IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2024–05–24.
Submitted on 2024–04–16, accepted on 2024–05–03.
ISSN 2105–1232 © 2024 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2024.542

A Short Analysis of BigColor for Image Colorization

Rosana García[1], Gregory Randall[1], Lara Raad[1,2]

[1]IIE, Facultad de Ingeniería, Universidad de la República, Uruguay
[2]LIGM, ESIEE Paris, Université Gustave Eiffel, France

Communicated by Pablo Musé *Demo edited by* Lara Raad and Rosana García

Abstract

This work analyzes the BigColor method, a fully automatic colorization approach that aims to meet the challenge of providing realistic and vivid colorization for complex and diverse images in real-world scenarios. The method is a BigGAN-inspired encoder-generator network, using a spatial feature map, enabling single forward-pass colorization, supporting arbitrary input resolutions, and producing multimodal colorization results. We provide a short analysis of the method's results and highlight some limitations alongside its achievements.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README` file of the archive. The authors' original method implementation is available here[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: colorization; generative color prior; BigGAN

1 Introduction

Image colorization is a long-standing topic in computer vision, aiming to add chromatic information to grayscale images. It has numerous applications, such as colorizing old photographs, restoring legacy films, and transferring colors between artworks. In addition to user-guided scribble-based and example-based colorization methods, a variety of deep learning-based approaches have lately been introduced that allow fully automatic colorization (once the network is trained) [7]. Recent methods have incorporated generative priors learned from pre-trained Generative Adversarial Network [4] (GAN) models to address the challenge of synthesizing vivid and realistic colors using fully automatic colorization. By inverting an input grayscale image to a latent code of a pre-trained GAN model, these methods aim to leverage the learned generative priors of natural images. However, existing

[1]https://doi.org/10.5201/ipol.2024.542
[2]https://github.com/KIMGEONUNG/BigColor

colorization methods using generative priors still need help to handle complex in-the-wild images with intricate structures and semantics, resulting in desaturated and unnatural colorization outcomes. In this paper, we focus on the analysis of a method that tries to overcome the latter, called "BigColor: Colorization using a Generative Color Prior for Natural Images", proposed by G. Kim et al. in July 2022 [6]. This method aims to generate images with vivid and diverse colors for a wide range of scene complexity, and to provide the ability to use a color image as a reference, allowing user interaction in the inference. This article briefly describes the method and provides a thorough analysis of BigColor's performance on various types of images, including artistic ones.

2 BigColor

BigColor is an image colorization algorithm that uses generative color priors through a GAN-inversion strategy using a pre-trained BigGAN [1], a state-of-the-art class-conditional generative model. BigColor is an encoder-generator network trained in an adversarial manner in which the encoder and generator are designed based on BigGAN's generator. Furthermore, unlike conventional GAN-inversion colorization methods, the authors avoid simultaneous structure and color synthesis and focus solely on color synthesis by modifying the encoder output to be a spatial feature map rather than a commonly used spatially-flattened feature map. This should allow a larger set of images to be colorized correctly. Next, we will detail BigColor's modules: the encoder, the generator, the discriminator for the adversarial training, and the two control variables. An overview of BigColor can be seen in Figure 1.

Input/output. The method has only one input, a grayscale image x_g of H rows and W columns and outputs a color image $\hat{x}_{\text{rgb}}$ of 3 channels and same number of rows and columns. At inference, to preserve details, the luminance of the synthesized image $\hat{x}_{\text{rgb}}$ is replaced by the luminance of the input image x_g. In the code, a fusion module is provided, where $\hat{x}_{\text{rgb}}$ and x_g are transformed to the CIELab space, and the L-channel from $\hat{x}_{\text{rgb}}$ is replaced with the L-channel from x_g.

Control variables. The model has two control variables, a random code $z \in \mathbb{R}^{68\times 1}$ sampled from a normal distribution that allows various colorizations for a given grayscale image, and a class code $c \in \mathbb{R}^{128\times 1}$ that allows class-specific colorization. To generate the class code c, BigGAN's class embedding layer, denoted A, is adopted, transforming into c a 1000-dimensional class vector representing ImageNet-1K classes that is obtained from an off-the-shelf classifier network.

Encoder. The encoder E is a class-conditioned convolutional encoder that outputs the spatial feature map f given the input image, x_g, and the class code c

$$f = E(x_g, c).$$

The generated spatial features represent the structure of x_g. For an input of size 256×256, the spatial feature f has a spatial resolution of 16×16 and 768 channels. In fact, the encoder E is composed of five residual blocks (Figure 1), where all but the first block have average pooling layers to reduce by a factor of 2 the spatial resolution of the corresponding input features, resulting in a reduction of the spatial resolution of the input image by a factor of 16. The encoder was designed by inverting part of the original BigGAN generator architecture.

