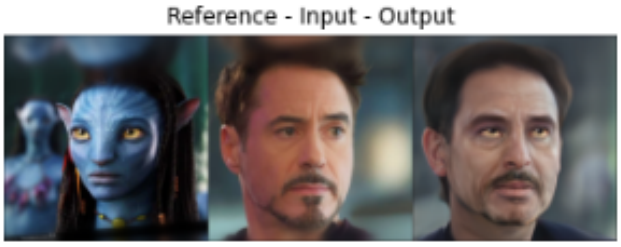

Fig. 16

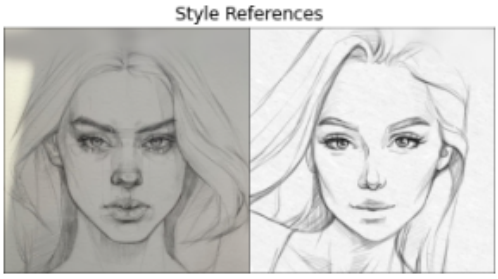

Fig. 17

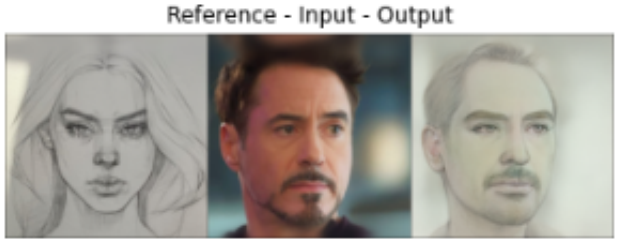

Fig. 18

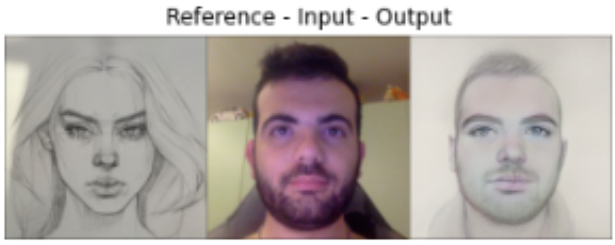

Fig. 19

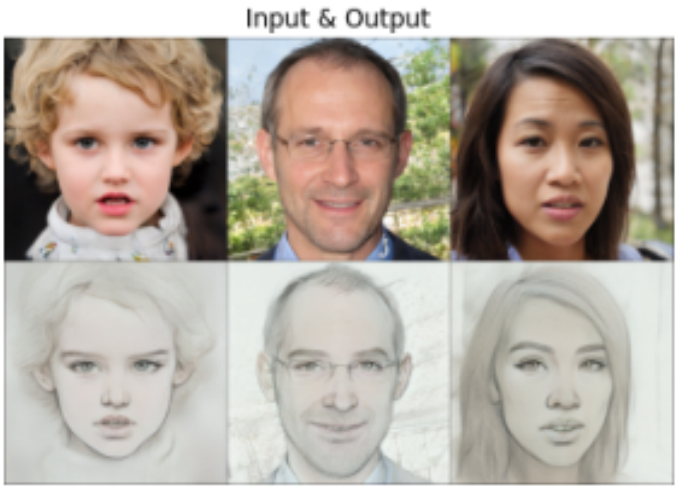

Fig. 20

Finally I also tried to do a train with more than one style reference image (2 images in particular) [Fig. 17], however it was very time and memory consuming, so I obtained acceptable outputs [Fig. 18-19-20] with 100 epochs (more than 2 hour of training), due to

