INTERACTIVE DEEP COLORIZATION USING SIMULTANEOUS GLOBAL AND LOCAL INPUTS

Yi Xiao [*], *Peiyao Zhou, Yan Zheng*[†]
College of CSEE., Hunan University
Changsha, Hunan Province, China

Chi-Sing Leung
Depart. of EE., City University of Hong Kong
Kowloon, Hong Kong SAR

ABSTRACT

Colorization methods using deep neural networks have become a recent trend. However, most of them do not allow user inputs, or only allow limited user inputs (only global inputs or only local inputs), to control the output colorful images. The possible reason is that it's difficult to differentiate the influence of different kind of user inputs in network training. To solve this problem, we propose a novel deep colorization method allowing inputting global and local inputs simultaneously or individually, which is not supported in previous deep colorization methods. The key steps include designing a neural network model that can appropriately combine the different inputs, and designing an appropriate loss function that can differentiate the influence of different inputs. Experimental results show that our method can magnificently control the colorized results and generate state-of-art results.

***Index Terms*—** Interactive colorization, Deep Convolutional Neural Network, Color Theme, Global and Local Inputs

1. INTRODUCTION

Image colorization refers to the technique that adds colors to monochrome images. Generally speaking, colorization is a ill-posed problem, which does not have a unique solution. To get satisfactory colorized results, two categories of methods have been proposed: user-guided edit propagation and data-driven automatic colorization.

The user-guided edit propagation methods [1, 2, 3, 4, 5, 6, 7, 8, 9] require users to draw colored strokes and propagate the colors across the image by solving a global optimization problem. These methods can achieve impressive colorized images, but often require a very large number of scribbles for images with complex textures. This is because each different color region must be explicitly marked by a different colored stroke, even regions with obvious semantic hints, such as a blue sky or green trees, need to be specified by the user. To address this problem, early data-driven colorization methods proposed to colorize a grayscale image automatically by learning the color hints from one or several exemplar color images which contain similar semantics [10, 11, 12, 13, 14, 15, 16]. Unfortunately, this is a time-consuming work because it's hard to find a suitable exemplar image sometimes.

With the popularity of deep learning, recent data-driven colorization methods using deep neural networks have become a recent trend [17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. Using a large number of input (e.g. gray image) and color image pairs, the deep colorization methods learn a parametric mappings for fully automatic colorization. These methods can generate plausible colorful images in most of the time. However, since a semantic region can have multiple choices of colors, the color and style of result image is single which may not meet user's expectations. For example, users may want a green mountain in spring, but get a yellow one in autumn. They can't control or change the results.

The recent excellent work by Zhang et al [27] combines the advantages of user-guided and data-driven methods. It provides better user controls by taking either global inputs or local inputs in the deep network training. The color of the colorized image can be controlled by a global color histogram, or a few local color points. The user-guide deep colorization [27] can generate plausible colorized images according to the users expectation with only a few inputs. However, it can only allow one kind of inputs in a single model, and does not support simultaneous global inputs and local inputs. More recently, He et al [28] also proposed a deep colorization method to implicitly control the results using exemplars. Deep neural networks are also applied to scribble based color image editing [29] or sketch colorization [30].

Movitation: We argue that supporting simultaneous global inputs and local inputs can provide better control on the output images. The ideal case is that a user can control the overall color style of the image with global color inputs, and meanwhile explicitly assign local colors to regions of interest with local inputs. What's more, the form of global input should be easily set by the users, such as color themes of variable color numbers used in colorful image enhancement [31, 32]. However, supporting multiple kinds of inputs simultaneously in deep networks is not straightforward. It's also difficult to differentiate the influence of different kinds of user inputs in network training. An example is shown in Figure 2, different inputs will affect each other and lead to unnatural results, if the loss function is not appropriately designed. To solve this problem, we propose a novel deep colorization method for coloring grayscale images, cooperated with loss functions that can differentiate the influence of input data, global inputs and local inputs.