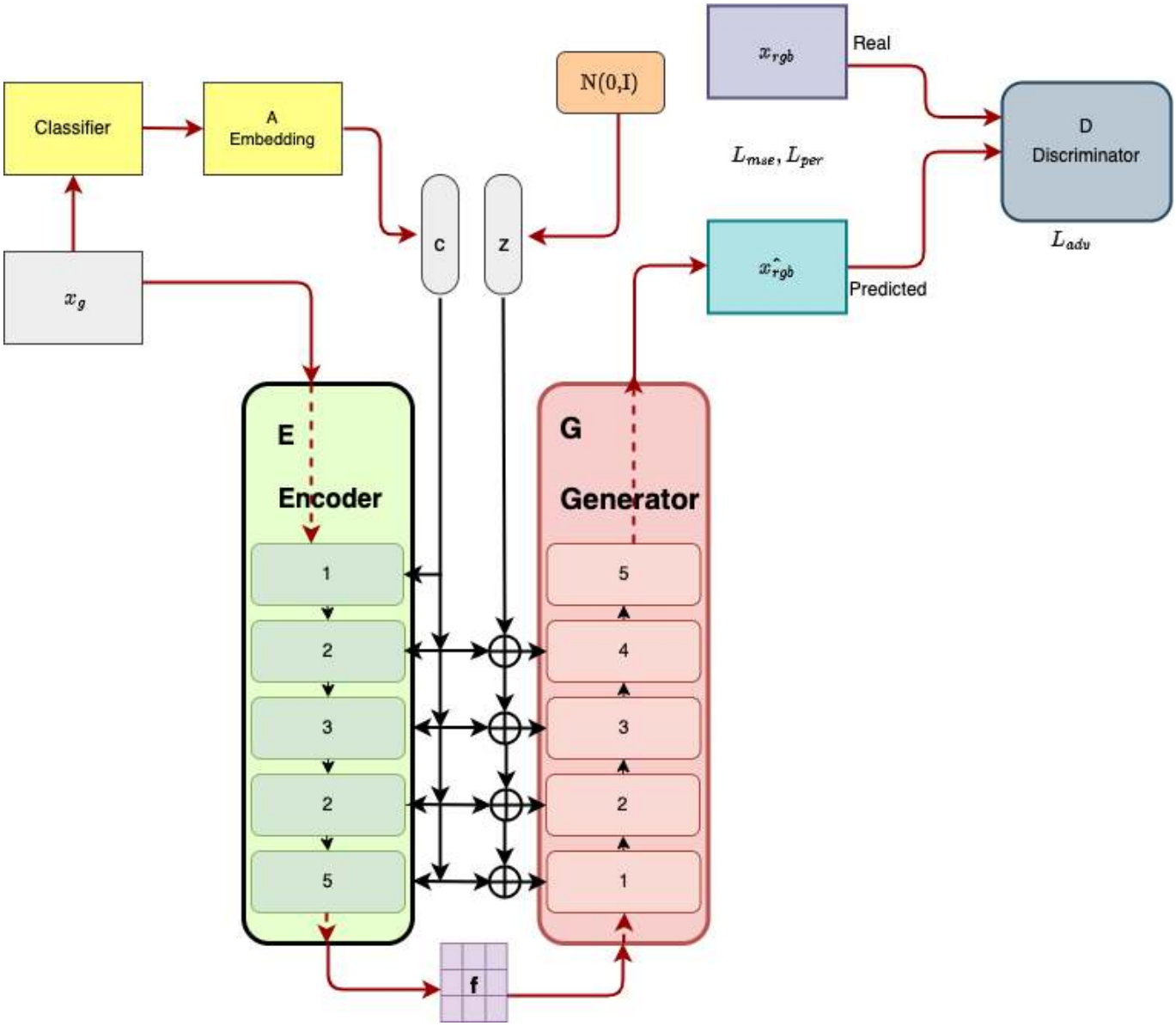


Figure 1: An overview of BigColor. The process extracts the spatial feature f given x_g. This is accomplished using a five residual blocks encoder E. The generator G, with an inverse architecture of E, takes as input the spatial feature f as well as the concatenation of the control variables c and z and synthesizes a color image $\hat{x}_{rgb}$. The class embedding layer, denoted as A, transforms a one-hot class vector, obtained from an off-the-shelf classifier, into a class code c. The encoder-decoder is trained in an adversarial way, hence the need for the discriminator D.

Generator. The inputs to the generator G are the concatenation of the class code c and the random code z, along with f the spatial feature map, and it produces $\hat{x}_{rgb}$, a color image in RGB color space

$$\hat{x}_{\text{rgb}} = G(f;\, c, z).$$

More precisely, the generator is composed of five residual blocks, each of which, with the exception of the last, has as its input the output features of the preceding block, or the feature map f in the case of the first block, as well as the concatenation of z and c. The random code z, of dimension 68×1, is divided into four vectors of dimension 17×1 each, concatenated to the class code c of dimension 128×1, resulting in a vector of dimension 145×1 (Figure 1). As the authors point out in [6], the dimension of the spatial feature map f ($16 \times 16 \times 768$) is larger than the original BigGAN spatially flattened latent code (119×1), which allows the generator to focus on generating colors on the structure information given by the spatial features. In addition, this allows BigColor to process

images with arbitrary, non-fixed sizes. Note that the generator produces an RGB image, which is unusual since most colorization approaches usually infer only the two chrominance components of Lab or YUV spaces. The authors show empirically that generating the RGB image directly is most appropriate, as they train the BigColor generator from the initial weights of BigGAN, which was trained with RGB images.

Discriminator. During training, the method also includes a BigGAN discriminator D to match the distribution of the generated color images $\hat{x}_{\text{rgb}}$ and the ground truth color images x_{rgb}. The discriminator is designed based on BigGAN's discriminator.

3 Training

This study concentrates on presenting the results obtained with BigColor and does not attempt to reproduce the training stage. Although a detailed reproduction of the training process would be interesting, it is reserved for future work. However, to understand the origin of the BigColor model used to generate our results, we present a brief summary of the training steps described by the authors. The key to the network's potential is the joint training of the encoder E, generator G, and discriminator D. This joint training allows G to learn a generative color prior by focusing on synthesizing colors on top of the spatial features f.

Training summary. BigColor was trained using 1.2 million color images from the ImageNet 1K training set [3] where 10% of the original images were excluded as they had a low color score [5]. For each RGB color image x_{rgb}, the corresponding grayscale image x_g was obtained as

$$x_g = 0.2989R + 0.5870G + 0.1140B,$$

where x_g is the grayscale input to the encoder, x_{rgb} is the color ground truth image, and R, G and B are the RGB color intensities of x_{rgb}. The training images were then resized and cropped to a size of 256×256 pixels.

Loss function. BigColor is trained in an adversarial manner. The encoder-generator loss L_G is composed of three terms

$$L_G = L_{G_{mse}} + \lambda_{per} L_{G_{per}} + \lambda_{adv} L_{G_{adv}}, \tag{1}$$

where $L_{G_{mse}}$ is the MSE reconstruction loss, $L_{G_{per}}$ is a perceptual loss based on VGG16 [10] that uses its 1st, 2nd, 6th, and 9th layers' features and $L_{G_{adv}}$ is the class conditional hinge loss [8] defined as

$$L_{G_{adv}} = -D(\hat{x}_{\text{rgb}}, c).$$

The hyperparameters in (1) are λ_{per}= 0.2 and λ_{adv}= 0.03 for balancing. To train the discriminator, the hinge loss [8] is also used

$$L_{D_{adv}} = -\min(0, -1 + D(x_{\text{rgb}}, c)) + \min(0, -1 - D(\hat{x}_{\text{rgb}}, c)).$$

Color augmentation. During training, color augmentation is applied to the real color images fed to the discriminator. This allows to synthesize vivid and more distinguishable colors for the various objects in an image. To do this, the chromaticity values of the images in YUV color space are scaled as $\{Y, U, V\} \leftarrow \{Y, 1.2U, 1.2V\}$.

