Surpassing Traditional Image-Colorization Problems with Conditional Generative Adversarial Networks

Vishnu Teja Yalakuntla, Rahul Kanojia, Kushagra Chauhan, Rohit Gurnani and Mukesh A. Zaveri

Abstract Color helps to understand the semantic information of the image more accurately and reveals a lot more details which grayscale images cannot. By looking at an image, humans can automatically segment different objects present in an image making it easier for us to color an image. We propose a completely automated system to colorize grayscale images which learns to segment and color images in a realistic manner. We leverage the recent advancements in deep learning, Generative Adversarial Networks and improved cost functions, to overcome the problems of traditional Convolutional Neural Networks with image colorization. Given the unconstrained nature of the problem, we propose this algorithm to make a colorization model that achieves realistic colorizations. We have experimented different deep network architectures with various training algorithms and cost functions to come up with this network where we can clearly see realistic colors for given gray scale image and differentiate the characteristics of generative adversarial network from a traditional convolutional neural network.

Keywords Conditional generative adversarial network · Skip connections · Localization · Image colorization

V. Teja Yalakuntla · R. Kanojia · K. Chauhan · R. Gurnani · M. A. Zaveri (✉)
Computer Engineering Department, Sardar Vallabhbhai National Institute of Technology, Surat 395007, Gujarat, India
e-mail: mazaveri@coed.svnit.ac.in

V. Teja Yalakuntla
e-mail: yvtheja@gmail.com

R. Kanojia
e-mail: kaanorahul8@gmail.com

K. Chauhan
e-mail: chauhankushagra1@gmail.com

R. Gurnani
e-mail: rkgi10@gmail.com

1 Introduction

Colorization is a task that is very simple for humans; with impressive imagination skills built on a composite platform of experiences and the brains power to abstract, the complexity of the problem is not even realized. With a machine, without these human skills, coloring an image without any manual input is a increasingly challenging task. Image colorization can make an image lively, but current techniques demand major user-interaction and involvement. Moreover, users can find it difficult to provide consistent and coherent color models/scribbles.

Generally classifying, there have been two directions in which colorization methods have been developed. The first approach focus more to reduce the effort and speed up the process of using tools with a need for human interaction, thus lowering the cost of colorization process. The later focus mainly to eliminate the user inputs completely from the system. Our objective is to eliminate user involvement completely, while improving the end result to be consistent, coherent and as natural looking as possible, doing it in a way thats computationally feasible.

Features are rich piece of information extracted from images in terms of numerical values that are difficult to understand and correlate by human. Several features can be extracted from a single image, such as corners, edges and blobs. The extraction of these features are the major overheads in processing an image.

To color an image accurately, different types of features on different scales are needed. Local features like, segmentation information and global features like, weather on the image are necessary to color an image. Global image features describes an image as whole whereas local feature represents more of pixel level information. Global features are generally used in image retrieval, object detection and classification, while the local descriptors used for object recognition/identification. A combination of both has served as a good input for a number of applications in computer vision.

In recent times, Convolutional Neural Networks (CNN) have served as an excellent tool to extract both global and local features, with various applications like object class recognition [2] and image retrieval [3]. Recent systems have used different approaches to embed and merge these features into the pipelines. Maintaining completely different networks to extract global and local features [4], constructing a new 3D layer called hypercolumns [1] by merging higher layers for semantic details and lower layers for localization details are significant examples. In both ways, the results have been improved and merging of global and local features played a crucial role in achieving them.

Some of the approaches like [5–11], are more of semi-automatic way, in which some or the other form of user input is required such as sparse inputs in the form of colors, or regionality of colors and color histograms. These systems require human intervention, but usually have more accurate results. For example, the approach followed in [5] requires a huge amount of color details from the user, which enables the system to generate variety of vibrant color images. However, designing a system which is completely independent and which wouldn't require any user inputs is a promising problem.