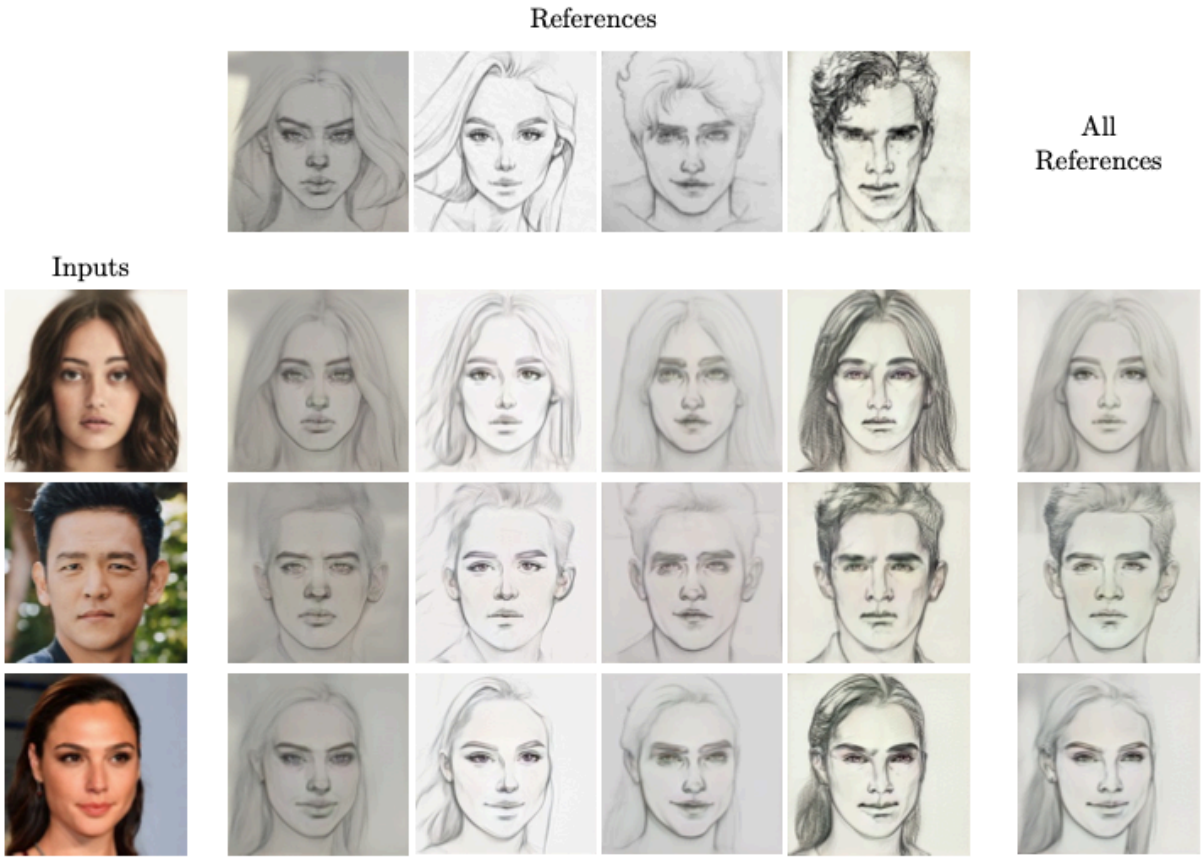

Fig. 21

this problems I'm not capable to train a new network with five reference images as shown in the paper [Fig. 21].
At the end of the notebook I used, I reported the main classes of Generator and Discriminator to better understand their structure.
[Fig. 22-23-24-25] show the training curve of the training I executed. Framework used is PyTorch.