4 Experiments

The aim of our experiments is to verify the results and limitations of BigColor, paying particular attention to artistic images. We carried out several experiments: colorizing images from different natural image datasets (ImageNet1K [3], CUB [11] and Oxford [9]), colorizing old black-and-white images, and colorizing images of works by various painters. These results are discussed in Section 5.

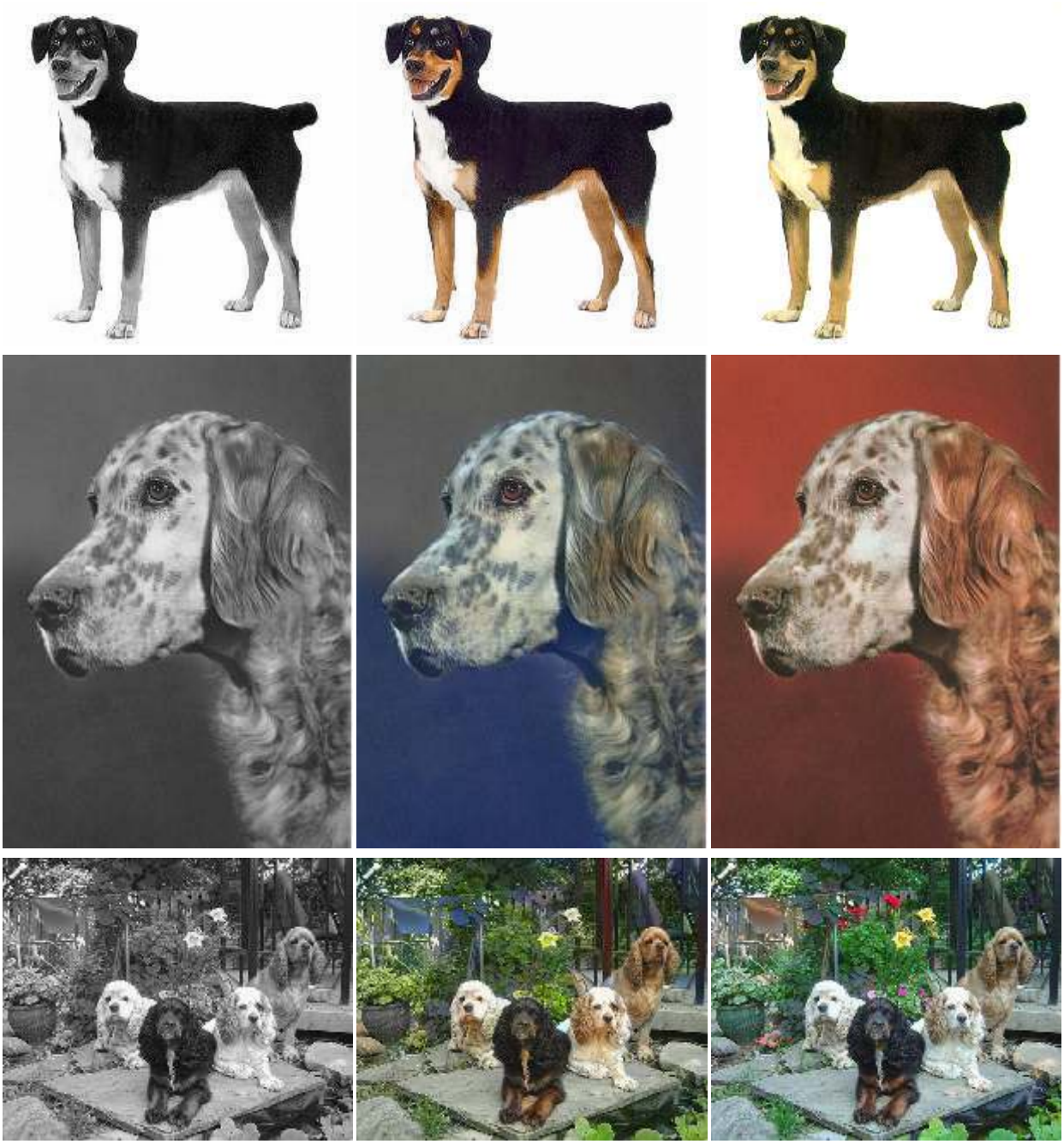


Figure 2: Colorization results on images from the ImageNet1K [3] validation dataset. The first row shows a simple semantic image (a subject and a plain background). The second row shows a more complex image due to the blurred background. The third row shows a scene of higher semantic complexity (multiple objects and inhomogeneous background). From left to right: grayscale input, color inference, and ground truth (GT).

4.1 Natural Image Datasets

Imagenet. The training dataset for BigColor was Imagenet1K [3]. We first tested the performance of BigColor on images from this dataset, since this is the most favorable scenario. The results are shown in Figures 2, 3 and 4. We can observe that for images with simple semantic content (first row in Figures 2 and 3), the colorization results are of high quality, showing vivid and photo-realistic colors. On the other hand, it can be observed that when the semantic content is more complex the algorithm starts showing drawbacks such as color bleeding (e.g. second row in Figure 2), inconsistent colorization for the same object (e.g. second row in Figure 3), bluish stains (e.g. second row in Figure 4), and lack of synthesis of multiple colors within the same image (e.g. third row in Figure 3). Figure 5 shows a close up of these artifacts. Note the different sizes of the images processed by the BigColor method.

