# Weak Segmentation-Guided GAN for realistic color edition

**Vincent Auriau[A], Emmanuel Malherbe[A,B] and Matthieu Perrot[A]**
[A]L'Oréal AI Research, France
[B]Artefact Research Center, Paris, France

**Editing the color of images in a realistic way finds many applications such as changing the perception of an image, data augmentation or film post processing. The design of an automatic tool is a complex and long addressed challenge. In particular, two properties are difficult to meet altogether: generating realistic results without artifacts and the possibility to precisely choose the future color. Conventional methods using segmentation and histogram matching maximize the controllability but also introduce a lack of realism or are complex to automate. On the contrary, GANs that specialize in realism are difficult to control. To overcome these challenges, we propose a novel GAN architecture leveraging any differentiable segmentation model. We demonstrate the genericness of our framework that presents state of the art results on different use cases. It generates images that look realistic while offering a precise color control.**

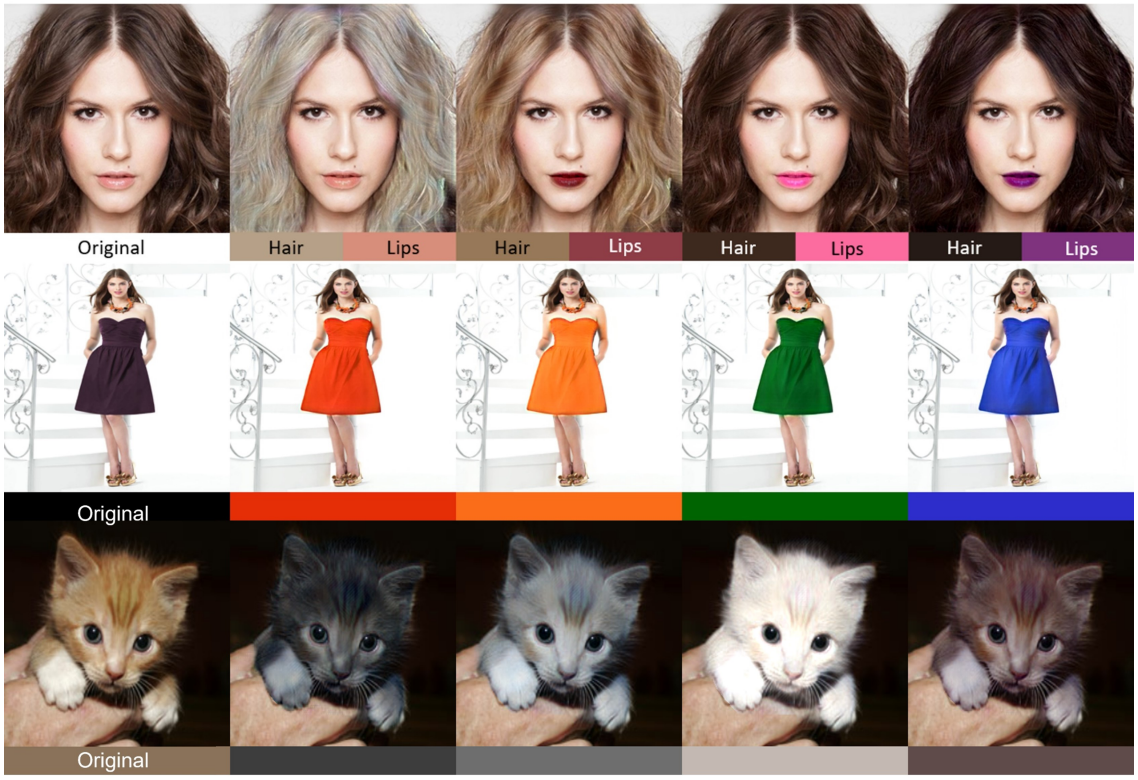



Figure 1 | Example results of our framework on three different use cases: coloring hair and lips, clothing and cats and dogs.

*Published at International Conference on Image Processing (ICIAP) 2024*
*Corresponding author(s): vincent.auriau@loreal.com*
Artefact Research Center, Paris, France

## 1. Introduction

Color edition control is the process of modifying an object's color in a graphical representation. A user can control the edition by specifying desired colors. It is a great tool for images post processing. By altering the color palette, it is possible to modify the global atmosphere or perception of the image [25]. One might want to be more precise and only adjust the color of a single area: a lighter blue sky in a landscape photograph or changing an actor's hair color in a movie. Simulating several color variations of the same car or garment can help to choose the right color among the millions of possibilities before any material realization. It can also be used to help customers select the right make up or hair coloration they desire. While physical renderers handle colors very well [26], it is still a real challenge to work with existing photographic pictures. Color transfer [15] and colorization [29] are two other applications that can be seen as specific cases of color edition. Most recent methodologies use neural networks and particularly GANs [30], however they still meet challenges with some non satisfying results.

Our goal is to realistically and precisely change an object's color in an image. We consider the general color perception of an object in the image, characterizing the object by a single color. In order to keep the human control simple, interaction of light with the object's geometry or local color variations coming from the textures are to be handled automatically.

