6 Appendix

6.1 Implementation Details

Model's architectures The Generator is a neural network with a common encoder-decoder architecture. It is described in table 6.1. The main input of the network is the image. The color conditions are fed to the generator using a depthwise concatenation with the different features of the decoder part of the generator. The procedure is described in[2][14]. The color values are spatially duplicated to match the right size depending on the considered layer. This is illustrated in Figure 7. The Discriminator network is kept simple with four convolutional and one fully-connected layers. The full details are given in table 6.1.

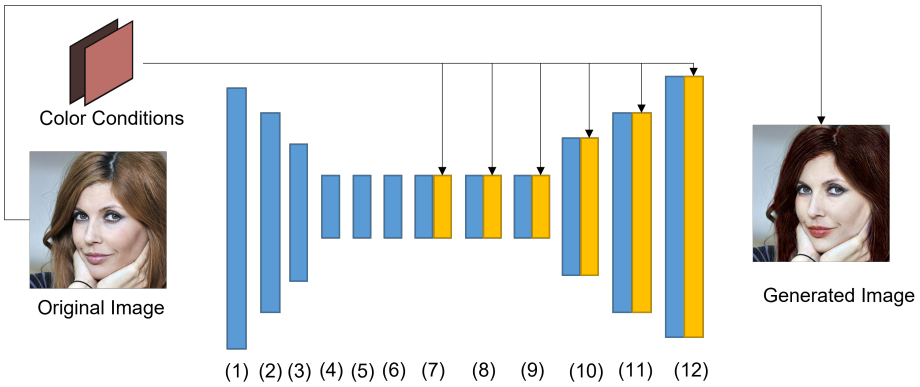

Fig. 7. Illustration of conditions incorporation in the generator. The colors are concatenated with the outputs of the convolutional layers. Finally the input image is added to the network's output. Numbers refer to the outputs of the different layers described in table 6.1.

Table 3. This table details the Discriminator network architecture.

Discriminator	Activation	Output Shape
Input image	-	256 x 256 x 3
Conv 3x3	LReLU	128 x 128 x 64
Conv 3x3	LReLU	64 x 64 x 128
Conv 3x3	LReLU	32 x 32 x 256
Conv 3x3	LReLU	16 x 16 x 512
Fully-Connected	linear	1 x 1 x 1

Table 4. This table details the Generator network architecture.

Discriminator	Activation	Output Shape
Input image	-	256 x 256 x 3
Conv 3x3 (1)	ReLU	256 x 256 x 64
Conv 3x3 (2)	ReLU	128 x 128 x 128
Conv 3x3 (3)	ReLU	64 x 64 x 256
Residual Block (4)	ReLU	64 x 64 x 256
Residual Block (5)	ReLU	64 x 64 x 256
Residual Block (6)	ReLU	64 x 64 x 256
Residual Block (7)	ReLU	64 x 64 x 256
Residual Block (8)	ReLU	64 x 64 x 256
Residual Block (9)	ReLU	64 x 64 x 256
DeConv 3x3 (10)	ReLU	64 x 64 x 256
DeConv 3x3 (11)	ReLU	128 x 128 x 128
DeConv 3x3 (12)	Tanh	256 x 256 x 64
Output image	-	256 x 256 x 3

Table 5. Metrics comparison for different values of λ_{foc}. First rows give references values of the FID and $\mathcal{L}^{color}$ between X_1 and X_2. We find $\lambda_{foc} = 600$ as the best trade-off between all the metrics.For all metrics, lower is better. Best results are in blue and second best in green.

Model	$L_{bg}^{avg} \times 10^3$	$FID_{1\to2}$	$L_{1\to2}^{color}$
Identity	0	0.045	25.2
$\lambda_{foc} = 0$	8.00	0.060	4.8
$\lambda_{foc} = 100$	4.00	0.053	5.10
$\lambda_{foc} = 250$	2.64	0.052	4.95
$\lambda_{foc} = 600 - ours$	0.92	0.050	5.03
$\lambda_{foc} = 1000$	0.89	0.048	6.42
$\lambda_{foc} = 4000$	0.87	0.051	7.05
$\lambda_{foc} = 6000$	0.78	0.056	9.49

6.2 λ_{foc} analysis

We present here a metrics analysis of the impact of λ_{foc} during the training. We observe in table 6.2 that optimising the object focus loss fulfills its objective that was limiting the modification of pixels outside of the colored object. Indeed, the higher λ_{foc} is, the smaller L_{bg}^{avg} is. This loss also impacts realism and color precision. Lower λ_{foc} values improve results, while higher values tend to have a negative influence. Penalizing too much the training through this loss tends to minimize the accepted modifications and does not let the model reach the best color precision possible. A compromise has to be found between these different objectives and for this dataset, we found $\lambda_{foc} = 600$ to be the best trade off.

