# Image Colorization with Generative Adversarial Networks

**K. Nazeri[1], and E. Ng[1]**
[1]**ImagingLab**
**Faculty of Science**
**University of Ontario Institute of Technology**
**Oshawa**
**Ontario**
**Canada**

ABSTRACT

Over the last decade, the process of automatic colorization had been studied thoroughly due to its vast application such as colorization of grayscale images and restoration of aged and/or degraded images. This problem is highly ill-posed due to the extremely large degrees of freedom during the assignment of color information. Many of the recent developments in automatic colorization involved images that contained a common theme throughout training, and/or required highly processed data such as semantic maps as input data. In our approach, we attempted to fully generalize this procedure using a conditional Deep Convolutional Generative Adversarial Network (DCGAN). The network is trained over datasets that are publicly available such as CIFAR-10 and Places365. The results between the generative model and tradition deep neural networks are compared.

**Keywords:** Image Colorization, Deep Convolutional Generative Adversarial Networks

## 1 INTRODUCTION

The automatic colorization of grayscale images has been an active area of research in machine learning for an extensive period of time. This is due to the large variety of applications such as the colorization of Manga (Japanese comic), color restoration, and colorization for animations. In this project, we will explore the method of colorization using generative adversarial networks (GANs) proposed by Goodfellow et al. [1]. The network is trained on the datasets CIFAR-10 and Places365 [2] and its results will be compared with those obtained using existing convolutional neural networks (CNN).

Colorization of grayscales began back in the early 2000s. In 2002, Welsh et al. [3] proposed an algorithm that colorized images through texture synthesis. Colorization was done by matching luminance and texture information between an existing color image and the grayscale image to be colorized. However, this proposed algorithm was defined as a forward problem, thus all solutions were deterministic.

Levin et al. [4] proposed an alternative formulation to the colorization problem in 2004. This formulation followed an inverse approach, where the cost function was designed by penalizing difference between each pixel and a weighted average of its neighboring pixels. This is a technique frequently used in modern medical image registration. However, both of these proposed methods still required significant user intervention which made the solutions less than ideal.

In [5], a colorization method was proposed by comparing colorization differences between those generated by convolutional neural networks (CNN) and GAN. The models in the study were trained and validated using still frames from anime (Japanese cartoons). These images all have objects that are highly distinguishable from its foreground. We aim to extend their approach by generalizing the colorization procedure to all kinds of images rather than a specific category of images.

In our method, we use L*a*b color space instead of RGB which contains a dedicated channel depict the brightness of the image and the color information are fully encoded in the remaining two channels. As a result, this prevents any sudden variations in both color and brightness through small perturbations in intensity values that are experienced through RGB.

## 2 GENERATIVE ADVERSARIAL NETWORK

In 2014, Goodfellow et al. [1] proposed a new type of generative models: generative adversarial networks (GANs). A GAN is composed of two smaller networks called the generator and discriminator. As the name suggests, the generator's task is to produce results that are indistinguishable from real data. The discriminator's task is to classify whether a sample originated from the generator's model distribution or the original data distribution. Both of these subnetworks are trained simultaneously until the generator is able to consistently produce results that the discriminator cannot classify.

The architectures of the generator and discriminator both follow a multilayer perceptron model. Since colorization is a class of image translation problems, the generator and discriminator are both convolutional neural networks (CNN). The generator can be represented as the mapping $G(z;\theta_g)$ , where $z$ is a noise variable (uniformly distributed) that act as the input of the generator. Similarly, the discriminator can be represented as the mapping $D(x;\theta_d)$ to produce a scalar between 0 and 1. The output of the discriminator can be viewed as the probability of the input originating from the training data. These constructions of $G$ and $D$ enables us to determine the optimization problem for training the generator and discriminator. $G$ is trained to minimize the probability that the discriminator makes a correct prediction in generated data, while $D$ is trained to maximize the probability of assigning the correct label. Mathematically, this can be expressed as

$$\min_{\theta^G} J^{(G)}(\theta_D,\theta_G) = \min_{\theta^G} \mathbb{E}_z\left[\log(1-D(G(z)))\right], \tag{1}$$

$$\max_{\theta^D} J^{(D)}(\theta_D,\theta_G) = \max_{\theta^D}\left(\mathbb{E}_x\left[\log(D(x))\right] + \mathbb{E}_z\left[\log(1-D(G(z)))\right]\right). \tag{2}$$