To better define the objective, we enumerate four properties to evaluate the performances of a color edition model:

(i) **Colorization accuracy:** The perceived resulting color of the object is as close as possible as the desired one
(ii) **Realism:** The generated image is realistic to the human eye
(iii) **Stability:** The geometry of the considered object and local color variations are kept unchanged
(iv) **Restraint:** Objects and areas that should not be modified, or "background", are left untouched

We describe our robust method for color edition that overcomes the two main difficulties of existing techniques: producing realistic results while reaching a precise color control. Our main contributions are:

- A generic model architecture and training strategy for editing human perceived color while ensuring realism, stability and constraint
- A versatile object-aware color edition tool based on segmentation that addresses various use-cases and applications: color edition, color transfer and colorization
- Experimental comparison on various datasets and use cases that show how our approach achieves the best trade-off compared to the state-of-the art models

## 2. Related Work

Color manipulation in images is a long-addressed problem. Different semi-automatic or fully automatic methods have been developed, with different advantages and drawbacks. We specifically focus on color edition on the core of this paper, however inspirational technologies have been developed for color transfer or colorization.

**Color Transfer** consists of extracting the right color features from a reference image, and to transfer them to the content image. In [21], a simple but effective function for color transfer is given. The color reference image $X$ and the content image $\tilde{X}$ are described respectively by $(\mu, \sigma)$ and $(\tilde{\mu}, \tilde{\sigma})$,

the mean and standard deviation of the color distributions of each image in the L*a*b* space. The color transfer function is then defined as:

$$g(\tilde{X}_{i,j}) = \mu + \frac{\sigma}{\tilde{\sigma}}(\tilde{X}_{i,j} - \tilde{\mu}) \tag{1}$$

More complex transformation manipulate the whole color histogram [18][27]. It enables modifications of the color palettes that would not be possible with a simple linear translation. In return, the needed computation is much greater. Most state of the art are today produced by deep learning. In [5] a neural color transfer method between images is proposed. It uses a trained VGG-19 to match semantically meaningful entities between the reference and content images, before applying Equation 1. While having an image to transfer the color from is of great help, it might be long or complicated to find the very example image that perfectly fits the user's intention. It also limits creativity, forcing the use of already existing color palettes.

Convolution neural network have also shown the first good results on automatic **Gray Image Colorization** [6]. The use of conditional GAN with the pix2pix model [7] have led to more realistic results. These methods do not allow any color controlability as the resulting color cannot be decided by the user. Such control is introduced in [24] with colored drawn strokes. No measure of the color precision is developed and background is often impacted, creating undesired artifacts.

In **Color Edition Control**, a user can modify at will the colors of an object in an image. Pixels to be edited can be selected using a segmentation model [20], which was more recently performed using neural segmentation [13, 27]. The desired color is defined by the user in the form of a histogram or color look up tables. Respectively histogram matching algorithms and interpolation are then applied to modify the selected pixels. While providing good results, these methods suffer in terms of realism, particularly around the edges of the computed mask where artifacts tend to appear. Moreover the color formats are complicated to manipulate for a user.

Another strategy that overcomes those limitations is to use a conditional GAN architecture [17] such as the one of StarGAN [2], with the colors as condition variables. CAGAN [10] proposes to automatically extract colors and let the generator indirectly focus on the right area without further explicit supervision. The major persisting challenges are generated images color precision and the inherent modifications of background pixels, that create inconsistencies. To overcome this issue, MagGAN [31] suggests to feed the model with manually labelled masks to improve the focus on a specific zone. However it is not designed for continuous color space and requires mask labels as input.

In this paper, we aim at bridging the different approaches presented above, GANs for realism and segmentation for object selection. It leads us to create an accurate, controllable and versatile coloring tool for edition.

## 3. Proposed model

Our approach consists of Generator $G$ that produces the transformed image $\hat{X} = G(X, c)$ from the input image $X$, with the considered object colorized as described by the color vector $c$. We also define $S$, a generic and differentiable segmentation model such that $S(X)$ is the probability for each pixel to represent the object: $S(X) \in [0; 1]^{W \times H}$ where $S(X)_{i,j} = \mathbb{P}(x_{i,j} \in object)$. In practice we use a neural network for $S$ that is trained separately from the rest of the framework and frozen, but it can be generalized to any differentiable function.

$G$ is a convolutional neural network with an encoder-decoder architecture. The decoder takes the image features created by the encoder and the desired color as inputs and generates the colorized image. $G$ is trained to optimize four modules that ensure the required properties: $C$ for colorization

accuracy, $D$ for realism, $R$ for stability and finally $OF$ for restraint. The general organisation is illustrated Figure 2, and the modules described in the following sections.

### 3.1. Color Estimation module $C$

We need a robust color estimation function that should be differentiable for later optimization purposes. We describe an object's color as an average $RGB$ value over its representing pixels. This estimator characterizes well the general color of a uniformly colored object and can be easily represented and understood by a user. The weighted mean color of an object $C(X)$ is then:

$$C(X) = \mathbb{E}_{object}[X] = \frac{\sum_{i,j} S(X)_{i,j} * x_{i,j}}{\sum_{i,j} S(X)_{i,j}} \tag{2}$$

with $x_{i,j}$ the color vector at pixel (i, j). It is illustrated in Figure 3.

This color estimator is used to estimate the color of our object in the generated image $\hat{X}$ and compare it to the desired color $c$. The color loss is a $L_2$ loss computed in the $CIEL^*a^*b^*_{76}$ color space as it better matches human perception [10, 22].

$$\mathcal{L}_{color} = ||C(\hat{X}) - c||_{L^*a^*b^*_{76}} \tag{3}$$

Thanks to the use of $S$, we are sure to extract and therefore control the color of the object itself.