2 Approach

2.1 LAB Colorspace

In 1976, CIE defined a new colorspace named LAB. It expresses color as three numerical values L* represents the lightness channel while a* and b* represents the color channels. The three channels can represented in a three dimensional space where each channel is located at one of the axis. The vertical L* axis ranges from 0–100 depicting lightness. The other (horizontal) axes are represented by a* and b*. The a* is a green-red component being green at one extremity of axis (represented by −a) and red at other (+a). Similarly, b* is blue-yellow component. In practice, the values of horizontal axis ranges from −128 to +127 (256 values).

In colorizing grayscale images, the LAB colorspace is preferred over RGB as LAB encoded image has one layer of grayscale (lightness channel L*) and have packed three color layers namely RGB, into two (a* and b*). This means we can utilize raw grayscale images in colorization and only need to predict only two channels. Additionally, LAB colorspace encompasses the entire spectrum, resulting in more realistic color predictions.

2.2 Generative Adversarial Networks

Firstly proposed by Ian J. Goodfellow [12], Generative Adversarial Networks is an example of adversarial learning from generative models. A GAN [13] comprises of two competing neural network models, G a generative model that learns to generates new data and a discriminative model D that computes the probability that whatever sample data G produces to feed it, whether it is from training data or from G itself.

The two models play a minimax game against each other—minimizing a maximum possible loss that one of two players can make. G is fed with noise and it generates new data based on the discriminations of discriminator; every time G generates a new sample, D will try to determine if the sample is from the model or the training set. Ultimately G will learn to create data which is nearly impossible for the discriminator D to distinguish.

To reproduce images that looks similar to images in the sample, generator G is fed with noise z. The mapping can be represented as G(z) where G is a differential function. The output is then fed to discriminator D along with the ground truth image x represented as D(G(z)) and D(x) respectively. D is trained to maximize the probability of assigning correct labels while simultaneously G is trained to minimize the same. This can be expressed mathematically using a value function V(G, D) [12] as shown in Eq. 1.

$$\min_{\mathbf{G}} \max_{\mathbf{D}} V(G, D) = E_{x \approx p_{data}(x)}[log D(x)] + E_{z \approx p_z(z)}[log(1 - D(G(z)))] \quad (1)$$

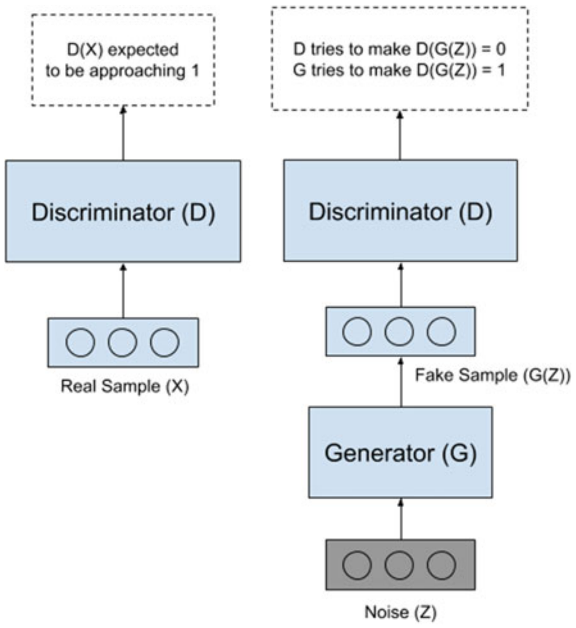


Fig. 1 Basic working of GAN [1]