The above two equations provide the cost functions required to train a GAN. In literature, these two cost functions are often presented as a single minimax game problem with the value function $V(G,D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_x\left[\log D(x)\right] + \mathbb{E}_z\left[\log(1-D(G(z)))\right]. \tag{3}$$

In our model, we have decided to use an alternate cost function for the generator. In equation 1, the cost function is defined by minimizing the probability of the discriminator being correct. However, this approach presents two issues: 1) If the discriminator performs well during training stages, the generator will have a near-zero gradient during back-propagation. This will tremendously slow down convergence rate because the generator will continue to produce similar results during training. 2) The original cost function is a strictly decreasing function that is unbounded below. This will cause the cost function to diverge to $-\infty$ during the minimization procedure.

To address the above issues, we have redefined the generator's cost function by maximizing the probability of the discriminator being mistaken, as opposed to minimizing the probability of the discriminator being correct. The new cost function was suggested by Goodfellow at NIPS 2016 Tutorial [6] as a heuristic, non-saturating game, and is presented as

$$\max_{\theta^G} J^{(G)^*}(\theta_D,\theta_G) = \max_{\theta^G} \mathbb{E}_z\left[\log(D(G(z)))\right], \tag{4}$$

which can also be written as the minimization problem

(5)

The comparison between the cost functions in equations 1 and 5 can be visualized in figure 1 by the blue and red curves respectively.

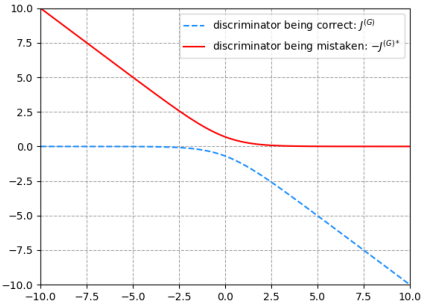


*Figure 1: Comparison of cost functions $J^{(G)}$ discriminator being correct (dashed blue) and $-J^{(G)*}$ discriminator being mistaken (red).*

In addition, the cost function was further modified by adding an $L1$-norm based regularizer [7]. This produces an effect where the generator is forced to produce results that are similar to the ground truth on the pixel level. This will theoretically preserve the structure of the original images and prevent the generator from assigning arbitrary colors to pixels just to fool the discriminator. The regularized cost function takes the form

$$\begin{aligned} &\min_{\theta^G} J^{(G)*}(\theta_D, \theta_G) = \\ &\min_{\theta^G} -\mathbb{E}_z \left[\log(D(G(z)))\right] + \lambda \|G(z) - y\|_1 \end{aligned} \tag{6}$$

where $\lambda$ is a regularization parameter and $y$ is the ground truth color labels.

### 2.1 CONDITIONAL GAN

In a traditional GAN, the input of the generator is randomly generated noise data $z$. However, this approach is not applicable to the automatic colorization problem due to the nature of its inputs. The generator must be modified to accept grayscale images as inputs rather than noise. This problem was addressed by using a variant of GAN called conditional generative adversarial networks [8]. Since no noise is introduced, the input of the generator is treated as zero noise with the grayscale input as a prior, or mathematically speaking, $G(\mathbf{0}_z|x)$. In addition, the input of the discriminator was also be modified to accommodate for the conditional network. By introducing these modifications, our final cost functions are as follows:

$$\begin{aligned} &\min_{\theta^G} J^{G}(\theta_D, \theta_G) = \\ &\min_{\theta^G} -\mathbb{E}_z \left[\log(D(G(\mathbf{0}_z|x)))\right] + \lambda \|G(\mathbf{0}_z|x) - y\|_1 \end{aligned} \tag{7}$$

$$\begin{aligned} &\max_{\theta^D} J^{D}(\theta_D, \theta_G) = \\ &\max_{\theta^D} \left(\mathbb{E}_y \left[\log(D(y|x))\right] + \mathbb{E}_z \left[\log(1 - D(G(\mathbf{0}_z|x)|x))\right]\right) \end{aligned} \tag{8}$$

The discriminator gets colored images from both generator and original data along with the grayscale input as the condition and tries to tell which pair contains the true colored image. Figure 2 depicts this process.