### 3.2. Discriminator module $D$

The GAN structure ensures that the main coloring model $G$ produces realistic results by adversarially training a discriminator neural network, $D$. It tries to distinguish real images from the one edited by $G$. Our adversarial loss is the standard Wasserstein GAN loss with gradient penalty [1]:

$$\mathcal{L}_{adv} = D(\hat{X}) - D(X) + \lambda_{gp} \nabla_{\tilde{X}} D(\tilde{X}) \tag{4}$$

with $\tilde{X}$ randomly obtained for each batch by interpolating between real images $X$ and generated images $\hat{X}$.

### 3.3. Reconstruction module $R$

In order to ensure stability, we use the popular cycle consistency loss introduced in CycleGAN [33] for image to image translation. The model is trained to be able to recreate the original image from its own generation. It ensures that $G$ maintains the objects geometry and that no pixel or information is lost through the process. The reconstruction loss is defined as:

$$\mathcal{L}_{rec} = ||X - G(G(X, c), C(X))||^2 \tag{5}$$

The reconstruction module is illustrated in Figure 4.

### 3.4. Object Focus module $OF$

With our frozen segmentation model $S$, we can also ensure restraint, meaning that $G$ keeps background pixels unchanged. We only want pixels that relate to the object's color change to be edited by the

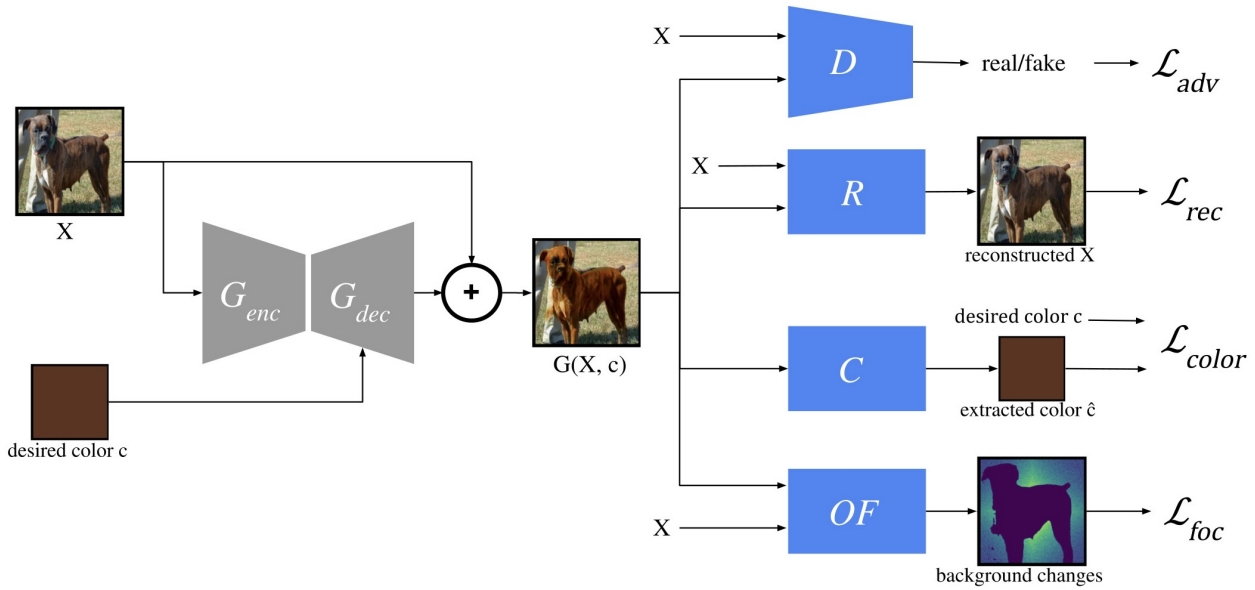



Figure 2 | General organization of our framework. Generated images are evaluated through the four defined modules that design $G$'s training: $C$, $D$, $OF$ and $R$. $G$ actually predicts a difference that is applied to the input image.

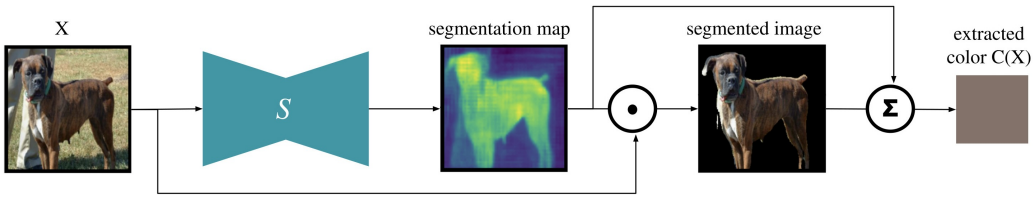



Figure 3 | Color estimation module $C$: from the segmentation map given by the segmentation model $S$, the weighted mean $C(X)$ is computed to represent the color of the object.