Fig. 22

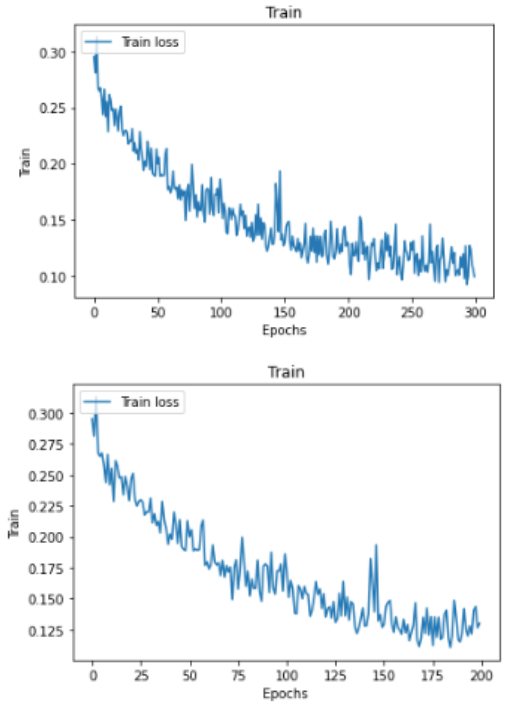

Fig. 23

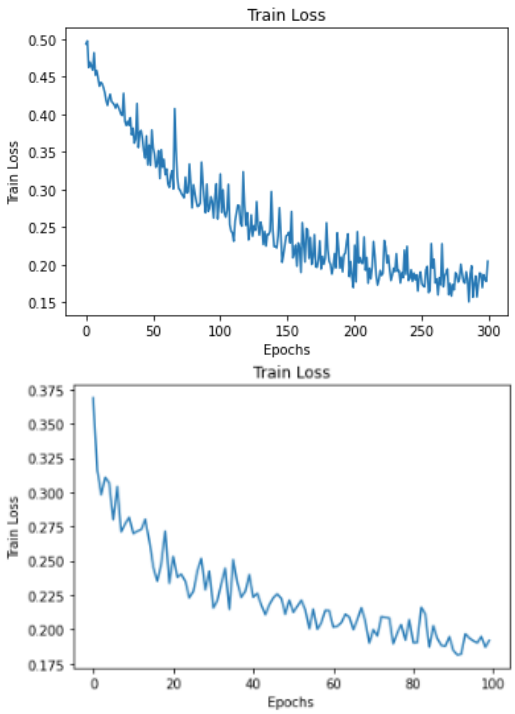

Fig. 24

Fig. 25

6. Conclusions

This work show that JoJoGAN is very competitive with respect to the state of the art in face stylization, producing extremely high quality images that get the style details right. [Fig. 26] show the output obtained in the original paper.
I reproduced the output both trough pre-trained models and training new models on new reference styles, obtaining good results as shown in the original paper.
Moreover JoJoGAN show that it is capable to produces smooth and consistent stylization as the face moves and changes expressions. Also it is possible to train a model on a different domain rather than faces and it works with same good results, [Fig. 27] show an example from original paper.

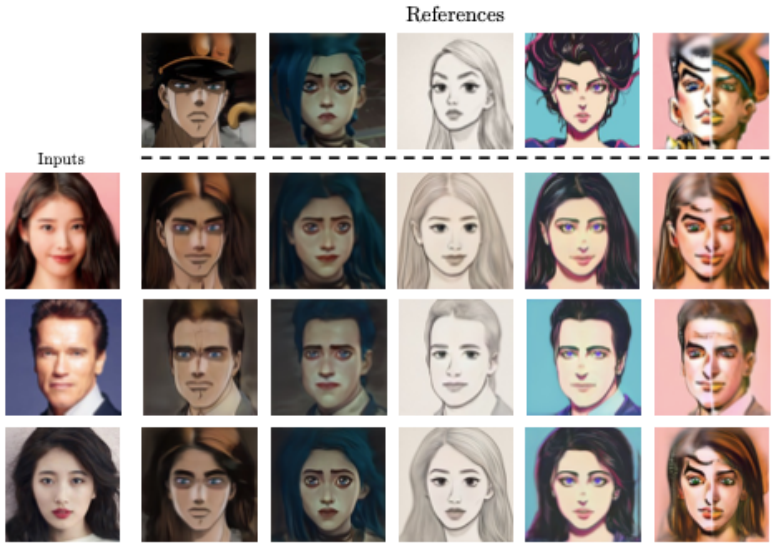

Fig. 26

Fig. 27

References

- Min Jin Chong and D.A. Forsyth: JoJoGAN: One Shot Face Stylization
- Ian J. Goodfellow, Jean Pouget-Abadie∗, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, Yoshua Bengio‡: Generative Adversarial Nets
- Tero Karras, Samuli Laine, Timo Aila: A Style-Based Generator Architecture for Generative Adversarial Networks
- Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, Daniel Cohen-Or: Designing an Encoder for StyleGAN Image Manipulation
- Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, Qifeng Chen: High-Fidelity GAN Inversion for Image Attribute Editing
- Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila: Analyzing and Improving the Image Quality of StyleGAN
- Antonio Gulli, Amita Kapoor, Sujit Pal : Deep Learning with Tensorflow 2 and keras