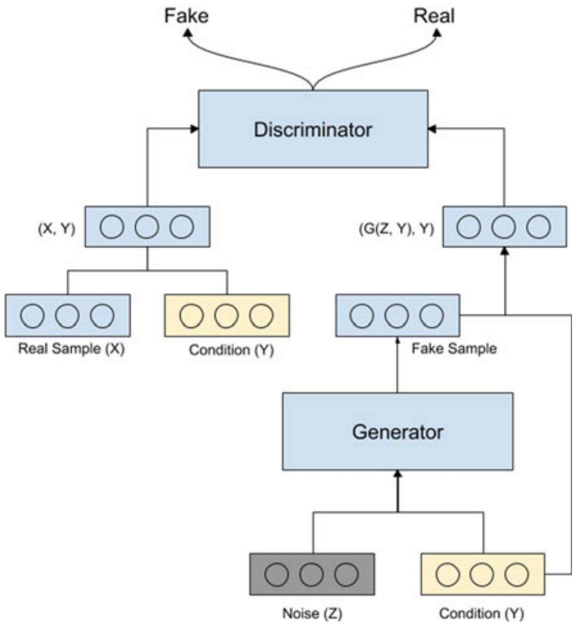
In value function V(G, D), the first term is entropy that the data x is from real distribution (pdata(x)) and is fed to discriminator. Discriminator tries to maximize D(x) to 1. The second term is entropy that the data from random noise p(z) is input to generator, which generates a sample using z and feed it to discriminator. Discriminator tries to maximize D(G(z)) to 0 (i.e. the log probability that the data from generator is fake and therefore is equal to 0). Overall discriminator tries to maximize function V. On the other hand, generator tries to minimize this function so that both real and fake images become indistinguishable. Figure 1 depicts the basic algorithm for a GAN.

2.3 *Conditional GAN*

In ordinary GANs, the input to generator is only the randomly generated noise z, and for that reason we have no control in directing the data generation process. Mirza et al. [14] extended idea of GAN and proposed a conditional model called Conditional Generative Adversarial Networks (CGAN). Using the conditional version of GAN, one can direct the data generation process of generator and restrict it to a desired subset.

Basically, in CGAN both generator and discriminator are conditioned on additional information y, which could be any auxiliary information such as class labels or data from other modalities [14]. The conditioning is performed by feeding y into both discriminator and generator as an additional input layer. We can think of this

Fig. 2 Simple structure of conditional GAN



information y as a particular setting or *mode* in which the model will be working. The consequence of making model work in a particular *mode* is that we can directly control the output of generator. Altogether it's like restricting generator in it's output and discriminator in it's input.

As a result of conditioning input y, the generator function G(z) now becomes G(z, y) and similarly discriminator function D(x) becomes D(x, y) where z is random noise and x is an input from dataset. The objective function will be transformed to:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} V(G, D) = E_{x \approx p_{data}(x)}[log D(x, y)] + E_{z \approx p_z(z)}[log(1 - D(G(z, y)))] \tag{2}$$

Figure 2 illustrates basic structure of conditional GAN. In contrast with the conventional GANs, we now has an additional input layer in both generator and discriminator network.

2.4 Network Architecture

Colorization with CGANs have shown success in overcoming the problem of picking average colors from the training dataset which is a major problem faced by a CNN network. We propose training a CGAN with condition as a gray scale along with the noise making generator produce LAB colorspace images. Discriminator is trained to classify between fake and real images with LAB image along with the condition, a grayscale image, as input. Figure 3 represents the overview of the network.

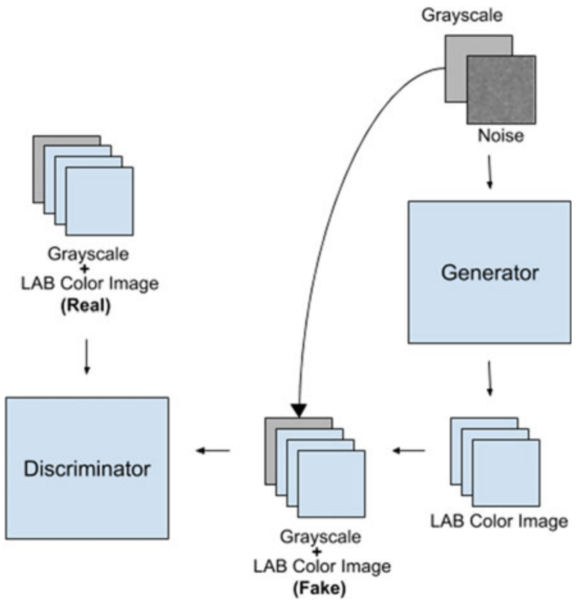

Grayscale
Noise
Generator
Grayscale + LAB Color Image (Real)
Discriminator
Grayscale + LAB Color Image (Fake)
LAB Color Image

Fig. 3 Overview of GAN for image colorization

Initially, generator network produces fake color images which can be easily differentiated from real color images making the job too easy for the discriminator. Receiving strong gradients at this phase, generator starts to get better at producing fake color images making the job gradually difficult for discriminator. As the generator gets fully trained, it produces fake color images which are almost indistinguishable from the real color images making it impossible for discriminator to differentiate between fake and real color images. As a result, discriminator makes random guesses for predictions and its loss keeps oscillating around 0.5 making the GAN fully trained.