The main contributions of this paper are as follows: (1) We propose a deep neural network based colorization method that supports global inputs and local inputs simultaneously or individually, which is not supported in previous deep colorization methods; (2) We enable color themes as global inputs in the context of colorization by preparing suitable training data; (3) We propose a loss function that can differentiate the influences of the global color theme and local inputs without causing artifacts; (4) We conduct sufficient experiments and analysis including user studies, which verify that our methods can better control the colorized images and generate state-of-art results.

[*]The work is supported by the National Key R&D Program of China (2018YFB0203904), NSFC from PRC (61872137, 61502158, 61803150), Hunan NSF (2017JJ3042, 2018JJ3067), China Postdoctoral Foundation (2016M590740), and GRF from Hong Kong (CityU 11259516).

[†]Corresponding author: yanzheng@hnu.edu.cn

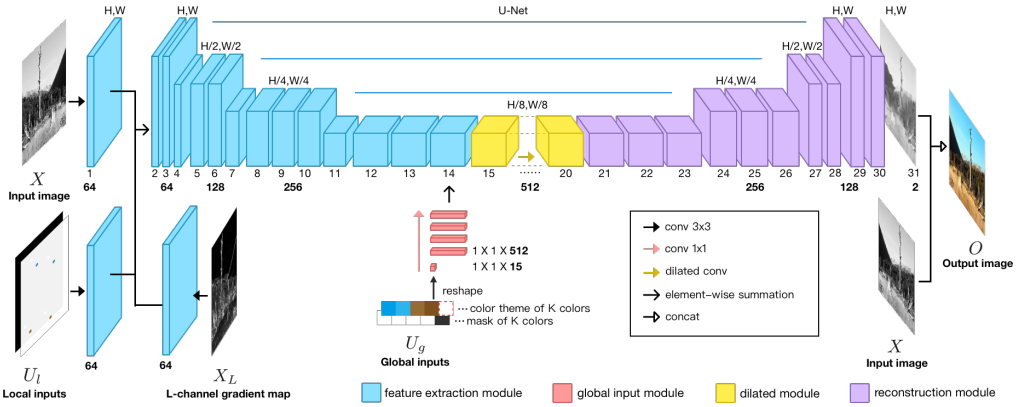


Fig. 1. Our network model for colorization. Convolution layers without detailed instruction in article are using 3×3 kernel. The activation function of each layer is Relu, except Conv31 which uses tanh. Batch normalization is applied in each convolution layer.

2. OUR METHOD

2.1. Problem Formulation

As shown in Figure 1, the input of our deep neural model is consisted of a grayscale image $X \in R^{H\times W\times 1}$, a L channel gradient map $X_L \in R^{H\times W\times 1}$ to avoid artifacts such as color bleedings, a global input tensor $U_g \in R^{1\times K\times 3}$, and a local input tensor $U_l \in R^{H\times W\times 3}$, where H, W are the height and width of the input image respectively, and K is the number of colors in the color theme. The L channel gradient map X_L is generated by processing $X \in R^{H\times W\times 1}$ with the Sobel operator. The output of the model is a tensor $O \in R^{H\times W\times 2}$, which is the *ab* channel in *Lab* space. We aim to train a convolution neural network (CNN), denoted by $\mathcal{F}(X, X_L, U_g, U_l; \theta)$, to approximate the mapping between gray and color, under the constraint of user inputs. Therefore, the colorization problem can be formulated as

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathscr{D}}[\mathcal{L}(\mathcal{F}(X, X_L, U_g, U_l; \theta), U_g, U_l, Y)], \quad (1)$$

where $\mathscr{D}$ denotes the training data set, $\mathcal{L}$ denotes the loss function, and $Y \in R^{H\times W\times 2}$ is ground truth image color. More details of the loss function will be presented in Section 2.4. It should be noticed that we train a single model which can handle global inputs, or local inputs, or simultaneous global inputs and local inputs. Moreover, our loss function is explicitly related to the global inputs and the local inputs. Also, we enable color theme as our global inputs. These are the three key points which **differ** our method from [27].

