# Colorization Using ConvNet and GAN

Qiwen Fu
qiwenfu@stanford.edu
Stanford Univeristy

Wei-Ting Hsu
hsuwt@stanford.edu
Stanford University

Mu-Heng Yang
mhyang@stanford.edu
Stanford University

## Abstract

*Colorization is a popular image-to-image translation problem. Because most people nowadays still read grayscale manga, we decided to focus on manga colorization. We implemented two models for the task: ConvNet and conditional-GAN and found that GAN can generate better results both quantitatively and qualitatively. Since some manga only contains edges instead of grey-scale images, we also experimented with both inputs and test on various sources of animations and found that using grey-scale images can lead to clearer boundaries and brighter colors. Many examples of generated images and error analysis are discussed in the last part.*

## 1. Introduction

Automatic colorizations is an area of research that possesses great potentials in applications: from black & white photos reconstruction, augmentation of grey scale drawings, to re-colorization of images. Specifically, we will investigate a subcategory of colorizations: automatic colorizations in Manga (Japanese comics). Most Manga are drawn without colors until they are made into animations and aired on televisions. We think automatic colorization in Manga can be beneficial in providing readers with more pleasant and comprehensive reading experience, though it can also be generalized and expected to work on any hand-drawn images.

We envision the applications to be two-folded: 1) the model can be trained specific to a particular animation, and apply on not-yet animated manga to automate the colorization step of animations. This way the model can learn how particular characters/animals/objects should be colored. The model can also be 2) train on myriad of animations and learn how to color drawings in general. We focus on 1) in our project due to limited resource and time to collect wide variety of training data. However, we will touch on 2) by testing on different mangas to evaluate the colorization capability on different styles of drawings by training on a particular animation.

The ultimate goal is to color manga. However, it is difficult to obtain one-to-one correspondence of uncolored and colored manga images as training data. Therefore, we manufacture uncolored manga images based on colored images. Typical uncolored manga are consisted of black and white, with some use of screentone (dotted texture) to apply shades and grayscale effect. We manufacture two types of images: grayscale and edge-only from color images to simulate uncolored manga, which should be in between these two types of estimations. The two types would both be the input of our models and evaluated.

We propose two models for this task: ConvNet and GAN. Both models are design to take either grayscale or edge-only images and produce color (RGB) images. While there are previous work of similar types of networks on colorizations, there are not work tailored specific to manga colorization based on animation. Since colorings in animations are typically more fictitious and whimsical comparing to real-life photos, it would also be an interesting aspect to compare how our neural network would color a specific object as opposed to one trained on real-life photos.

## 2. Related Work

### 2.1. Colorization with hint

Hint-based colorization requires human supervision to complete the colorization. There are two popular hint-based colorization: scribble-based method and color transfer method. Scribble-based method proposed by Levin et al. [9] is very effective and popular. Fig. 1 is an example figure from the original paper. Given a user-drawn colored scribble, the model can colorize the area with that color-tone using convex optimization. The model can easily generate high quality image because the model doesnt really learn how to color a specific item. Its more like children coloring book with a parent telling which color to use. There are some extended works. Yatziv et al. [17] use chrominance blending for video colorization. Qu et al. [11] improve continuity of colorization on similar textures for manga.

Another famous hint-based colorization is the transfer colorization method proposed by Welsh et al. [16] and Ironi

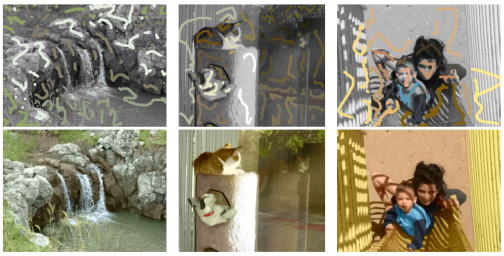


Figure 1. Example of scribble-based colorization method

et al. [5]. Besides grayscale image, the model needs another colored image for reference. The model can match the information between the grayscale image and referenced image. For example in Fig. 2, both images have faces, then the model will learn to color the grayscale image with skin and hair color in the referenced image.