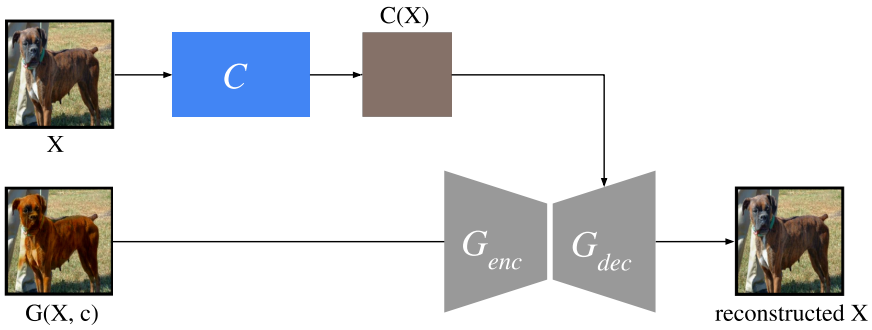



Figure 4 | Reconstruction module $R$: The edition model $G$ is asked to reconstruct an original image from one of its own edition.

model. These background alterations is a common artifact produced by GAN based approaches. The Object Focus loss $\mathcal{L}_{foc}$ penalizes any change operated outside of the segmentation mask representing our object:

$$\mathcal{L}_{foc} = ||(X - \hat{X})) \odot (\mathbf{1} - S(X))||^2 \tag{6}$$

with $\mathbf{1}$ the matrix of same shape as $S(X)$ and filled with ones and $\odot$ the element wise product on the R, G and B channels. The resulting object focus module is illustrated in Figure 5.

### 3.5. General objective

We summarize the final objective to train $G$ and $D$ as follows:

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{color}\mathcal{L}_{color} + \lambda_{foc}\mathcal{L}_{foc} + \lambda_{rec}\mathcal{L}_{rec} \tag{7}$$

$$\mathcal{L}_D = -\mathcal{L}_{adv} \tag{8}$$

where the different $\lambda$ are used to set up and control the importance of the different losses in the final objective. The general workflow of the model is illustrated in Figure 2. Models $G$ and $D$ are trained by this process, $S$ is trained beforehand separately.

We have presented our framework to edit a single object's color. It can be easily extended to any number of objects. One only needs to sum the losses over the different objects.

## 4. Experiments

### 4.1. Datasets

We have tested our framework on three different use cases supported by open-source datasets designed to train the segmentation model. First, we used CelebA-HQ dataset [12] to train our hair and lips segmentation network. It was then completed with the FFHQ dataset [8] to improve the data diversity. Secondly, the cat and dogs dataset [16] was used for coloring pets. For this use case, we used an already trained segmentation network from [4]. We plugged this model to our framework that works without further need of adaptation. Finally, the clothing dataset [11] was used to train the segmentation model as well as the color edition model for the garnments use-case. Example of results

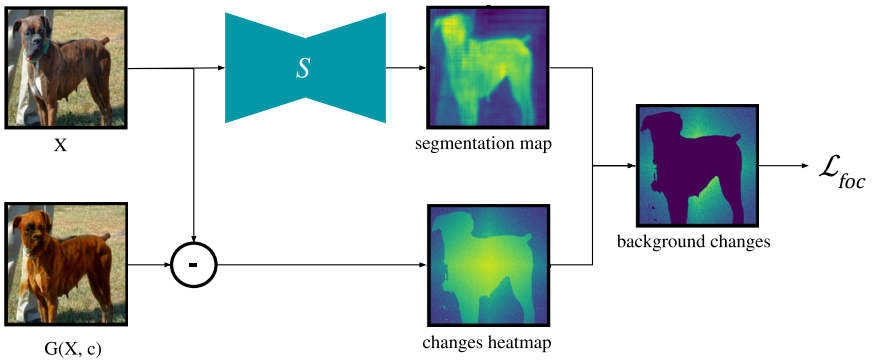



Figure 5 | Object focus module *OF*: Generated images are compared to the original ones and all pixels modified outside the segmentation maps are penalized. This helps to keep a background consistency between original and generated images.

can be found in Figure 1. The results presented in this part are all obtained using the first datasets. Additional material and examples are shared in the appendix [1].

### 4.2. Implementation

For the following experiments, the parameters described in this part were used, unless it is specifically specified otherwise. The models were trained on $256 \times 256$ pixels images resized by bicubic interpolation. For the trained segmentation models, we used a U-Net segmentation network [23]. The generator and discriminator architectures can be found in the Appendix[1]. For losses weighting, we used the parameters: $\lambda_{rec} = 1$, $\lambda_{cond} = 80$ , $\lambda_{foc} = 600$ and $\lambda_{gp} = 6$. We trained both networks with the Adam optimizer with learning rates $5 \times 10^5$ for the generator and $10^5$ for the discriminator.

### 4.3. Evaluation Protocol

In order to compare our framework with other approaches, we use quantitative metrics computed with the following protocol: We select a random subset $\mathcal{X}_1$ composed of 10k images that we keep for testing. We select a second subset $\mathcal{X}_2$, of same size as $\mathcal{X}_1$. We generate $G(X_1^i, C(X_2^i))$ for each $(X_1^i, X_2^i) \in (\mathcal{X}_1, \mathcal{X}_2)$ to build the dataset $\mathcal{X}_{1\to2}$.

**Metrics** For realism quantification, we use the Fréchet Inception Distance, or FID [19]. We compare our generated images $\mathcal{X}_{1\to2}$ with $\mathcal{X}_2$: $FID_{1\to2} = FID(\mathcal{X}_{1\to2}, \mathcal{X}_2)$. For color precision estimation, we use $L_{1\to2}^{color} = \frac{1}{n}\sum_{i=1}^{n} ||C(X_{1\to2}^i) - C(X_2^i)||_{L^*a^*b^*_{76}}$. We also compute a modified version of $\mathcal{L}_{foc}$ to quantify how well the different models perform on the task of focusing on the right pixels. $L_{bg}^{avg}$ is defined as:

$$L_{bg}^{avg} = \frac{1}{3n}\sum_{i=1}^{n} \frac{||(X_{1\to2}^i - X_1^i) \odot (\mathbf{1} - M_1^i)||_1}{||\mathbf{1} - M_1^i||_1} \tag{9}$$

with $(X_1^i)_n$ being all the images of $X_1$, $M_1^i$ the manually annotated mask corresponding to $X_1^i$, $X_{1\to2}^i$ the corresponding generated image and $\mathbf{1}$ the matrix filled with ones. $L_{bg}^{avg}$ represents the average modification of a background pixel's channel. The $\mathcal{L}_1$ norm is taken to better express human perception in $RGB$ [3].