2.2. User inputs

Global inputs: To control the overall color style of the output, we design the global input as the form of a color theme consisting of K colors, since color theme is simple and straightforward (can freely set by users), which is possibly associated with verbal description [31]. To choose the color number K, we did a user study on 10 volunteers, which shows that the majority of people observe 3 to 5 main colors at the glance of an image, and prefer to use 3 to 5 colors for controlling colorization. So we set K randomly varies in the interval [3,5] in the training data. To prepare training data for the global input, we use the K-mean clustering algorithm to find the K representative colors for each image, which are the most frequently appearing colors in the image. The *ab* channels of the K representative colors $U_g^c \in R^{1\times K\times 2}$ and its mask $M_g \in R^{1\times K\times 1}$ form the global input $U_g = \{U_g^c, M_g\} \in R^{1\times K\times 3}$ by concat. An example of color theme with 4 colors and the corresponding mask are shown in Figure 1.

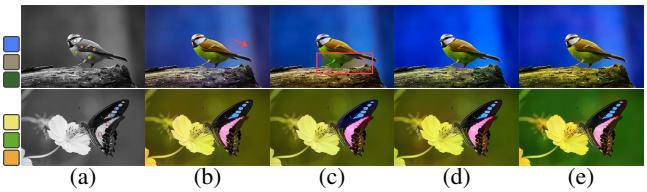


Fig. 2. Ablation study of loss functions. (a) Gray-scale with local inputs and global inputs, (b) With $\mathcal{L}_{GT}$, (c) Without $\mathcal{L}_{Grad}$, (d) Without $\mathcal{L}_{Global}$, (e) Final loss.

Local inputs: We prepare the training data for the local inputs colors $U_l^c \in R^{H\times W\times 2}$ by randomly selecting some points from the *ab* channels of every color image. If there are no samples in certain position, the *ab* values are set to zeros. A one-channel mask $M_l \in R^{H\times W\times 1}$ is also generated to indicate the positions of local inputs. "1" in certain position means there is an input there, while "0" means there are no inputs in the position. Finally, the local input is $U_l = \{U_l^c, M_l\} \in R^{H\times W\times 3}$.

2.3. NetWork model

As shown in Figure 1, our network model uses the popular U-Net structure [33, 25, 27] as a basic model, with extensions to accept multiple kinds of inputs. It is mainly composed of 31 layers, divided to four parts: the feature extraction module(blue), the global input module(pink), the dilated module (yellow) and the reconstruction module (purple).

Feature extraction module: The feature extraction module corresponds to the blue part in Figure 1. It inputs a grayscale image $X \in R^{H\times W\times 1}$, a L channel gradient map $X_L \in R^{H\times W\times 1}$ and a local input $U_l \in R^{H\times W\times 3}$. Initially, X, X_L and U_l are respectively convolved to a tensor of size $H \times W \times 64$. The L channel gradient map is helpful in guiding the network to avoid artifacts and bleeding of the backgrounds, we show examples in Figure 4. These three tensors are merged to a single tensor of size $H \times W \times 64$ by element-wise summation. The merged tensor is then fed to following convolution blocks.

In convolution block 2 to 14, all the convolution kernels are 3×3 and the stride is 1 except in the downsampling layers, where the kernel size is 1×1 and the stride is 2. We use depth-wise convolution in downsampling layers to halve the tensor sizes of the input, and double tensor dimensions with convolution. In conv14, the tensor is $\frac{H}{8} \times \frac{W}{8} \times 512$. We use depth-wise convolution instead of using max-pooling layers to reduce the tensor sizes because it can use less parameters to achieve downsampling.

Global input module: The global input module, the pink part in Figure 1, is one of the characteristics of our method. It takes in the global inputs $U_g \in R^{1\times K\times 3}$, which consists of the *ab* channels of the color theme U_g^c and the corresponding mask M_g. To unify its size with the size of feature extraction module, the global inputs are reshaped to a tensor of size $1 \times 1 \times 512$ and processed by 4 convolution layers with 1×1 kernel size. Then, we add this tensor to the output of feature extraction module by element-wise summation as the input of the dilated module.

Dilated module: The dilated module, shown with yellow color in Figure 1, is an important step to fuse the user inputs and extracted features. Dilated convolution is used to increase the receptive field of filters without increasing the number of parameters. In this module, there are six dilated convolution layers to process input tensor. The input and the output feature tensor size of this module is $\frac{H}{8} \times \frac{W}{8} \times 512$.