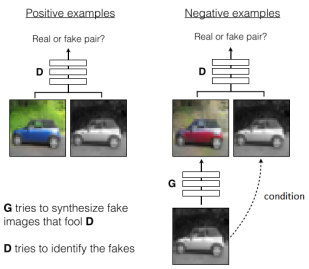


*Figure 2: Schematic representation of conditional GANs.*

## 3 NETWORK ARCHITECTURE

Image colorization is an image-to-image translation problem that maps a high dimensional input to a high dimensional output. It can be seen as pixelwise regression problem where structure in the input is highly aligned with structure in the output. That means the network needs not only to generate an output with the same spatial dimension as the input, but also to provide color information to each pixel in the grayscale input image. We provide an entirely convolutional model architecture using a regression loss as our baseline and then extend the idea to adversarial nets.

### 3.1 BASELINE NETWORK

For our baseline model, we follow the "fully convolutional network"[9] model where the fully connected layers are replaced by convolutional layers which have upsampling instead of pooling operators. This idea is based on encoder-decoder networks [10] where input is progressively downsampled using series of contractive encoding layers, and then the process is reversed using a series of expansive decoding layers to reconstruct the input. Using this method we can train the model end-to-end without consuming large amounts of memory. Note that the subsequent downsampling leads to a much more compact feature learning in the middle layers. This strategy forms a crucial attribute to the network, otherwise the resolution would be limited by GPU memory.

Our baseline model needs to find a direct mapping from the grayscale image space to color image space. However, there is an information bottleneck that prevents flow of the low level information in the network in the encoder-decoder architecture. To fix this problem, features from the contracting path are concatenated with the upsampled output in the expansive path within the network. This also makes the input and output share the locations of prominent edges in grayscale and colored images. This architecture is called U-Net [11], where skip connections are added between layer $i$ and layer $n$-$i$. Figure 3 shows this architecture.

The architecture of the model is symmetric, with 5 encoding units and 5 decoding units. The contracting path has the typical architecture of a convolutional networks: two $3 \times 3$ convolution layers, each followed by batch normalization [12], ReLU activation function and $2 \times 2$ max pooling operation with stride 2 for downsampling. The number of channels are doubled after each downsampling step. Each unit in the expansive path consists of an upsampling layer, followed by a $2 \times 2$ convolution layer that halves the number of channels, concatenation with the activation map of the mirroring layer in the contracting path, and two $3 \times 3$ convolution layers each followed by batch normalization and ReLU activation function. The last layer of the network is a $1 \times 1$ convolution which is equivalent to cross-channel parametric pooling layer. The number of channels in the output layer is 3 with L*a*b* color space.

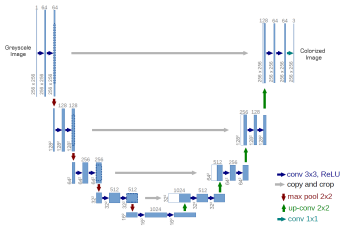


*Figure 3: U-Net architecture. The gray arrows represent the skip connections from encoder to the mirroring decoder layer.* [11]

We train our baseline model to minimize the Euclidean distance between predicted and ground truth averaged over all pixels:

$$J(x;\theta) = \frac{1}{[illegible]} \tag{9}$$

where $x$ is our grayscale input image, $y$ is the corresponding color image, $p$ denotes each pixel, $n$ denotes the total number of pixels and $h$ is a function mapping from grayscale to color images.

### 3.2 CONVOLUTIONAL GAN

For our generator and discriminator models, we followed Deep Convolutional GANs (DCGAN) [13] guidelines and employ convolutional networks in both generator and discriminator architectures. The architecture was also modified as a conditional GAN instead of a traditional DCGAN.

The architecture of generator $G$ is the same as our baseline: 5 convolution units and 5 convolution-transpose units, with skip connections connecting mirroring layers. Each layer in the contractive path was followed by batch normalization and leaky rectified linear unit (leaky ReLU) [14] with slope 0.2. In the expansive path, we used batch normalization and ReLU after each convolution layer. We did not include batch normalization or any activation function for the last layer of the network.