**User test** We have performed a user study for human quantitative evaluation of the different approaches. We have asked five annotators which was the most realistic edition among the one generated by the different approaches. We have repeated the experience with 30 random triplets of editions, totalling 150 annotations.

### 4.4. Comparison with other approaches

We compare our framework with other approaches that aim at coloring objects in images. The first approach, alpha blending [13][18] uses segmentation paired with RGB shift. For fair comparison, the area to edit is selected using $S$, then Equation 1 is applied for the average color of segmented pixels to match the desired one.

The second one, CAGAN [10], is a straightforward adaptation of StarGAN [2] for color control. A third model is trained besides $D$ for color extraction. It uses labels, without any guidance on the

[1]Appendix can be found at: https://github.com/VincentAuriau/wsg-gan-color-edition/blob/main/pdf_files/appendix.pdf

object. For fair comparison we extracted color labels using $S$. This third model induces noise in the global color precision of the editions.

Finally, we also benchmark our model with one of the most performing GAN, StyleGAN [9]. As it is not built specifically for coloring, we developed a straight forward strategy on the style space [32] which is detailed in the Appendix [1]. This simple adaptation enables to perform color edition using StyleGAN. More complex adaptation could be designed, but it is beyond the scope of this paper.

**Qualitative evaluation** Figure 6 shows examples of results for hair colorization. We observed the main defaults of compared approaches. Alpha Blending creates artifacts on the edges of the segmented mask (see (a)). CAGAN tends to edit pixels outside hair pixels. It is particularly obvious on (a) with the face color that is different and on (d) where the background is affected. Finally, StyleGAN which is not built for image-to-image translation changes the faces attributes, the projection failing at perfectly reproducing the original image.

**Quantitative evaluation** Table 1 shows a metric analysis of the approaches. We observed that our model is a good balance between color precision and realism. Our objective was to compensate alpha blending's lack of realism using GANs. However, typical GAN approaches come at a cost, particularly on color precision. In addition, it is more difficult to control the outcome of the GAN. It can be seen with $L_{bg}^{avg}$ for CAGAN and ctrl-StyleGAN, showing that many pixels are modified whereas they should not. We observe that our model outperforms other GAN approaches on these topics.

Our approach obtained the majority of users votes and particularly outperforms the other GAN-based approach, CAGAN. Compared to Alpha Blending, our method seems to have better results particularly with the lack of textures in images with darker colored objects.

### 4.5. Ablation Study

We have also analysed the behaviour of the different modules within our framework. We have trained our model and computed the metrics with and without the reconstruction and object focus losses. The results can be found in Table 2. In order to emphasize the importance of $S$, we also train $GTmasks$, an equivalent of our framework with losses computed with ground truth masks instead of $S$'s predictions. We test this strategy for Object Focus, $GTmask_{OF}$, and combined with Color Estimation, $GTmask_{C+OF}$. $GTmask_{C+OF}$ is thus optimized without a fully differentiable $C(X)$.

We observe that optimising the object focus loss fulfills the objective of limiting the modification of pixels outside of the colored object. Combined with reconstruction loss, it further improves all metrics. Finally, the use of a differentiable $S$ function considerably improves the results.

Table 1 | Quantitative results of the color edition frameworks. We show that our approach outperforms alpha blending in terms of $FID$ and other approaches on all metrics. Our model also outperforms others on the users votes. For the first three metrics, lower is better. Best results are in blue and second best in green.

| Model | $L_{bg}^{avg} \times 10^3$ | $FID_{1\to2}$ | $L_{1\to2}^{color}$ | User % Most Realistic |
|---|---|---|---|---|
| Ours | 0.92 | 0.050 | 5.03 | 61% |
| Ctrl- StyleGAN | 9.51 | 0.670 | 17.2 | 0% |
| CAGAN | 121 | 0.096 | 11.3 | 0% |
| Alpha Blending | 0.00 | 1.301 | 0.00 | 34% |