Figure 2. Example of transfer colorization method

### 2.2. Fully Automated Colorization

Despite the quality of the results, the hint-based models still requires lots of human labor to generate hints. Thereby, fully automated colorization models that requires only gray-scale are proposed. Many works use feature like HoG [2], DAISY [15] [1], or color histogram [4] to generate colored images. However, with the advent of deep learning [8] and big data, convolutional neural nets [7] have shown great results in computer vision by learning hierarchical feature representation and gradually replaced the feature engineering part of the above mentioned features. We can now train an end-to-end model using only gray-scale image to generate colored image. With simple pixel-wise L2 loss over the ground-truth, the model can learn to generate results similar to ground-truth. However, L2 will impose a averaging effect over all possible color candidates. Therefore, the results are dimmer and sometimes causing patches with different colors in the same area.

### 2.3. Generative Adversarial Networks

Goodfellow et al. proposed GAN [3] to generate images from random noise. Using the adversarial learning from generator and discriminator, the minimax loss is very different from the L2 loss defined above. It will choose a color to fill an area rather than averaging. Many extended works of GAN have been proposed this year, including DCGAN [12], Conditional-GAN [10], iGAN [18], and Pix2Pix [6]. DCGAN [12] stacks deep convolutional neural nets as generator and discriminator to learn hierarchical visual representation. DCGAN is nowadays the standard architecture for image generation. Instead of generating images from random noise, conditional GAN [10] is given a condition to generate an output image. For example, gray-scale image is the condition for colorization. iGAN [18] is an extension to conditional GAN. Similar to scribble-based colorization mentioned above, given the colored scribbles of different patches, iGAN can generate the image according to the scribbles. Pix2Pix [6] is conditional-GAN with images as the condition. Besides learning the mapping from input image to output image, it can also learn a loss function to train this mapping. It is consider the state-of-the-art in image-image translation problem like colorization.

## 3. Methods

### 3.1. ConvNets

The ConvNets model uses convolution layers in an encoder-decoder fashion to generate colorized images from input grayscale or edge-only images and use pixel-wise L2 loss as the objective function.

#### 3.1.1 Architecture

The ConvNet is consisted of a few layers of encoder and a few layers of decoder to generate colorized images from input grayscale images or edge images. For a deep convolution neural network without dimension reduction, one serious problem is that the size of output tensor will remains the same (180x320) after every layer, and thereby lead to very large memory consumption. With strides larger than 1 in each conv2d layer, the tensor shape will shrink quickly. This is what the encoder is doing. We can then use conv2d_transpose to upsample the tensor back to the original shape to get the color images, which is the function of decoder. Using this model can lead to much more compact feature learning in the middle of the layers without consuming large memory.

The architecture of our ConvNet model is symmetrical: it consists 6 layers of encoding and 6 layers of decoding; with increasing number of filters when encoding and decreasing number of filters when decoding. Each encoding layer is consisted of a conv2d for downsizing, batch normalization, and leakly relu activations; each decoding layer is consisted of a conv2d_transpose for upsizing, batch normalization and relu activations. However,at each decoding layer, we concatenate the mirroring layer from encoder before moving on to the next one, creating ability for network to skip layers. This architecture is called U-Net [13]. With skipped layers, the model can learn weights to ignore deeper

layers. This can help model to retain components from original input more easier in deep CNN, which is particularly useful in colorization task where we wish to only change the RGB values of image. The architecture is depicted in Fig. 3.