For discriminator $D$, we use a conventional convolutional neural network classifier architecture: a series of $3 \times 3$ convolutional layers followed by max-pooling layer with the number of channels being doubled after each downsampling. All convolution layers are followed by batch normalization, leaky ReLU activation with slope 0.2 and dropout [15] with a dropout rate of 20% to prevent the discriminator from overfitting. After the last layer,

a convolution is applied to map to a 1 dimensional output, followed by a sigmoid function to return a probability value of the input being real or fake. The input of the discriminator is a colored image either from generator or true labels, concatenated with the grayscale image. Figure 4 shows this architecture, where each *Encode* unit denotes Convolution-BatchNorm-LeakyReLU-Dropout-MaxOut.

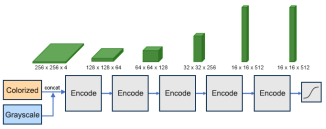


*Figure 4: Discriminator architecture: Encode units denotes Convolution-BatchNorm-LeakyReLU-Dropout-MaxOut*

### 3.3 TRAINING STRATEGIES

For training our network, we used minibatch stochastic gradient descent with Adam [16] optimization and weight initialization proposed by [17]. We used initial learning rate of $5 \times 10^{-4}$ for both generator and discriminator and manually decay the learning rate by a factor of 10 whenever the loss function started to plateau. For the extra hyper-parameter $\lambda$ we followed the protocol from [7] and chose $\lambda = 100$, which forces the generator to generate images similar to ground truth.

GANs have been known to be very difficult to train as it requires finding a Nash equilibrium of a non-convex game with continuous, high dimensional parameters [18]. We followed a set of constraints and techniques proposed by [7, 13, 18, 19] to encourage convergence of our convolutional GAN and make it stable to train:

- **Alternative Cost Function**
  This heuristic alternative cost function [6] was selected due to its non-saturating nature; the motivation for this cost function is to ensure that each player has a strong gradient when that player is “losing" the game.
- **Batch Normalization**
  One of the main difficulties when training GANs is for the generator to collapse to a parameter setting where it always emits the same output [18]. This phenomenon is called *mode-collapse*, also known as **the Helvetica scenario** [6]. When mode-collapse has occurred, the discriminator learns that this single output comes from the generator. Batch normalization [12] is proven to be essential to train both networks preventing the generator from collapsing all samples to a single point. [13]
- **One Sided Label Smoothing**
  Deep neural networks normally tend to produce extremely confident outputs when used in classification. It is shown that replacing the 0 and 1 targets for a classifier with smoothed values, like .1 and .9 is an excellent regularizer for convolutional networks [20]. Salimans et al [18] demonstrated that *one-sided label smoothing* will encourage the discriminator to estimate soft probabilities and reduce the vulnerability of GANs to adversarial examples. In this technique we smooth *only* the positive labels to 0.9, leaving negative labels set to 0.
- **Reduced Momentum**
  We use Adam optimizer [16] for training both networks. Recent research have shown that using a large momentum term $\beta_1$ (0.9 as suggested), could result in oscillation and instability in training. We followed [13] suggestion to reduce the momentum term to 0.5.
- **LeakyReLU Activation Function**
  Radford et al. [13] showed that using leaky ReLU [7] activation functions in the discriminator resulted in better performance over using regular ReLUs. We also found that using leaky ReLU in the encoder part of the generator as suggested by [7] works slightly better.