Generator is a convolutional neural network with 17 convolutional layers and zero pooling layers. Strides of length more than one is used instead of pooling layers to make network to learn the effective downsampling. As the number of layers increases, the localization, or pixel information is lost and upper layers are more likely to learn global features [15]. To effectively color the images pixel level information is very much need in the upper layers of the CNN. As the Generator predict colors on pixel level, the localization information present in the lower level layers of the network is crucial to segment different objects in the image and draw accurate boundaries for the colors. To assist the network to pass this localization information to higher level layers, skip connections are added in the generator network (Fig. 4).

Initially filter maps are downsampled gradually with stride length of two and Batch normalization with Leaky ReLu as an activation function is used for all convolutional layers except for the last layer of the generator which uses hyperbolic tangent function as the activation. At the skip connections, the filter maps are upscaled by resizing them with bilinear interpolation technique and added to the corresponding upper

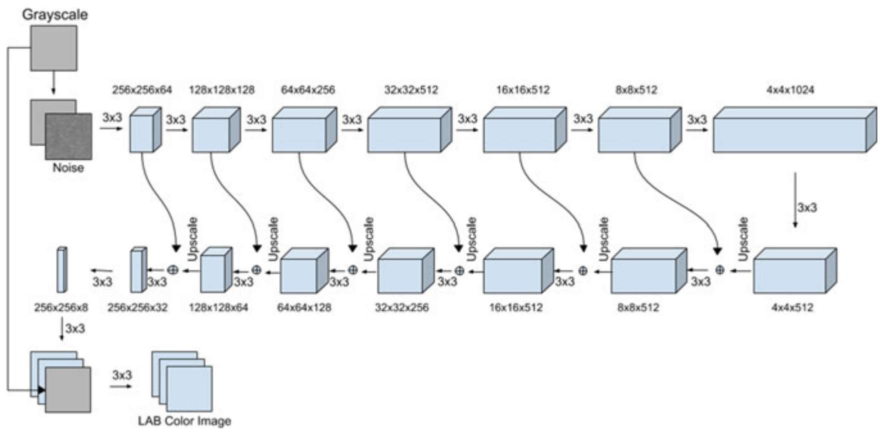


Fig. 4 Generator network

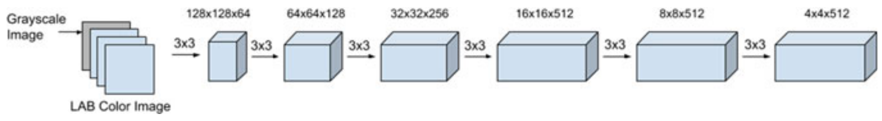


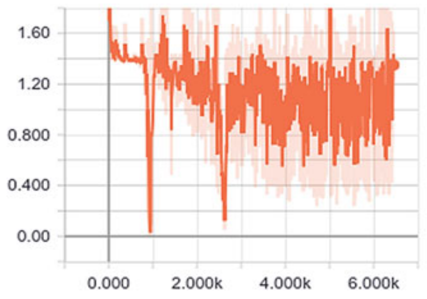
Fig. 5 Discriminator network

layers of the network. Figure 5 shows the architecture of the generator network. Skip connections play a very important role in passing the localization information to upper layers of the network and it is found that the generator is incapable in producing color samples without them. Even concatenating the grayscale image to filter maps of last few layers couldnt help overcome the problem. Hypercolumns [5] can also be used to preserve the localization information till the output layer [12]. Concatenating all the filter maps and adding convolutional layers on top of this huge number of filter maps require a lot of memory and computational power. By downsampling the image for only four times, a hypercolumn containing 932 filter maps has to be generated which couldnt be trained on 4GB Nvidia Quadro M1000M GPU. However, skip connections are known to perform better compared to hypercolumns.