Reconstruction module: The feature tensors are then processed by a set of convolution layers and deconvolution layers for *ab* channels reconstruction, shown with purple color in Figure 1. Deconvolution layers use kernels of 4×4 with stride 2, which doubles the size of the tensor and reduce the channels by half. Conv31 is processed by a convolution with kernel size of 1×1, and output a tensor of size $H \times W \times 2$, which combines the input grayscale image $X \in R^{H\times W\times 1}$ to generate the final color image $O \in R^{H\times W\times 3}$.

2.4. Loss function

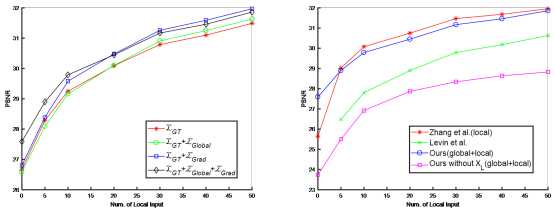
Designing an appropriate loss function is the key part of our work. In [27], the Huber loss between the output image and the ground truth is shown to generate plausible images for either local inputs or global inputs. The Huber loss is given by

$$\mathcal{L}_H(O,Y) = \begin{cases} \frac{1}{2}(O-Y)^2 \text{ for } |O-Y| \le \delta \\ \delta\,|O-Y| - \frac{1}{2}\delta^2 \text{ otherwise,} \end{cases} \tag{2}$$

where δ is the parameter of the Huber loss, which is set to 1. However, it is not enough for simultaneous global and local inputs, the influence of the global color theme is not clear (as shown the Figure 2(b)). Also, only using Huber loss is more like to cause artifacts(the sky in Figure 2(b)). To this end, we design our loss function $\mathcal{L}$ as

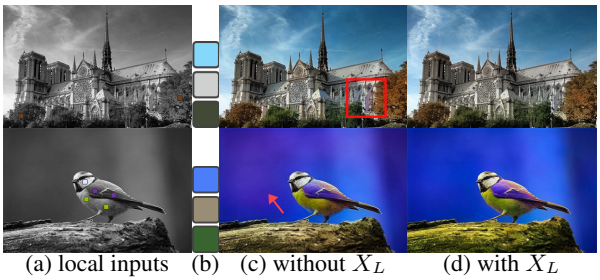
$$\begin{aligned} &\mathcal{L}(\mathcal{F}(X, X_L, U_g, U_l; \theta), U_g, U_l, Y) \\ &= \alpha_1 * \mathcal{L}_{GT} + \alpha_2 * \mathcal{L}_{Global} + \alpha_3 * \mathcal{L}_{Grad}. \end{aligned} \tag{3}$$

where $\mathcal{L}_{GT}$ is the Hubber loss of the output image and the ground truth; $\mathcal{L}_{Global}$ is the Hubber loss of the output image and the K-color map by decoding the ground truth with its K representative colors; $\mathcal{L}_{Grad}$ is the mean squared error of the Sobel gradient of the output image and the ground truth; $\alpha_1, \alpha_2, \alpha_3$ are three parameters to weight the influence of three parts. We empirically set $\alpha_1 = 0.9, \alpha_2 = 0.1, \alpha_3 = 10$ in all our experiments. As shown in Figure 2(e), the defined loss not only can remove the color overflow, but also can remove the strange color on background and promote colors in color theme to better reflect in the output.



(a) (b)

Fig. 3. Average PSNRs of testing images with different loss functions and methods. Our method uses both global and local inputs.



(a) local inputs (b) (c) without X_L (d) with X_L

Fig. 4. Effect of the L channel gradient map. (a) Grayscale images with local inputs. (b) Global inputs. (c) Results without gradient maps. (d) Results with gradient maps.