## 4 EXPERIMENTAL RESULTS

To measure accuracy, we have chosen to employ mean absolute error (MAE) and accuracy. MAE is computed by taking the mean of the absolute error of the generated and source images on a pixel level for each color channel. Accuracy is measured by the ratio between the number of pixels that match perfectly with the source and the total number of pixels. Note that we have omitted training Places365 on U-Net due to time constraints. The training results for each model are summarized below in Table 1.

| Dataset | Network | Batch Size | EPOCHs | MAE | Accuracy |
|---|---|---|---|---|---|
| CIFAR-10 | U-Net | 128 | 500 | 5.9 | 24.8% |
| CIFAR-10 | GAN | 128 | 500 | 1.2 | 43.7% |
| Places365 | GAN | 8 | 50 | 3.2 | 33.4% |

*Table 1: Training Results of Baseline Model and GAN.*

Some of the preliminary results using the CIFAR-10 ($32 \times 32$) dataset are shown in Appendix A. The images from GAN had a clear visual improvement than those generated by the baseline CNN. The images generated by GAN contained colors that were more vibrant whereas the results from CNN suffered from a light hue. In some cases, the GAN was able to nearly replicate the ground truth. However, one drawback was that the GAN tend to colorize objects in colors that are most frequently seen. For example, many car images were colored red. This is most likely due to the significantly larger number of image with red cars than images with cars of another color.
The preliminary results using Places365 ($256 \times 256$) are shown in Appendix B. We noticed that there were some instances of mis-colorization: regions of images that have high fluctuations are frequently colored green. This is likely caused by the large number of grassland images in the training set, thus the model leans towards green whenever it detects a region with high fluctuations in pixel intensity values. We also noticed that some colorized images experienced a "sepia effect" seen with CIFAR-10 under U-Net. This hue is evident especially with images with clear sky, where the color of the sky experience a strange color gradient between blue and light yellow. We suspect that this was caused by insufficient training, and will correct itself over time.

## 5 CONCLUSION

In this study, we were able to automatically colorize grayscale image to an acceptable degree using GAN. With the CIFAR-10 dataset, the model was able to consistently produce better looking (qualitatively) images than U-Net. Many of the images generated by U-Net had a brown-ish hue in the results known as the "Sepia effect" across L*a*b* color space. This is due to the L2 regularization that was applied to the baseline CNN, which is known to cause a blurring effect.
We obtained mixed results when colorizing grayscale images using the Places365 dataset. Mis-colorization was a frequent occurrence with images containing high levels of textured details. This leads us to believe that the model have identified these regions as grass since many images in the training set contained leaves or grass in an open field. In addition, this network was not as well-trained as the CIFAR-10 counterpart due to its significant increase in resolution ($256 \times 256$ versus $32 \times 32$) and the size of the dataset (1.8 million versus $50,000$). We expect the results will improve if the network is trained further.

We would like to consider the replacement of spatial pooling functions with strided convolutions. This effectively allows the model to learn its own downsampling/upsampling rather than relying on a fixed downsampling/upsampling method. This idea was proposed in [21] and have shown to improve training performance. This is especially true with the Places365 dataset, as the reduced quality image was due to the model not being trained well due to our time constraint. We would also need to seek a better quantitative metric to measure performance. This is because all evaluations of image quality were qualitative in our tests. Thus, having a new or existing quantitative metric such as peak signal-to-noise ratio (PSNR) and root mean square error (RMSE) will enable a much more robust process of quantifying performance.

Source code is publicly available at: https://github.com/ImagingLab/Colorizing-with-GANs

## REFERENCES

[1] *I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, 2014, pp. 2672–2680.*

[2] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva, "Places: An image database for deep scene understanding," *arXiv preprint arXiv:1610.02055*, 2016.

[3] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 277–280.

[4] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM transactions on graphics (tog)*, vol. 23, no. 3. ACM, 2004, pp. 689–694.

[5] *Q. Fu, W.-T. Hsu, and M.-H. Yang. (2017) Colorization using convnet and gan. http://cs231n.stanford.edu/reports/2017/pdfs/302.pdf.*

[6] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[8] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[12] *S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in International Conference on Machine Learning, 2015, pp. 448–456.*

[13] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[14] *A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in Proc. ICML, vol. 30, no. 1, 2013.*

[15] *N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research, vol. 15, no. 1, pp. 1929–1958, 2014.*

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[18] *T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in Advances in Neural Information Processing Systems, 2016, pp. 2234–2242.*

[19] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *arXiv preprint arXiv:1710.07035*, 2017.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