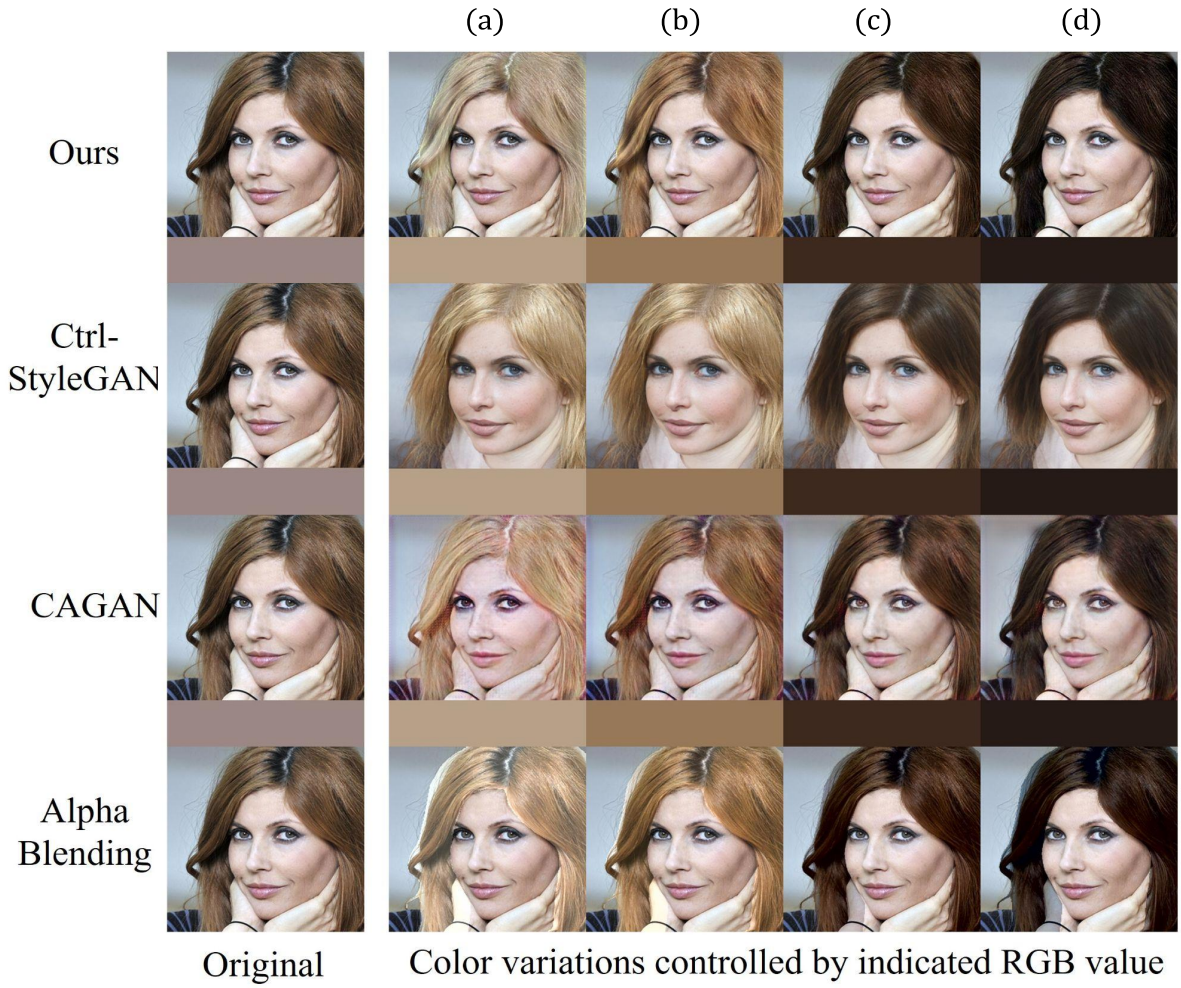



Figure 6 | Comparison of different methods for hair coloring. The desired hair color is illustrated on each column between rows.

Table 2 | Ablation study of our proposed model. The Object Focus loss improves realism and background preservation while reconstruction positively affects all metrics. For all metrics, lower is better. Best results are in blue and second best in green, identity excluded

| Model | $L_{bg}^{avg} \times 10^3$ | $FID_{1\to2}$ | $L_{1\to2}^{color}$ |
|---|---|---|---|
| *Identity* | *0.00* | *0.045* | *25.2* |
| $\lambda_{foc} = 0, \lambda_{rec} = 0$ | 8.38 | 0.051 | 4.46 |
| $\lambda_{foc} = 0, \lambda_{rec} = 1$ | 8.00 | 0.060 | 4.80 |
| $\lambda_{foc} = 600, \lambda_{rec} = 0$ | 1.01 | 0.056 | 5.99 |
| Ours | 0.92 | 0.050 | 5.03 |
| $GTmask_{C+OF}$ | 3.40 | 0.082 | 8.51 |
| $GTmask_{OF}$ | 1.79 | 0.052 | 5.20 |

## 5. Conclusion

We presented our framework for generic objects colorizing. Our new object-aware colorization model based on segmentation, introduces specific losses to improve the realism and the color precision of the results. It can be used on any type of image of objects and only needs a pretrained and differentiable segmentation model to work. We have shown state of the art results on several datasets, with the best balance between realism of generated images, accurate color of transformed object as well as maintained geometry and background. Our framework can be adapted for different applications such as color edition, color transfer or colorization, as shown through the experiments on several use-cases. As future work, we will focus on spatial distribution of the color. In order to let the user manipulate more than one general color of the object while keeping the simplicity of use.

## References

[1] Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan (2017)

[2] Choi, Y., Choi, M., Kim, M., Ha, J., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. CoRR **abs/1711.09020** (2017)

[3] Concha, A., Civera, J.: An evaluation of robust cost functions for rgb direct mapping. In: 2015 European Conference on Mobile Robots (ECMR)

[4] Emil Zakirov, b.: Keras implementation of deeplabv3+ (2019)

[5] He, M., Liao, J., Yuan, L., Sander, P.V.: Neural color transfer between images. CoRR **abs/1710.00756** (2017)

[6] He, Z., Zuo, W., Kan, M., Shan, S., Chen, X.: Attgan: Facial attribute editing by only changing what you want (2018)

[7] Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. CoRR **abs/1611.07004** (2016)

[8] Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: IEEE CVPR 2019 (2019)

[9] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. CoRR **abs/1912.04958** (2019)