Figure 3: Colorization results on images from the ImageNet1K [3] validation dataset. The first row shows a simple semantic image (one subject and a simple background). The second row shows a more complex image due to the camera's viewpoint. The third row shows a scene of higher semantic complexity (perspective of multiple objects). From left to right: grayscale input, color inference, and ground truth (GT).

Figure 4: Colorization results on images from the ImageNet1K [3] validation dataset. The first line shows a simple semantic image (three people and a foggy background), and the second line shows an example of greater semantic complexity (position of subjects and a cluttered background). From left to right: grayscale input, color inference, and ground truth (GT).

Figure 5: Color artifacts close up. Top example: colorization result and a close-up of a color artifact: a rainbow of colors is applied to the background car. Bottom example: colorization result and a close-up of a large bluish inconsistent region.

CUB. The Caltech-UCSD Birds-200-2011 (CUB) [11] dataset is a public bird's images dataset mainly used for classification tasks. BigColor is a class-specific colorization network and it was interesting to check the performance with some of CUB's images. As can be seen in Figure 6, the results are free of artifacts. The method produces satisfactory color outputs, except when many colors are present in the original image, in which case the algorithm cannot reproduce this feature, as can be seen in the second and fourth examples. Note that images from this dataset were not used for training.

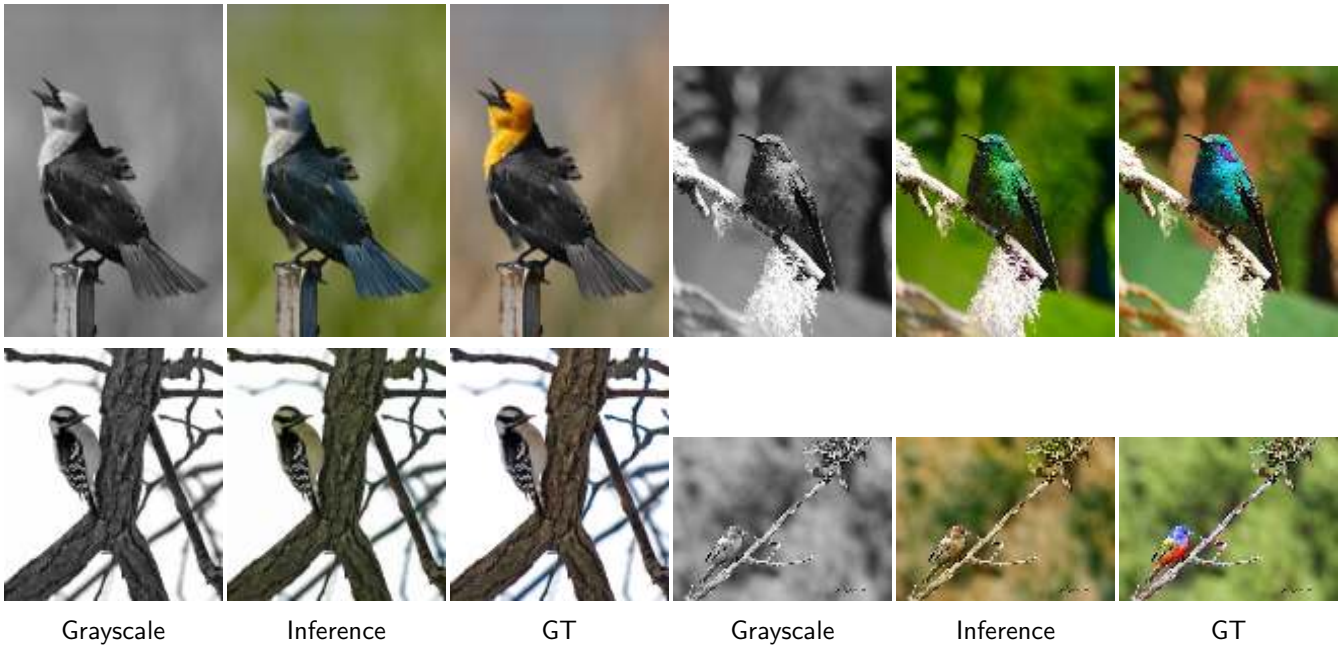


Figure 6: Some results obtained using images from the CUB dataset [11]. These images were selected to represent simple and complex colors present in natural images. From left to right: grayscale input, color inference, and ground truth (GT).

Oxford. The Oxford-IIIT Pet Dataset [9] is a balanced 37 classes dataset. It is used primarily on segmentation tasks. We colorize images from this class-specific dataset to evaluate BigColor, as can be seen in Figure 7. The examples show once again that when fed with images with simple semantic content, BigColor produces high-quality colorized results, not necessarily identical to the ground truth, but highly plausible. Note that images from this dataset were not used for training.

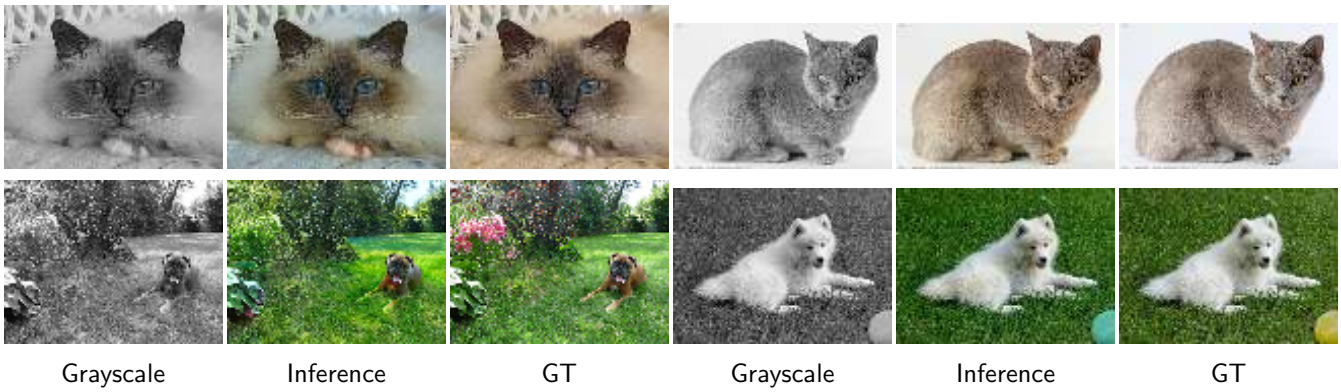


Figure 7: Results using images from the Oxford-IIIT Pet Dataset [9]. From left to right: grayscale input, color inference and ground truth (GT).