## APPENDIX A CIFAR-10 RESULTS

**1-CIFAR-10 Results**

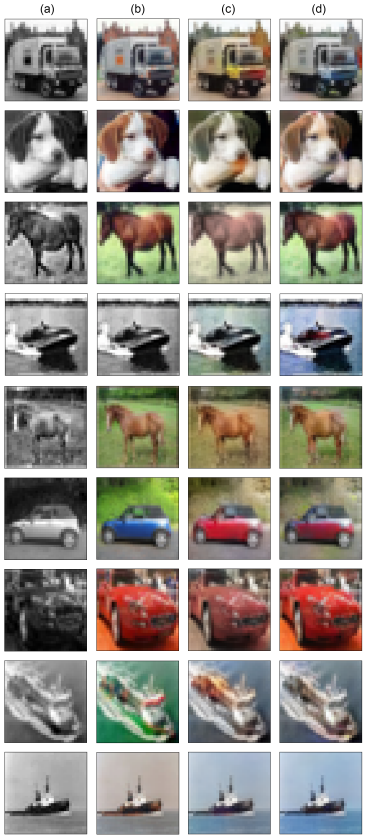


*Figure 5: Colorization results with CIFAR10. (a) Grayscale. (b) Original Image. (c) Colorized with U-Net. (d) Colorized with GAN.*

APPENDIX B PLACES365 RESULTS

**2-Places365 Results**

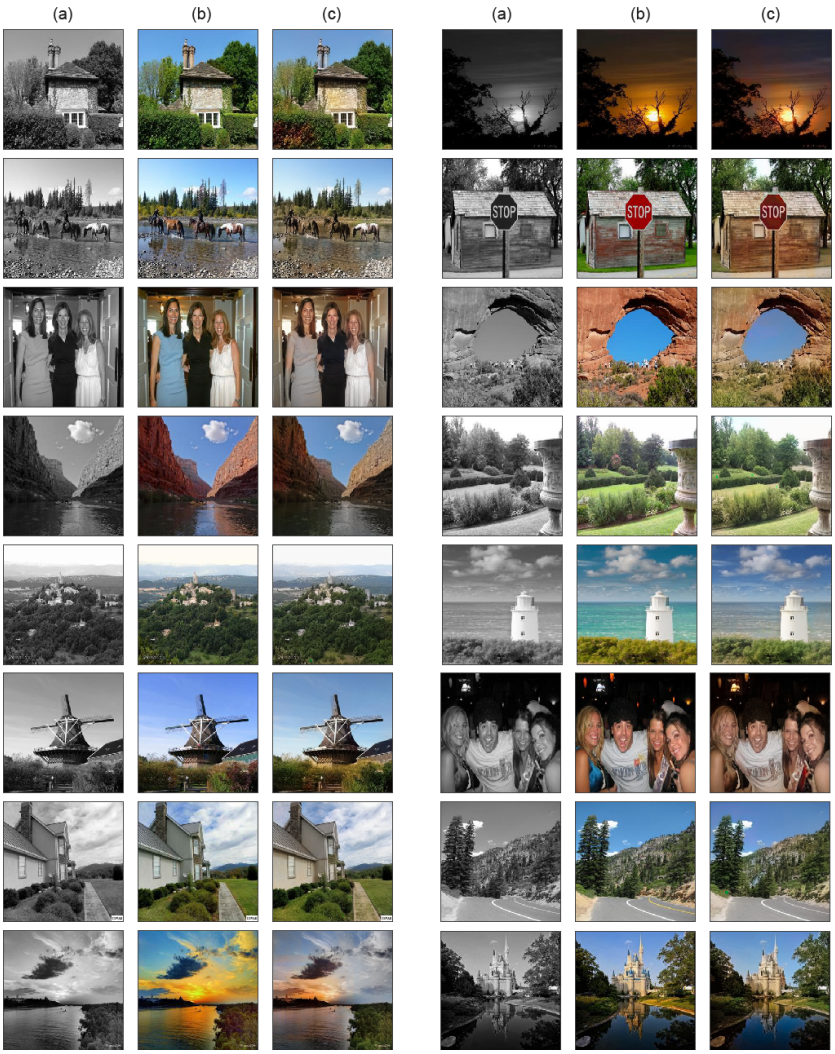


*Figure 6: Colorization results with Places365. (a) Grayscale. (b) Original Image. (c) Colorized with GAN.*