3. EXPERIMENT

We implemented our model on a NIVIDIA GTX1080Ti GPU with TensorFlow. We randomly choose 1000 images from ImageNet dataset as testing set, and use the remaining images in ImageNet dataset plus 150,000 outdoor images in Places to train our model. The number of local points is uniformly distributed in [10,50]. We train the model using a batch size of 16 for 300,000 iterations, using about five days. In the training, 90% training data are provided with simultaneous global inputs and local inputs, 5% training data are provided with only global inputs, and the remaining 5% training data are provided with only local inputs. The coloration time for a 256×256 and a 512×512 image is about 0.0230 seconds and 0.0748 seconds, respectively. Therefore, our colorization method can work in real-time for normal size images.

3.1. Ablation Study of Loss Functions

We conduct a visual comparison (Figure 2) and a numerical comparison (Figure 3(a)) for different combinations of loss functions. Removing either component of our loss function causes artifacts ("dirty" colors in the background of first row of Figure 2(b)(c))), or degrades the influence of the color theme(Figure 2(d)). For the numerical comparison, the PSNRs of the final loss function are higher than other loss functions, but are slightly lower than the one without $\mathcal{L}_{Global}$ when the number of local points is larger than 20. This is because our final loss function includes the global color theme loss, the difference between the output and the K-color map, which slightly drives our output further from the ground truth.

3.2. Effect of L channel gradient map

The inputs of our method include a L channel gradient map X_L, which can provide the network with gradient information of the

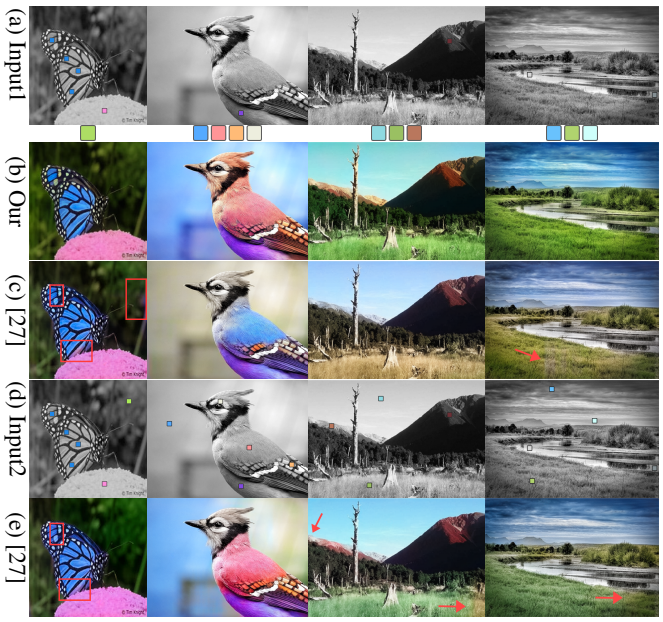


Fig. 5. Comparison with the state-of-art [27]. (a) Grayscale images and user inputs. (b) Our results. (c) [27]'s results. (d) An other input for [27] by inserting the colors in our color theme to local regions. (e) [27]'s results with input in (d).

grayscale image to make the color diffusion smoother, as shown in Figure 4. Results in column (c) without the gradient map show abnormal colors. Obviously, the results with the gradient map can improve the quality of colorizaiton as show column (d). We also evaluate the PSNRs of the results with and without X_L, as shown in Figure 3(b). The results without X_L (the purple line) have much lower PSNRs compared to those with X_L (the blue line).

3.3. Comparison to State-of-art

Visual: We compare our method with the state-of-art user guide method [27] in Figure 5. Our method uses global inputs and local inputs simultaneously, while [27] uses local inputs only. Note that, since our method takes more inputs (global color theme) than [27], we prepare another input (row (d)) for [27] by inserting the colors in our color theme to local regions, for a more fair comparison. In this case, the color numbers of input in row (d) are equal to our simultaneous global inputs plus the local inputs. As shown in Figure 5, [27]'s method is more likely to cause color overflows (the Butterfly image in Row (c)(e)), or fail in diffusing the colors consistently in to the whole image (the Bird image in Row (e), the Mountain image in Row (e), and the Grass image in Row (c)(e)). This phenomenon implies that more local color points is required to generate better results. In contrast, our method generates plausible results without artifacts such as color overflows and color inconsistences.