4.2 Old Black and White Pictures

Restoration of old photographs is an important application of colorization algorithms. We tested all the old black and white pictures provided by the authors in their GitHub Repository and show a few results in Figure 8. We also observe some differences with the results published in the article. This could be due to the stochastic nature of the algorithm and the selection of results to show. In the first example of Figure 8, we can observe one such difference. In the result shown for this image in [6], the coloration is more homogeneous, and the color difference between the wall and the door is better distinguishable. Some failure cases are shown in the last row of Figure 8. A possible explanation is that neither the semantics of those images nor their quality are well represented by the training set (in this case, Imagenet1K [3]).

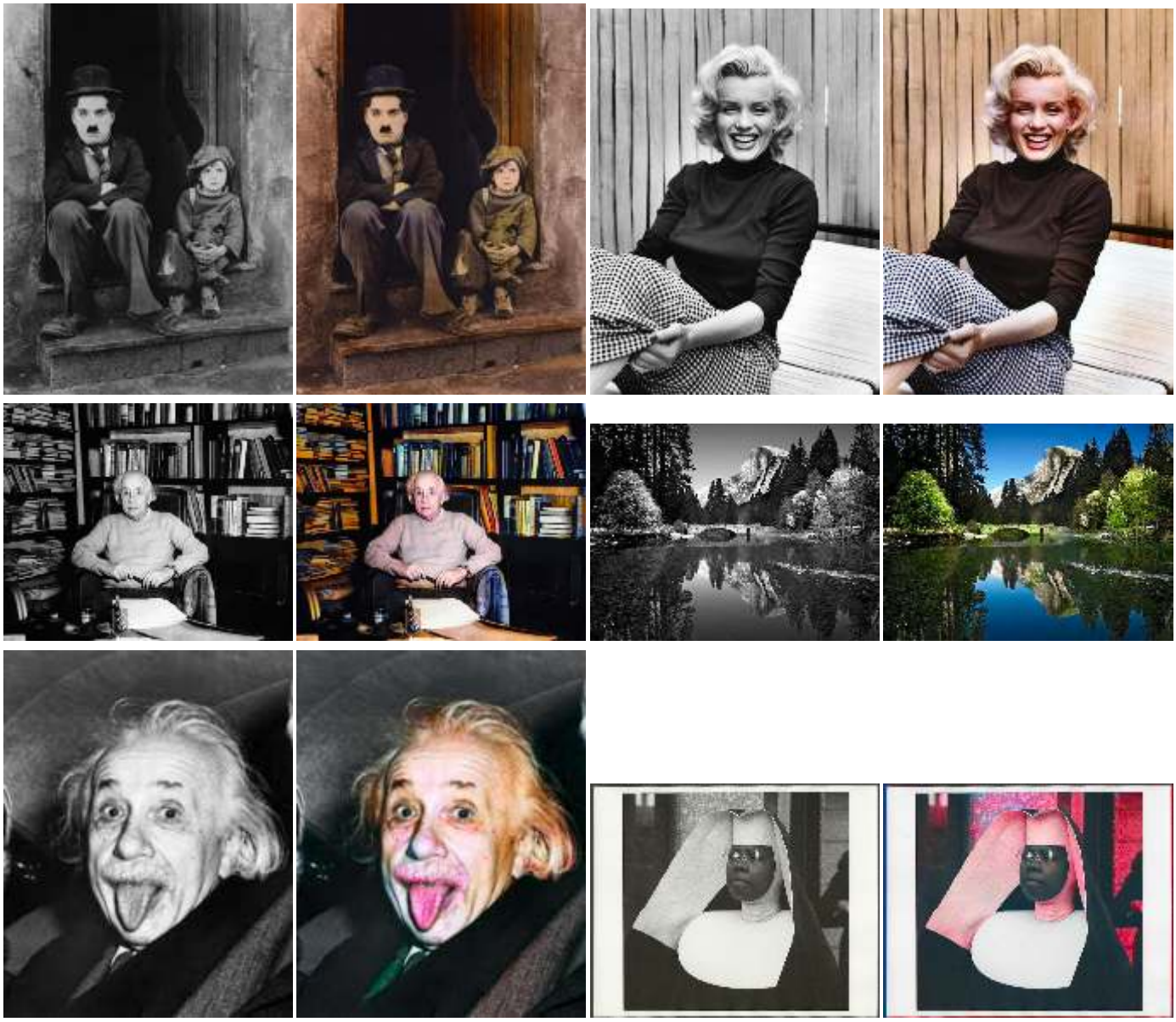


Figure 8: Colorization results of old black and white pictures. The ground truth is unavailable in this case, so only the black-and-white/inference pairs are shown.

4.3 Black and White Photographs of Paintings

This section focuses on colorizing monochrome photographs of abstract paintings. This is of interest since the colors and semantics of abstract paintings are quite different from those in the training set.

Figure 9: Experiments of colorization of black and white photographs of Joaquín Torres García paintings. From left to right: grayscale version of the painting, color inference, and color photo of the painting (GT).

Torres García's paintings. Joaquín Torres García was a Uruguayan painter known for creating the major artistic movements of Modern Classicism and Universal Constructivism. In 1978, 74 of his best paintings were destroyed in a major fire at the Museum of Modern Art in Rio de Janeiro. The color records of these paintings are either of poor quality or did not exist at all. As a part of an ongoing project, we want to restore Torres García's paintings. Figure 9 shows some experiments using color photographs taken from existing paintings that belong to the Joaquín Torres García Museum in Montevideo, Uruguay. In general, we can observe how far these results are from the ground truth, which is to be expected since this algorithm works by default without user interaction and the training set on which BigColor was trained didn't include this type of image. Note, for example, in the first result of Figure 9, how the coloring is generally yellowish. The semantic content of these pictures is completely different from the semantics of natural images. We then ran a second experiment on the same paintings. A photographer took black and white photos of them, the old-fashioned way, and we were able to compare the colorizations of the black and white vs. grayscale inputs. As seen in Figure 10, the results are quite different for the same z and $\sigma = 0.8$. The coloration of the grayscale input tends to be more yellow and red than for the black and white input.