6.3 Color control using StyleGAN

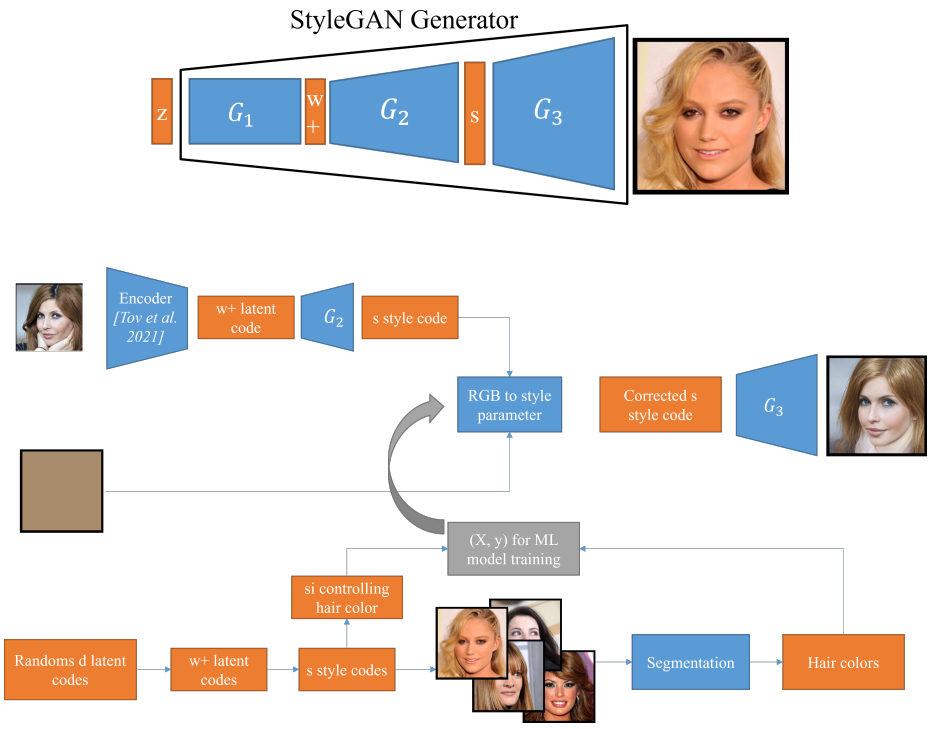

Fig. 8. Top graphic represents a simplified decomposition of StyleGAN's model Generator. In particular it shows the two intermediate states that are used, $\mathcal{W}+$ and $\mathcal{S}$. Bottom graphics illustrates the process used to achieve color control with StyleGAN's Generator.

StyleGAN's original purpose is to generate realistic images from random vectors. Its effectiveness has led to the development of diverse methods for image

to image translation. We wanted a tangible comparison element with our framework, and have investigated StyleGAN. We present here how to add color control to this existing model. This method can edit an object's color from a given RGB value, like our framework. We have specifically focussed on hair color for this study.

We created a dataset $\mathcal{X}$ of a thousand images randomly generated with StyleGAN. For each $X \in \mathcal{X}$ corresponds a style vector $s \in S$ with S the StyleSpace described in[32]. In this paper, the auhtors also describe how to only modify s^j, specific elements of s as to only modify hair color of the generated image. By choosing the right ds as only to change these s^j elements, we create the two new style vectors $s + ds$ and $s - ds$. Using StyleGAN's model, this leads to two manipulations of X with different hair colors, X_{ds^-} and X_{ds^+}. We are now able to also compute $dc^+ = C(X_{ds^+}) - C(X)$ and $dc^- = C(X_{ds^-}) - C(X)$. This is enough to have a sufficient dataset to train a machine learning model to predict ds from the desired hair color difference, dc.We can now automatically modify a StyleSpace vector as to modify the hair color according to a desired RGB value.

The workflow to use StyleGAN for hair coloring illustrated in Figure 8 is as follows:

- Encode the original image in StyleGAN's latent space, using [28]
- This first encoding, $w \in W$ leads to a $s \in S$ StyleSpace vector
- Computing ds to reach the desired hair color
- Generate resulting image from corrected $s + ds$ using StyleGAN generator

The objective here was not to find the best method to control color generations using StyleGAN. It was merely to find a simple workaround that would let us compare our approach with one of the most performing method in image generation.

6.4 Additional results

Color Edition Control We present here a few additional results of our model on the three use cases: lips and hair in Figure 9, cats and dogs in Figure 10 and clothes in Figure 11.

If color edition is the application targeted by our framework, it can also be used for color transfer and colorization.

Color Transfer From a color reference image X and a content image $\tilde{X}$, the image with the transferred colors is $G(\tilde{X}, C(X))$. Our framework can be used easily for color transfer, adding a step of color extraction using C. Some results can be found in Figure 12.

Colorization By feeding G the gray-scale version of the images X during the training, it possible to obtain a colorization model. The benefit being that it is possible to control the future colors for the defined objects. Figure 13 shows results of gray images recolorization and some possible variations. Figure 14 shows results on an originally gray image. If backgrounds tend to stay in gray colors, it is particularly performing at reconstituting hair and lips colors.

Fig. 9. Example results of our framework on hair and lips color edition. Color conditions are indicated under generated images.

Fig. 10. Example results of our framework on cat and dogs color edition. Color conditions are indicated under generated images.

Fig. 11. Example results of our framework on clothes color edition. Color conditions are indicated under generated images.

Fig. 12. Example of color translation results. First row shows original images and second row shows results of generation using the neighbour's image as the desired colors.

Fig. 13. Example results for colorization. The recolored image is generated using the hair and lips color extracted from the original image and given to the model along the grayscale image. On third row is colorization with additional generated colors.

Fig. 14. Example results for colorization. Different variations from different hair and lips colors are generated from an originally grayscale image.