# Image Colorization using Generative Adversarial Networks

Sahil Raj

IIT Guwahati, India
r.sahil@iitg.ac.in

**Abstract.** Automatic colorization has recently gained interest for old and damaged pictures. However, it is a tough job because of varying possibilities of color introduction. Unlike most current methods that focus on particularities such as themes or fine-grained inputs like semantic maps, our method departs to a more general approach. We propose the use of the DCGAN to colorize as a conditional Deep Convolutional GAN, as trained on the CelebA dataset. The comparison is also drawn between the performance of the proposed generative model and standard deep learning models.

## 1 Introduction

Colorizing grayscale images automatically has been an old interest in machine learning, with uses such as photo restoration of old photographs and generating colored versions for animation. Here, we examine image colorization with Generative Adversarial Networks (GANs), as presented by Goodfellow et al. Unlike previous methods that depended on hand-designed rules and user intervention, GANs learn how to introduce color from data.

Earlier colorization methods, like that of Welsh et al. (2002), employed texture matching between grayscale images and reference images. Subsequently, Levin et al. (2004) suggested a method with consideration given to the relationships between neighboring pixels. Nevertheless, the two methods required considerable hand labor.

Recent techniques involving CNNs and GANs performed better by learning the image mapping and the loss function. Taking inspiration from this, our work utilizes a conditional GAN in the form of a Deep Convolutional GAN (DCGAN) trained on the CelebA dataset. Our aim is to generalize earlier methods to perform well on high-resolution images, and also introduce training strategies that enhance speed and stability.

## 2 Dataset

For this project, we used the CelebA dataset, which contains approximately 200,000 high-quality color images of celebrity faces. From this dataset, we selected a subset of 10,000 images for training and evaluation. We split this subset into 80% for training and 20% for validation.

Each image was converted from the RGB color space to the Lab color space. In this format, the L channel (lightness) was used as the input (X) to the model, while the original RGB image served as the ground truth output (Y) for training. This setup helps the model learn to predict the missing color information from the grayscale input.

## 3 Generative Adversarial Network

Generative Adversarial Networks (GANs), introduced by Goodfellow et al. in 2014, consist of two components: a generator and a discriminator. The generator learns to produce realistic images, while the discriminator tries to distinguish between real and generated images. Both models are trained together in a competitive setup until the generator becomes good enough to fool the discriminator.

Since our task—image colorization—is a type of image translation, we use Convolutional Neural Networks (CNNs) for both the generator and discriminator. The generator takes a grayscale image (L channel from Lab space) as input and predicts the corresponding color image. The discriminator evaluates whether the generated image is close to real color images.

To address this, we use an alternate loss function for the generator, as suggested by Goodfellow in a later tutorial. Instead of minimizing the success of the discriminator, the generator maximizes the chance of fooling it, leading to better gradients and faster convergence:

$$\min_{\theta_G} -E_z \left[\log \left(D(G(z))\right)\right]$$

To further improve training stability and ensure realistic colorization, we add an L1 regularization term that encourages the generated image to stay close to the ground truth color image. The final loss function becomes:

$$\min_{\theta_G} -E_z \left[\log \left(D(G(z))\right)\right] + \lambda \|G(z) - y\|_1$$

Here,  is a regularization parameter, and y is the true color image. This helps preserve the structure of the image and avoids assigning arbitrary colors just to trick the discriminator.

### 3.1 Conditional GAN

In traditional GANs, the generator typically takes random noise z as input. However, for the image colorization task, our input is a grayscale image, not random noise. To handle this, we use a modified version called Conditional GAN (cGAN), where the grayscale image serves as the condition.

Instead of generating images from noise, the generator now learns a mapping from the grayscale input (L channel) to the corresponding color image (ab or RGB channels). Mathematically, this is represented as G(0z|x), where x is the grayscale input and 0z represents zero noise.

Similarly, the discriminator is also conditioned on the grayscale image. It receives a pair—either the real color image with the grayscale input or the generated color image with the same grayscale input—and tries to classify which one is real. The final loss functions become:

Generator Loss (with L1 regularization):

$$\min_{\theta_G} -E_z \left[\log \left(D(G(0z|x))\right)\right] + \lambda \|G(0z|x) - y\|_1$$

Discriminator Loss:

$$\max_{\theta_D} E_y \left[\log D(y|x)\right] + E_z \left[\log \left(1 - D(G(0z|x)|x)\right)\right]$$

## 4 Method

The image colorization task is rendered as an image-to-image translation problem, which aims to transform a grayscale version of an input image into a colored one. This view can also be expressed as a pixel-wise regression of color attributes, where the structural similarity between the input image (in grayscale) and the output one (in color) is of utmost importance. In other words, it concerns the task of predicting color values for each pixel while retaining the spatial dimensions.

We begin with a fully-convolutional neural-network baseline that predicts colors using regression loss. This baseline is subsequently extended forward by GANs to render results more plausible and aesthetically pleasing.

To separate brightness from color effectively, we employ the L*a*b* color space. This means that the:

L channel is for lightness (grayscale), and the a* and b* channels are for color components.

The use of L*a*b* color space effectively reduces noise and unpredictable changes in color or brightness, which RGB-based models generally tend to exhibit. The model takes the L channel as input and learns to predict a* and b* channels, which are then combined to generate the final color image.

### 4.1 Baseline Network

For our baseline, we adopt the Fully Convolutional Network (FCN) approach, replacing fully connected layers with convolutional operations and utilizing upsampling instead of pooling. This design is inspired by encoder-decoder architectures, where the input is progressively compressed through a series of contractive encoding layers and reconstructed via expansive decoding layers. This allows end-to-end training while keeping memory usage manageable, thanks to the compact feature representation in the middle layers.