[10] Kips, R., Gori, P., Perrot, .M., Bloch, I.: CA-GAN: weakly supervised color aware GAN for controllable makeup transfer. CoRR **abs/2008.10298** (2020), https://arxiv.org/abs/2008.10298

[11] Lakshmanamoorthy, R.: People clothing segmentation. https://www.kaggle.com/rajkumarl/people-clothing-segmentation (2021)

[12] Lee, C.H., Liu, Z., Wu, L., Luo, P.: Maskgan: Towards diverse and interactive facial image manipulation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[13] Levinshtein, A., Chang, C., Phung, E., Kezele, I., Guo, W., Aarabi, P.: Real-time deep hair matting on mobile devices (2018)

[14] Liu, M., Ding, Y., Xia, M., Liu, X., Ding, E., Zuo, W., Wen, S.: Stgan: A unified selective transfer network for arbitrary image attribute editing. In: IEEE CVPR (2019)

[15] Liu, S.: An overview of color transfer and style transfer for images and videos (04 2022)

[16] Microsoft: Cats and dogs: a dataset for kaggle challenge. https://www.microsoft.com/en-us/download/details.aspx?id=54765/ (2017)

[17] Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR **abs/1411.1784** (2014), http://arxiv.org/abs/1411.1784

[18] Neumann, L., Neumann, A.: Color style transfer techniques using hue, lightness and saturation histogram matching. In: Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging. p. 111–122. Computational, Eurographics Association, Goslar, DEU (2005)

[19] Nunn, E.J., Khadivi, P., Samavi, S.: Compound frechet inception distance for quality assessment of GAN created images. CoRR **abs/2106.08575** (2021)

[20] Pal, N.R., Pal, S.K.: A review on image segmentation techniques. Pattern Recognition **26**(9), 1277–1294 (1993)

[21] Reinhard, E., Ashikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Computer Graphics and Applications **21**, 34–41 (2001)

[22] Reinhard, E., Pouli, T.: Colour spaces for colour transfer

[23] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. CoRR **abs/1505.04597** (2015)

[24] Sangkloy, P., Lu, J., Fang, C., Yu, F., Hays, J.: Scribbler: Controlling deep image synthesis with sketch and color (2016). https://doi.org/10.48550/ARXIV.1612.00835

[25] Seifi, H., DiPaola, S., Enns, J.T.: Exploring the Effect of Color Palette in Painterly Rendered Character Sequences. In: Computational Aesthetics in Graphics, Visualization, and Imaging (2012)

[26] Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., Pandey, R., Fanello, S., Wetzstein, G., Zhu, J., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D., Zollhöfer, M.: State of the art on neural rendering. Computer Graphics Forum (2020)

[27] Tkachenka, A., Karpiak, G., Vakunov, A., Kartynnik, Y., Ablavatski, A., Bazarevsky, V., Pisarchyk, S.: Real-time hair segmentation and recoloring on mobile gpus. CoRR **abs/1907.06740** (2019)

[28] Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., Cohen-Or, D.: Designing an encoder for stylegan image manipulation. CoRR **abs/2102.02766** (2021), https://arxiv.org/abs/2102.02766

[29] Varga, D., Szirányi, T.: Fully automatic image colorization based on convolutional neural network. In: 23rd ICPR (2016)

[30] Vitoria, P., Raad, L., Ballester, C.: Chromagan: Adversarial picture colorization with semantic class distribution. In: The IEEE Winter Conference on Applications of Computer Vision. pp. 2445–2454 (2020)

[31] Wei, Y., Gan, Z., Li, W., Lyu, S., Chang, M., Zhang, L., Gao, J., Zhang, P.: Maggan: High-resolution face attribute editing with mask-guided generative adversarial network. CoRR **abs/2010.01424** (2020)

[32] Wu, Z., Lischinski, D., Shechtman, E.: Stylespace analysis: Disentangled controls for stylegan image generation. CoRR **abs/2011.12799** (2020)

[33] Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks (2020)

# 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 **??**. 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 **??**.

## 6.2. $\lambda_{foc}$ analysis

We present here a metrics analysis of the impact of $\lambda_{foc}$ during the training. We observe in table **??** that optimising the object focus loss fulfills its objective that was limiting the modification of pixels outside of the colored object. Indeed, the higher $\lambda_{foc}$ is, the smaller $L_{bg}^{avg}$ is. This loss also impacts realism and color precision. Lower $\lambda_{foc}$ 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

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.