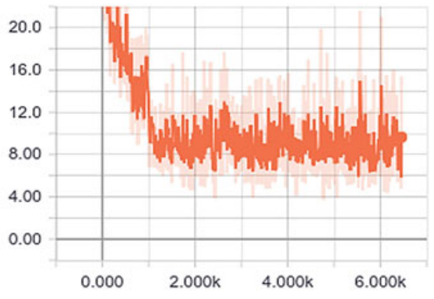
Discriminator is a simple classification network which tries to classify based on the condition and color image fed to it. Figure 5 shows the architecture of the discriminator network. Discriminator network can be very easily trained and once it becomes very confident about its predictions, the gradients for the generator network vanishes. To avoid this, discriminator is trained once for every three times the generator is trained.

$$\min_{\theta_G} J^{(G)^*}(\theta_D, \theta_G) = \min_{\theta_G} -E_z[log D(G(z))] + \lambda \, ||G(z) - y|| \tag{3}$$

The above cost function explained in [14] has been used with λ as 100. From the loss plots of both generator and discriminator networks, the characteristics of a



(a) Discriminator Loss Plot

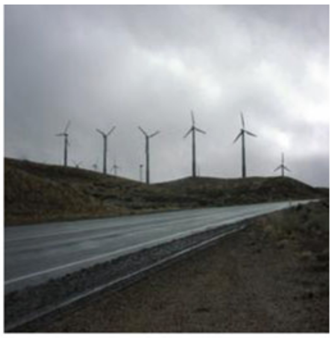


(b) Generator Loss Plot

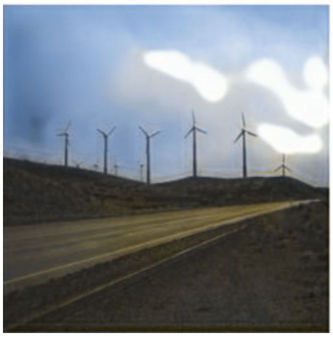
Fig. 6 GAN loss plots

GAN can be clearly observed. Figure 6a shows the generator loss and Fig. 6b shows the combined loss for fake and real images of discriminator during the training. As the generator network gets better at producing fake color images, as the loss of the generator decreases, the discriminator gets more confused about its predictions.