Numerical: The numerical comparison of our method and [27] is shown in Figure 3(b). In this comparison, the color themes and local inputs are randomly extracted from the ground truth of testing images, as done in the training phase. Our method is better than the classical optimization method [1], but a little worse than [27]. This is due to the global input loss, which makes the output a little further from the ground truth. Note that, this is not a big problem, since the purpose of colorization is not to recover the original image, which may not exist.

Fig. 6. More results. (a)An example using three themes. (b) Some examples generated by users in user study.

User stuty: We further conduct a user study to compare our method and [27]. We randomly selected 20 results of both methods from the testing set. 20 volunteers are invited to observed each pair of images for ten seconds and to choose one image which looks more natural or real. Results show that 61% of our results are chosen to be better. Among the volunteers, 90% of volunteers think our results are better (choose more than 10 our results). The volunteers are also invited to experience our method with two kinds of inputs and the method in [27] with local inputs (both implemented with similar GUIs). Results show that 70% of the volunteers think our interactive mode is more convenient and efficient.

Then, we interviewed 5 volunteers about the feelings of using the two softwares. Three people said that global inputs are very quick and easy to use, and it can colorize the background very well. For unsatisfactory regions, they can use local inputs to adjust. Using the combination of two inputs, they can quickly colorize a desired image. One volunteer said that he personally prefers to adjust each detail directly with local inputs. Another volunteer said that both methods can produce expected results. Therefore, our method is what users expect, and it does improve colorizing efficiency.

3.4. Evaluation of Global Inputs

We also evaluate the influence of the global color themes. Figure 6(a) show an example using different color themes. As we can see, the color theme faithfully control the colorized results. We also make a user study to evaluate the impact of color themes. We randomly selected 20 gray images from the test set. To colorized these images, we invite 5 volunteers, each of which is required to colorize 4 images by a color theme within one minute. Some results are shown in Figure 6(b). The results are organized into a test questionnaire where each image is provided four options. One option is the color theme used for colorization, and the other three are interference items. Another 20 volunteers are required to choose the correct theme for each image within 15 seconds. Results show that the average correct rate has reached 85%. The highest correct rate is 95%, and the lowest correct rate is 65%. The volunteers interviewed also said that the color themes are naturally reflected in the images.

4. CONCLUSION

In this paper, we propose an CNN based interactive colorizing methods supporting either global inputs or local inputs or both inputs simultaneously. By designing a suitable loss function, our model can differentiate the impact of different inputs. Experiments and user studies show that users can better control the plausible images without artifacts using our method. In the future, we would like to extend our method to other type of images or videos.

5. REFERENCES

[1] Anat Levin, Dani Lischinski, and Yair Weiss, "Colorization using optimization," in *SIGGRAPH*, 2004, pp. 689–694.

[2] T. Horiuchi and H. Kotera, "Colorization for monochrome image with texture," in *Color Imaging Conference*, 2005, pp. 245–250.

[3] Yi Chin Huang, Yi Shin Tung, Jun Cheng Chen, Sung Wen Wang, and Ja Ling Wu, "An adaptive edge detection based colorization algorithm and its applications," in *ACM MM*, 2005, pp. 351–354.

[4] Liron Yatziv and Guillermo Sapiro, "Fast image and video colorization using chrominance blending," *Image Processing, IEEE Transactions on*, vol. 15, no. 5, pp. 1120–1129, 2006.

[5] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying Qing Xu, and Heung Yeung Shum, "Natural image colorization," in *Eurographics Conference on Rendering Techniques*, 2007, pp. 309–320.

[6] M. Kawulok and B. Smolka, "Competitive image colorisation," in *Proceedings of 17th International Conference on Image Processing*, 2010, pp. 405–408.

[7] Chen Yao, Xiaokang Yang, Li Chen, and Yi Xu, "Image colorization using bayesian nonlocal inference," *Journal of Electronic Imaging*, vol. 20, no. 2, pp. 023008–1–023008–6, 2011.

[8] Michal Kawulok, Jolanta Kawulok, and Bogdan Smolka, "Textural features for scribble-based image colorization," *Computer Recognition Systems 4*, vol. 95, pp. 269–278, 2011.