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 |

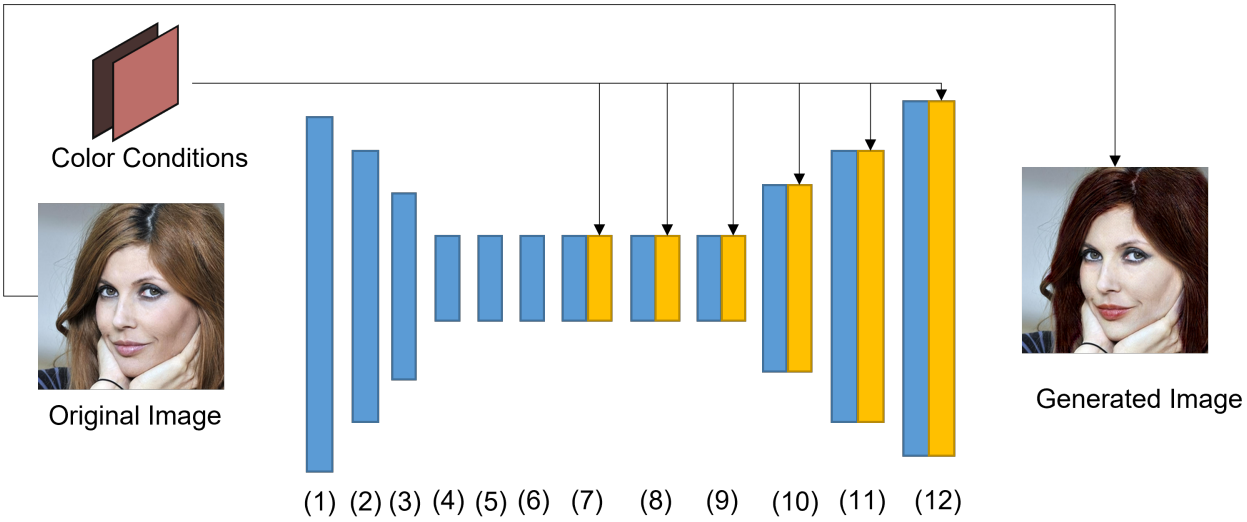



Figure 7 | Illustration of conditions incorporation in the generator. The colors are concatenated with the outputs of the convolutional layers. Finally the input image is added to the network's output. Numbers refer to the outputs of the different layers described in table **??**.

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 $\lambda_{foc}$. 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 |

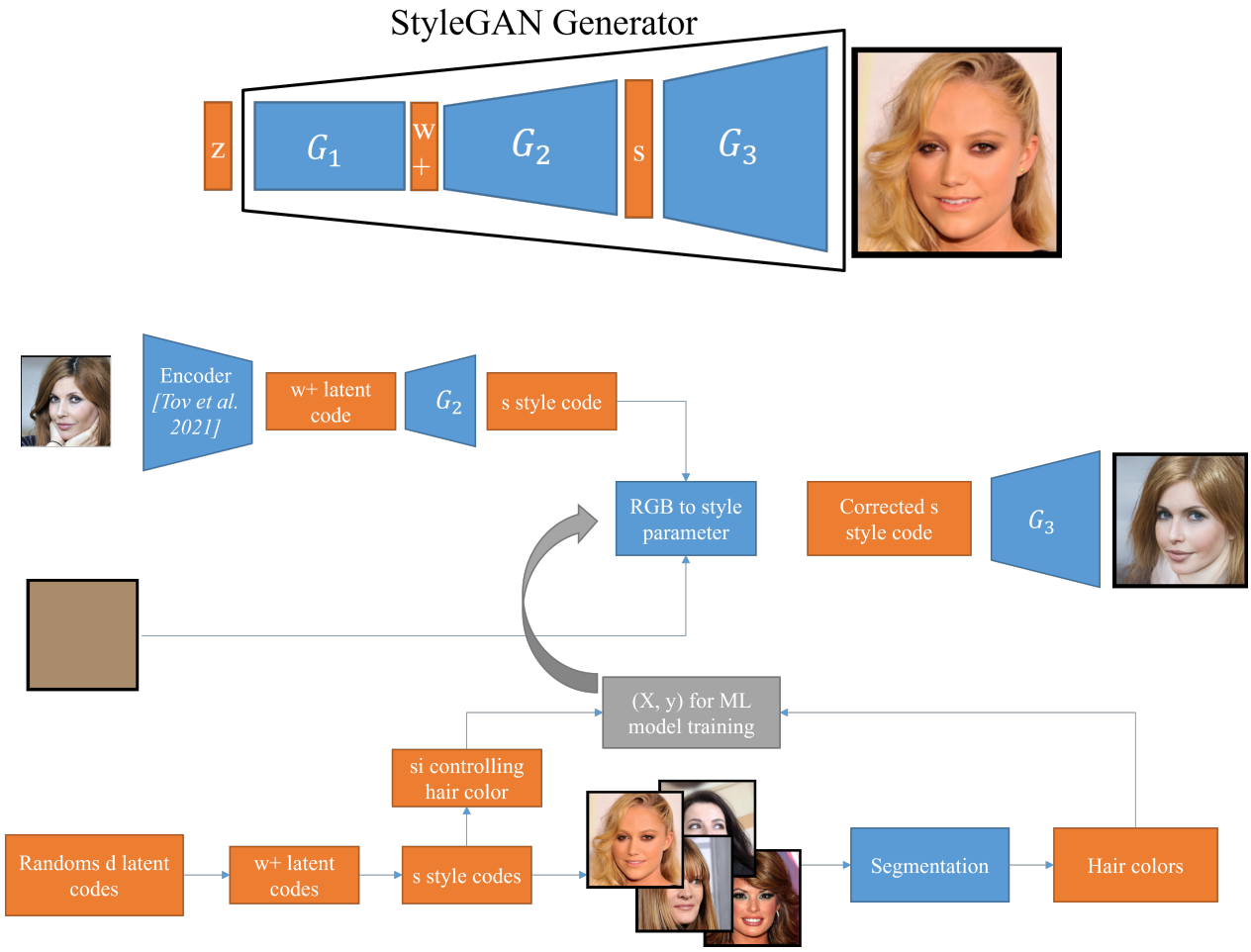



Figure 8 | Top graphic represents a simplified decomposition of StyleGAN's model Generator. In particular it shows the two intermediate states that are used, $\mathcal{W}+$ and $\mathcal{S}$. Bottom graphics illustrates the process used to achieve color control with StyleGAN's Generator.

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.




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




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




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




Figure 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.




Figure 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.




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