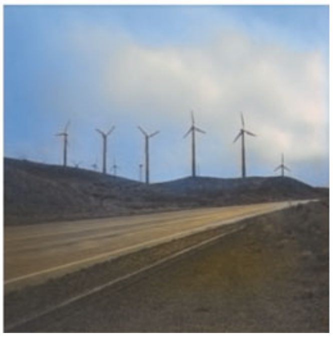
(a) Gray scale Image



(b) Original Image



(c) CNN Output



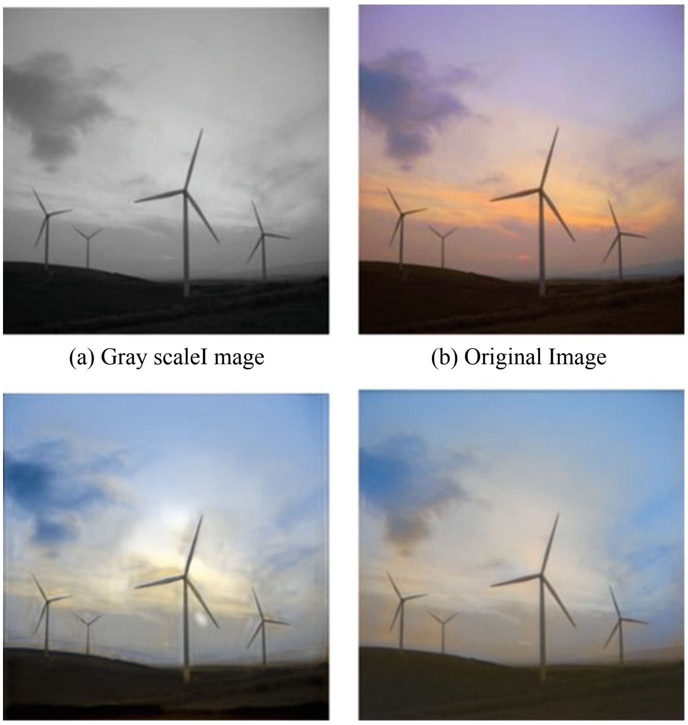
(d) GAN Output

Fig. 7 Color comparison between GAN and CNN

3 Results

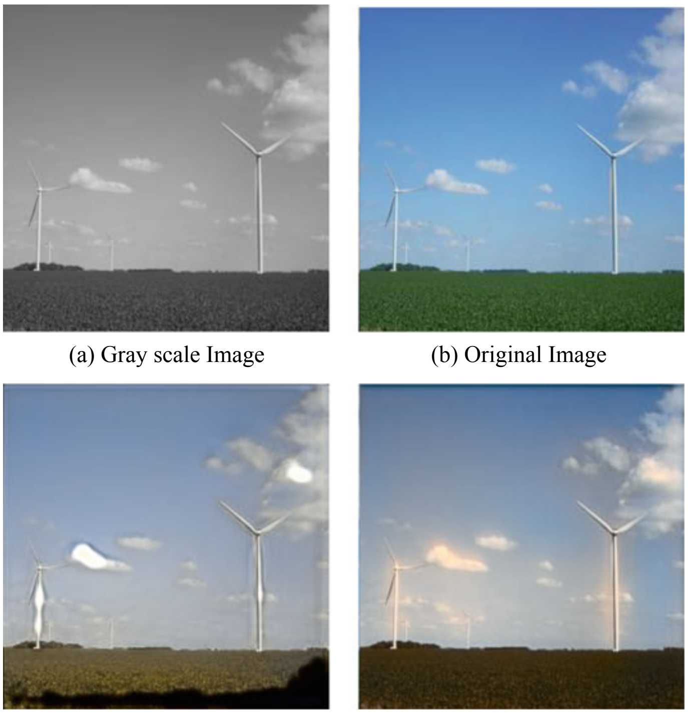
To compare results between a normal CNN and a GAN, only the generator network is trained with Least Absolute Deviations (l^1) cost function. After training with around 4000 images of wind farm from place 365 [16] dataset the characteristics of GAN is clearly observed and could be differentiated from the results of normal CNN. For instance, from Fig. 7d it can be observed that GAN network colored the sky with more of sky blue color but the normal CNN colored with a shade of blue which appears to be an average from dataset. The same can be observed with the color of grass in Fig. 9d.

The CNN network also didn't learn to pass on the localization information to the high level layers. From Fig. 8 it can be observed that some part of the image is lost and covered with a uniform color trying to reduce the loss. This effect is not seen in the GAN as the discriminator is trained to discriminate these type of images as fake.


(a) Gray scaleI mage (b) Original Image

(c) CNN Output (d) GAN Output

Fig. 8 Localization problem with CNN network


(a) Gray scale Image (b) Original Image

(c) CNN Output (d) GAN Output

Fig. 9 Localization problem with CNN network

4 Conclusion

Auto colorization of gray scale images has seen rapid research in recent years with new papers being published each month; although the limited realism and accuracy of these implementations leaves further scope of more novel and complex iterations of improving results. Our implementation offer a novel addition to the techniques that can be used for colorization by incorporating skip connections and modified cost functions, features at different scales and GANs which have significantly overcome the problem of picking average colors and color localization in general.

GANs are capable of producing high quality images at higher resolution [13]. When it comes to problems like images colorization, where there a conditional input, localization information becomes crucial. As seen in the Figs. 8c and 9c, the objects are not accurately localized because of which the colors are incorrectly spread across different objects. Building a RNN network on top of the CNN would be one of the solutions to address this problem.

The loss function for generator network includes L1-norm as one of the components which makes the network to produce average colors to a small extent. Removing this component made the generator network more sensible and as the discriminator became more confident of its predictions the generator network couldn't receive any gradients. To rectify this, l1 weight can be decayed over iterations which helps generator network to initially train properly and regain its GAN characteristics as the l1 weight decreases.