However, a pure encoder-decoder suffers from an information bottleneck, limiting the transfer of low-level features. To overcome this, we integrate skip connections, passing features from the encoder directly to their corresponding

decoder layers. This helps retain fine-grained spatial information such as edges. This enhanced architecture is known as U-Net, where each skip connection links layer i of the encoder to layer n–i of the decoder.

The model architecture is symmetric:

- **Encoding path**: Each unit contains a 4×4 convolution (stride 2) for downsampling, followed by batch normalization [11] and a Leaky ReLU [12] activation (slope = 0.2). The number of channels doubles at each step.
- **Decoding path**: Each unit consists of a 4×4 transposed convolution (stride 2) for upsampling, concatenation with the corresponding encoder feature map, followed by batch normalization and a ReLU activation.
- The final layer is a 1×1 convolution followed by a tanh activation, producing 3 output channels representing the L*a*b* color space.

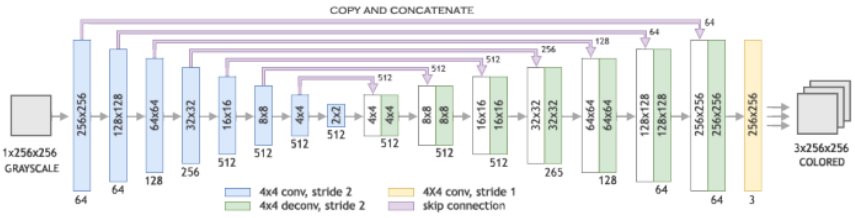


**Fig. 1.** U-Net architecture (256 × 256 input)

We train the model to minimize the Euclidean (L2) loss between predicted and ground truth values, averaged across all pixels and channels. The objective function is defined as:

$$J(x;\theta) = \frac{1}{3n}\sum_{\ell=1}^{3}\sum_{p=1}^{n}\left\|h(x;\theta)_{(p,\ell)} - y_{(p,\ell)}\right\|_2^2 \tag{1}$$

where x is our grayscale input image, y is the corresponding color image, p and l are indices of pixels and color channels respectively, n is the total number of pixels, and h is a function mapping from grayscale to color images.

## 5 Results

During training, we observed a consistent improvement in generator performance. From epoch 1 to 20, the L1 loss of the generator (loss-G-L1) decreased significantly, indicating improved pixel-wise accuracy in the colorization task. Additionally, a slight increase in discriminator loss suggests the generator became better at fooling the discriminator over time.

| Loss Metric | Epoch 5 | Epoch 10 | Epoch 15 | Epoch 20 |
|---|---|---|---|---|
| loss_D_fake | 0.55271 | 0.58009 | 0.60650 | 0.62224 |
| loss_D_real | 0.60198 | 0.63222 | 0.64532 | 0.64407 |
| loss_D | 0.57735 | 0.60615 | 0.62591 | 0.63316 |
| loss_G_GAN | 1.14168 | 1.04498 | 0.96457 | 0.91493 |
| loss_G_L1 | 7.30180 | 6.77980 | 5.78608 | 4.89597 |
| loss_G | 8.44348 | 7.82478 | 6.75065 | 5.81090 |

**Table 1.** Generator and Discriminator Losses at Epochs 5, 10, 15, and 20

## 6 Conclusion

In this project we are able to automatic image colorization techniques through cGANs and baseline convolution networks. We could separate brightness and color information by utilizing the Lab* color space, which would lead to better colorization results than that of RGB space images.

Our new results show that the cGAN model is a better performer as it produced a more realistic and accurate colored image. It could be said that the U-Net architecture in conjunction with skip connections helps preserve detail during the colorization. The improvements resulted from adding the regularization term and conditional inputs.

In the future, we will also try to find efficient methods to increase training efficiency, test with higher resolution images, and further optimize the model to yield better results. It holds great promise for future application use cases for color restoration, animation, and content creation.

## References

1. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative adversarial nets*. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014.
2. Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. *Places: An image database for deep scene understanding*. 2016.
3. Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. *Transferring color to greyscale images*. In ACM TOG, volume 21, pages 277–280, 2002.
4. Anat Levin, Dani Lischinski, and Yair Weiss. *Colorization using optimization*. In ACM Transactions on Graphics (TOG), volume 23, pages 689–694. ACM, 2004.
5. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. *Image-to-image translation with conditional adversarial networks*. 2016.
6. Ian Goodfellow. *NIPS 2016 tutorial: Generative adversarial networks*. 2016.
7. Mehdi Mirza and Simon Osindero. *Conditional generative adversarial nets*. 2014.
8. Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully convolutional networks for semantic segmentation*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3431–3440, 2015.
9. Geoffrey E Hinton and Ruslan R Salakhutdinov. *Reducing the dimensionality of data with neural networks*. Science, 313(5786):504–507, 2006.

10. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-net: Convolutional networks for biomedical image segmentation.* In International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 2015.
11. Sergey Ioffe and Christian Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift.* In International Conference on Machine Learning, 2015.
12. Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. *Rectifier nonlinearities improve neural network acoustic models.* In Proc. ICML, volume 30, 2013.
13. Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised representation learning with deep convolutional generative adversarial networks.* 2015.
14. Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: A simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research, 15(1):1929–1958, 2014.
15. Diederik Kingma and Jimmy Ba. *Adam: A method for stochastic optimization.* 2014.