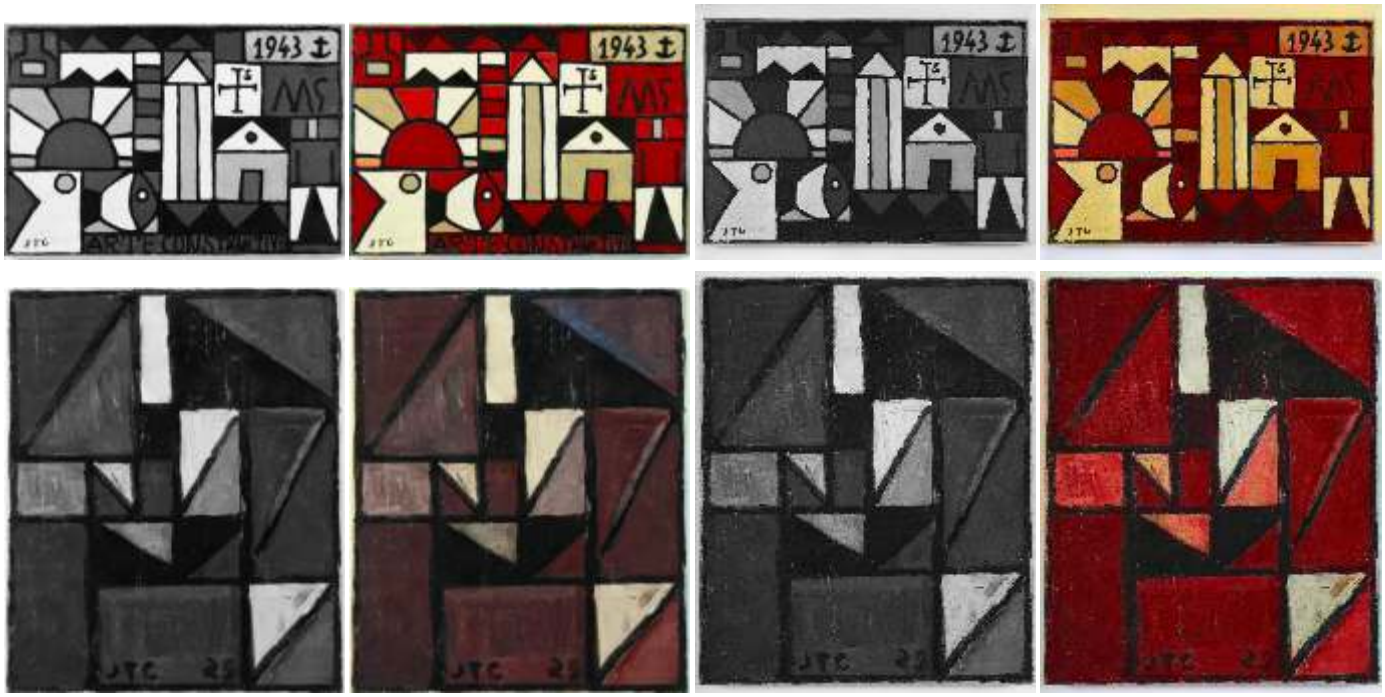


Figure 10: Experiments results of colorization of black and white photographs of Joaquín Torres García paintings vs. grayscale colorization using same z and $\sigma = 0.8$. From left to right: Black and white, color inference, grayscale version of the painting, and color inference.

Other abstract paintings. We also tested BigColor with photographs of paintings by Kandinsky and Mondrian, to provide other examples of coloring abstract artworks. The results can be seen in Figure 11 with similar observations to the Torres García paintings. Note, for example, in the third example, how the discs are colored in the same tone, in a clear inconsistency with the painter's aim. In the fourth example, maybe the coloring was done with some kind of brick wall "in mind", resulting in this general brown color.