Coloring a gray scale image with desired qualities is also more important. Global features of the image such as whether it is cloudy, landscape, portrait will affect the colorization to a large extent. To achieve this, more than one condition can be fed to generator network. Additional conditions such as color histograms, hue, saturation, weather and camera mode will help to achieve desired colors for a gray scale image. Video colorization would be the next appendage to image colorization. Instead of colorizing a video frame by frame using a CNN model, LSTM networks can be used to store the color histograms and semantics of a buffer of frames and use them to color the adjacent or similar frames.

References

1. Yu F, Koltun V (2016) Multi-scale context aggregation by dilated convolutions. In: Proceedings of international conference on learning representations
2. Lisin DA, Mattar MA, Blaschko MB, Benfield MC, Learned-Mille EG (2005) Combining local and global image features for object class recognition. In: IEEE computer society conference on computer vision and pattern recognition (CVPR), p 47. https://doi.org/10.1109/CVPR.2005.433
3. Ng JY-H, Yang F, Davis LS (2015) Exploiting local features from deep networks for image retrieval. In: IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp 53–61
4. Iizuka S, Simo-Serra E, Ishikawa H (2016) Let there be Color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. ACM Trans Graph (Proc SIGGRAPH 2016) 35(4)
5. Zhang R, Zhu J-Y, Isola P, Geng X, Lin AS, Yu T, Efros AA (2017) Real-time user-guided image colorization with learned deep priors. In: SIGGRAPH
6. Charpiat G, Hofmann M, Schlkopf B (2008) Automatic image colorization via multimodal predictions. In: Forsyth D, Torr P, Zisserman A (eds) ECCV 2008, Part III. LNCS, vol 5304. Springer, Heidelberg, p 126139
7. Chia AYS, Zhuo S, Gupta RK, Tai YW, Cho SY, Tan P, Lin S (2011) Semantic colorization with internet images. ACM Trans Graph (TOG) 30:6. ACM
8. Gupta RK, Chia AYS, Rajan D, Ng ES, Zhiyong H (2012) Image colorization using similar images. In: Proceedings of the 20th ACM international conference on multimedia. ACM
9. Irony R, Cohen-Or D, Lischinski D (2005) Colorization by example. In: Eurographics symposium on rendering
10. Levin A, Lischinski D, Weiss Y (2004) Colorization using optimization. In: SIG-GRAPH
11. Sapiro G (2005) Inpainting the colors. In: IEEE international conference on image processing (ICIP). Genova
12. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680
13. Wang X, Gupta A (2016) Generative image modeling using style and structure adversarial networks. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer vision ECCV 2016. Lecture notes in computer science, vol 9908. Springer, Cham

14. Mirza M, Osindero S (2014) Conditional generative adversarial nets. arXiv:1411.1784
15. Hariharan B, Arbelaez P, Girshick R (2015) Hypercolumns for object segmentation and fine-grained localization. In: Proceedings of conference on computer vision and pattern recognition (CVPR)
16. Goodfellow I (2016) NIPS 2016 tutorial: generative adversarial networks. arXiv:1701.00160
17. Dahl R (2016) Automatic colorization. http://tinyclouds.org/colorize. Accessed 30 Aug 20108
18. Nazeri K, Ng E, Ebrahimi M (2018) Image colorization with generative adversarial networks. In: Perales F, Kittler J (eds) Articulated motion and deformable objects. AMDO 2018. Lecture notes in computer science, vol 10945. Springer, Cham. arXiv:1803.05400
19. Karras T, Aila T, Laine S, Lehtinen J (2018) Progressive growing of GANs for improved quality, stability, and variation. In: Proceeedings of international conference on learning representations (ICLR). Vancouver
20. Zhou B, Lapedriza A, Khosla A, Oliva A, Torralba A (2017) Places: a 10 million image database for scene recognition. IEEE Trans Pattern Anal Mach Intell (ICLR) 40:1452–1464. https://doi.org/10.1109/TPAMI.2017.2723009