[9] Xiaowu Chen, Dongqing Zou, Qinping Zhao, and Ping Tan, "Manifold preserving edit propagation," *ACM TOG*, vol. 31, no. 6, pp. 132, 2012.

[10] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34–41, 2001.

[11] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller, "Transferring color to greyscale images," in *Conference on Computer Graphics and Interactive Techniques*, 2002, pp. 277–280.

[12] Youngha Chang, Suguru Saito, Keiji Uchikawa, and Masayuki Nakajima, "Example-based color stylization of images," *Acm Transactions on Applied Perception*, vol. 2, no. 3, pp. 322–345, 2005.

[13] Yong Sang Chia, Shaojie Zhuo, Raj Kumar Gupta, Yu Wing Tai, Siu Yeung Cho, Ping Tan, and Stephen Lin, "Semantic colorization with internet images," *ACM TOG*, vol. 30, no. 6, pp. 1–8, 2011.

[14] Raj Kumar Gupta, Yong Sang Chia, Deepu Rajan, Ee Sin Ng, and Zhiyong Huang, "Image colorization using similar images," in *ACM MM*, 2012, pp. 369–378.

[15] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng, "Intrinsic colorization," *ACM Transactions on Graphics (SIGGRAPH Asia 2008 issue)*, vol. 27, no. 5, pp. 152:1–152:9, December 2008.

[16] Yuji Morimoto, Yuichi Taguchi, and Takeshi Naemura, "Automatic colorization of grayscale images using multiple images on the web," in *SIGGRAPH*, 2009, pp. 1–1.

[17] Zezhou Cheng, Qingxiong Yang, and Bin Sheng, "Deep colorization," in *ICCV*, 2015, pp. 415–423.

[18] Tung Duc Nguyen, Kazuki Mori, and Ruck Thawonmas, "Image colorization using a deep convolutional neural network," *CoRR*, vol. abs/1604.07904, 2016.

[19] Matthias Limmer and Hendrik P. A. Lensch, "Infrared colorization using deep convolutional neural networks," in *ICMLA*, 2017, pp. 61–68.

[20] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa, "Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification," *TOG*, vol. 35, no. 4, 2016.

[21] Richard Zhang, Phillip Isola, and Alexei A. Efros, "Colorful image colorization," in *ECCV*, 2016, pp. 649–666.

[22] Domonkos Varga and Tams Szirnyi, "Fully automatic image colorization based on convolutional neural network," in *ICPR*, 2017, pp. 3691–3696.

[23] Yili Zhao, Dan Xu, and Yan Zhang, "Retracted chapter: Image colorization using convolutional neural network," in *IGTA*, 2016, pp. 238–244.

[24] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich, "Learning representations for automatic colorization," in *ECCV*, 2016.

[25] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[26] Kevin Frans, "Outline colorization through tandem adversarial networks," *CoRR*, vol. abs/1704.08834, 2017.

[27] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros, "Real-time user-guided image colorization with learned deep priors," *ACM TOG*, vol. 9, no. 4, 2017.

[28] Mingming He, Dongdong Chen, Jing Liao, Pedro V. Sander, and Lu Yuan, "Deep exemplar-based colorization," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 47:1–47:16, July 2018.

[29] Yuki Endo, Satoshi Iizuka, Yoshihiro Kanamori, and Jun Mitani, "Deepprop: Extracting deep features from a single image for edit propagation," *Computer Graphics Forum*, vol. 35, no. 2, pp. 189–201, 2016.

[30] Patsorn Sangkloy, Jingwan Lu, Chen Fang, FIsher Yu, and James Hays, "Scribbler: Controlling deep image synthesis with sketch and color," *CVPR*, 2017.

[31] Baoyuan Wang, Yizhou Yu, Tien-Tsin Wong, Chun Chen, and Ying-Qing Xu, "Data-driven image color theme enhancement," *TOG*, vol. 29, no. 6, pp. 146:1–146:10, December 2010.

[32] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein, "Palette-based photo recoloring," *ACM TOG*, vol. 34, no. 4, jul 2015.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015, pp. 234–241.