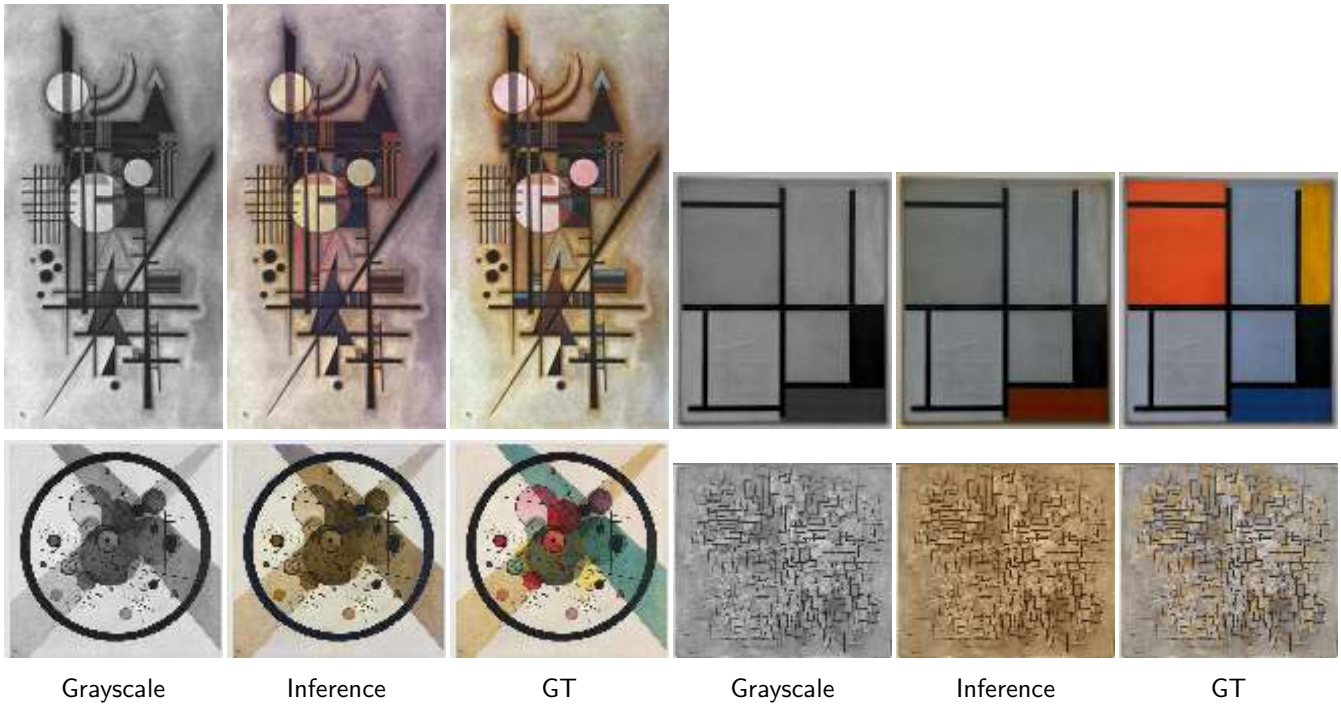


Figure 11: Results of colorizing different paintings by Kandinsky and Mondrian. From left to right: grayscale input, color inference, and ground truth (GT).

4.4 Diverse Colorization

In the authors' GitHub repository, three scripts are available: the first (`colorize_real.py`) produces a single colorization given a grayscale image, the second (`colorize_multi_z.py`) produces several colorizations given a grayscale image and different random z codes sampled from a normal distribution, and the third (`colorize_multi_c.py`) produces different colorizations given a grayscale image and different specific color images provided through their corresponding class code c. The latter option can be considered an example-based colorization method, where the example is supplied through its class code c. These two features (varying z and c) are of great interest in the field of colorization since they allow different results to be obtained for a given grayscale input image. The last two scripts need two explicit inputs in both cases: the random code z and the class code c. While generating different random codes z is straightforward, generating the class codes c requires digging into the authors' codes and isolating the class embedding layer, which is beyond the scope of this work. Therefore, to test at least the diversity of results obtained for different values of z, we have used the first script where at each run, a different random vector is sampled and it is not necessary to explicitly provide the c class code. We observed that the results did not vary much by sampling different z from a normal distribution $\mathcal{N}(0, \sigma^2 I), \sigma = 1$. However, by modifying the covariance matrix to make the variance larger, we can observe more or less variation for different values of σ. Some results are shown in Figure 12. We have considered three values for σ. For $\sigma = 0.8$ the color diversity is low; when $\sigma = 3$, the diversity is noticeable; and for $\sigma = 10$, the generated random codes z are out of distribution which explains the drop in quality. Note that the demo accompanying this mlbrief paper is based on the first script, producing only one coloring result at a time.

5 Discussion and Conclusion

BigColor is an automatic colorization method that works with a BigGAN-inspired encoder-generator architecture focused on synthesizing color upon the spatial structure of the grayscale image encoded

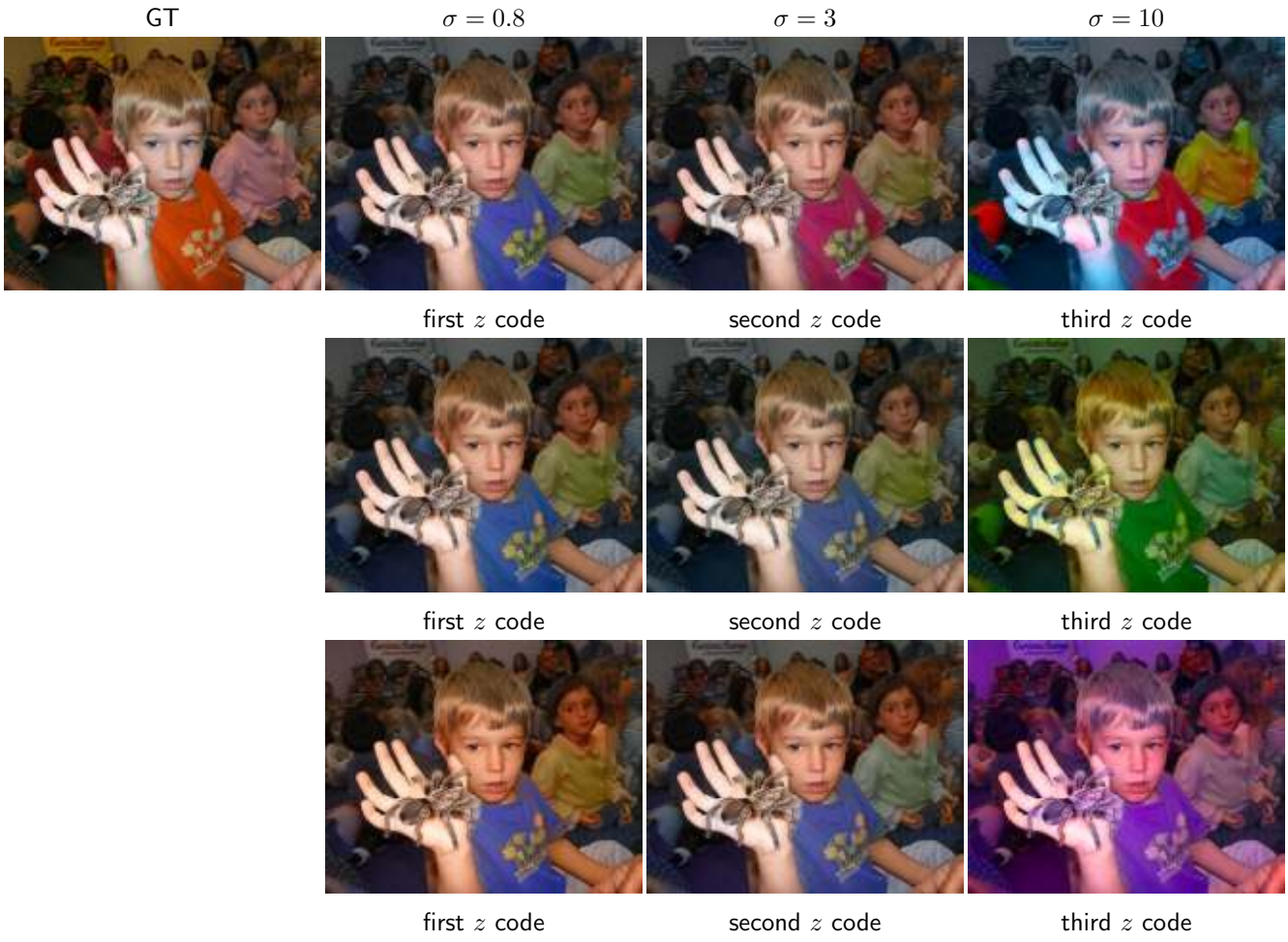


Figure 12: Diverse colorization results. These results were obtained for different random codes z sampled from $\mathcal{N}(0, \sigma^2 I)$. From left to right: $\sigma = 0.8$, $\sigma = 3$ and $\sigma = 10$. Three random codes were generated for each σ value, resulting in three different colorizations.

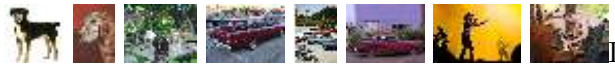
by the encoder module of BigColor. Generally, one can observe that the colors obtained in the colorized images are vivid, diverse, and realistic for simple and complex scenes. In particular, for images with simple semantics, the results are remarkable. Nevertheless, we observe that when scenes become more complex, some of the typical drawbacks of image colorization methods still arise, such as the variety of colors present in the ground truth not being synthesized (flowers in example 3 of Figure 2 and cars in example 3 of Figure 3, birds in examples 2 and 4 of Figure 6, flowers in example 3 of Figure 7). The results are of great quality for old black-and-white pictures, even if different from what can be seen in the published paper, which shows even better results. The results in the last two pictures of Figure 8, where colors inferred are not accurate, are probably caused by very different content of the image (proportion in the image, dresses, gestures) and also the quality of the image might impact (noise, scratches, resolution). Also note that colorization models are typically trained on grayscale/color image pairs, not black-and-white/color image pairs. It is then not surprising to observe a difference in quality when colorizing black-and-white photographs, since the model performs differently on both types of inputs, as shown in Figure 10. And when it comes to paintings, one must remember that BigColor, like many other data-driven colorization approaches, is trained on a dataset of natural images. Hence, the semantics of natural images and abstract paintings is totally different. Moreover, BigColor is a class-specific colorization method where the classification module was trained to classify natural images under 1000 classes. Taking all this into account, the results obtained are not bad, even if, in these cases, one would want to be close to the

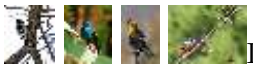
ground truth style of the painting. Here it would be more suitable to use user interaction approaches, and even if out of the scope of this work, it could be interesting to adapt BigColor to style-specific colorization by training the network on a set of artistic images and including a style classification module. BigColor can generate different colorizations for the same grayscale input by providing different code vectors z sampled from a normal distribution $\mathcal{N}(0, \sigma^2 I)$ where σ is set in advance for each image or by providing different class codes c making it extensible to an example-based method where the example image is provided through its class code c.

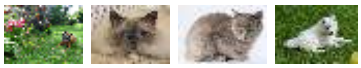
Acknowledgment

We thank the Museo Torres García in Montevideo, Uruguay, for providing us with images of paintings by Joaquín Torres García and for allowing their non-profit use in this article. We thank Luis Sosa and Ignacio Seimanas for the photographic work of taking images of those paintings.

Image Credits


Dataset Imagenet [2]

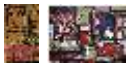

Dataset CUB [11]


Dataset Oxford [9]


Big color GitHub [6]


Flickr


Images from Museo Torres García, Montevideo, Uruguay, Taken by Luis Sosa.


Images from Museo Torres García, Montevideo, Uruguay, Taken by Alejandro Díaz.


Images from Public Domain Collections from picryl.com

References

[1] A. Brock, J. Donahue, and K. Simonyan, *Large Scale GAN Training for High Fidelity Natural Image Synthesis*, in International Conference on Learning Representations, 2019, https://doi.org/10.48550/arXiv.1809.11096.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 248–255, https://doi.org/10.1109/CVPR.2009.5206848.

[3] J. Deng, O. Russakovsky, J. Krause, M. Bernstein, A. C. Berg, and L. Fei-Fei, *Scalable Multi-Label Annotation*, in ACM Conference on Human Factors in Computing Systems (CHI), 2014, https://doi.org/10.1145/2556288.2557011.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Networks*, Advances in Neural Information Processing Systems, 3 (2014), https://doi.org/10.1145/3422622.

[5] D. Hasler and S. Suesstrunk, *Measuring Colourfulness in Natural Images*, Proceedings of SPIE - The International Society for Optical Engineering, 5007 (2003), pp. 87–95, https://doi.org/10.1117/12.477378.

[6] G. Kim, K. Kang, S. Kim, H. Lee, S. Kim, J. Kim, S.-H. Baek, and S. Cho, *BigColor: Colorization Using a Generative Color Prior for Natural Images*, in European Conference on Computer Vision (ECCV), 2022, https://doi.org/10.1007/978-3-031-20071-7_21.

[7] B. Li, Y.-K. Lai, and P. L. Rosin, *A Review of Image Colourisation*, Handbook Of Pattern Recognition And Computer Vision, (2020), pp. 139–157, https://doi.org/10.1142/9789811211072_0008.

[8] J. H. Lim and J. C. Ye, *Geometric GAN*, ArXiv:1705.02894, (2017), https://doi.org/10.48550/arXiv.1705.02894.

[9] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, *Cats and Dogs*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, https://doi.org/10.1109/CVPR.2012.6248092.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision (IJCV), 115 (2015), pp. 211–252, https://doi.org/10.1007/s11263-015-0816-y.

[11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, *CUB-200-2011*, 2022, https://doi.org/10.22002/D1.20098.