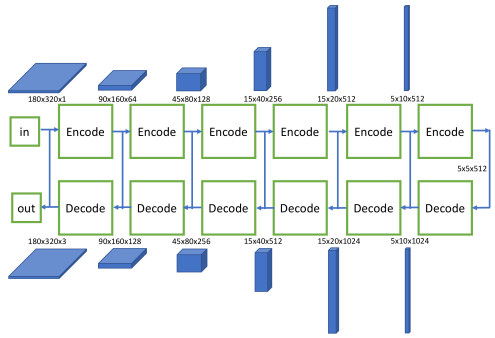



Figure 3. Encoder-Decoder ConvNets

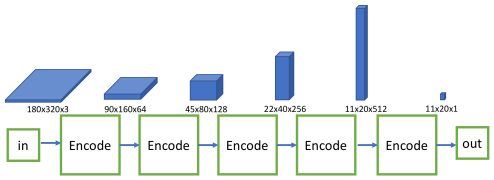



Figure 4. Discriminator

#### 3.1.2 Objective

The objective is to minimize the difference between model output and label. The most intuitive way is minimize the distance between two image pixel-wise. Let $F(x_i;\theta)$ denote the output of the $i^{th}$ training example from ConvNet model parameterized by $\theta$. We define our loss for the $i^{th}$ training example to be:

$$L_i = \frac{1}{2}\sum_{p=1}^{n} |F(x_i;\theta)^p - y_i^p|^2$$

where $p$ denotes each pixel and $n$ denotes the total number of pixel in an image, which is 180 x 320 in our case.

The overall objective of ConvNet can then be represented as the minimization of $L_{ConvNet}$:

$$L_{ConvNet} = \sum_{i}^{N} L_i$$

where N denotes the total number of training examples.

A behavior of this loss function is that, it will cause the model to choose safe color using averaging. If there are two plausible colors, the model will simply take the average as the results to minimize the loss. As a result, this might lead to gradient colors patches in similar objects.

### 3.2. Conditional GANs

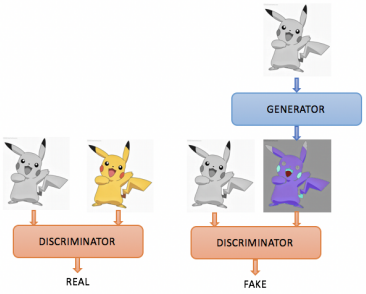



Figure 5. Conditional GANs

#### 3.2.1 Architecture

Generative Adversarial Nets (GAN) have two competing neural network models. The generator takes the input and generates fake image. The discriminator gets images from both the generator and the label, along with the grayscale or edge-only input, and try to tell which pair contains the real colored image. Fig. 5 depicts this process. During training, generator and discriminator are playing a continuous game. At each iteration, generator can produce more realistic photo, while the discriminator is getting better at distinguishing fake photo. Both models are trained together in a minimax fashion and the goal is to train a generator to be indistinguishable from real data.

Our GAN is conditioned on gray or edge-only image, which would be the input to our generator. The architecture of generator is the same as the one in ConvNets (at section 3.1.1). It is consisted of 6 convolution layers and 6 convolution-transpose layers, with skipping at mirroring layers, which would eventually outputs an image of the same size as input but with 3 channels, representing Red, Green, and Blue accordingly.

The input of the discriminator is the concatenation of grayscale or edge-only image with color images, either from generator or labels. The discriminator is consisted of 6 layers of encoder, just like the encoding portion of the generator: each encode layer is consisted of a convolution operation with stride greater than 1, a batch normalization, and a leaky relu activation. The last layer then goes through a sigmoid activation to return a number from 0 to 1 that can be interpreted as the probability of the input being real or fake. A depiction of the discriminator architecture can be seen at Fig. 4.

#### 3.2.2 Objective

With conditional GAN, both generator and discriminator are conditioning on the input $x$. Let the generator be parameterized by $\theta_g$ and discriminator be parameterized by $\theta_d$.The minimax objective function can be expressed as:

$$\min_{\theta_g}\max_{\theta_d}\Big[\mathbb{E}_{x,y\sim p_{data}}\log D_{\theta_d}(x,y)+ E_{x\sim p_{data}}\log(1-D_{\theta_d}(x,G_{\theta_g}(x))\Big]$$

Note that we do not introduce noise in our generator because we do not find it to work better. Also, we consider L1 difference between input $x$ and label $y$ in generator. On each iteration, discriminator would maximize $\theta_d$ according to the above expression and generator would minimize $\theta_g$ in the following way:

$$\min_{\theta_g}\Big[-\log(D_{\theta_d}(x,G_{\theta_g}(x)))+\lambda\big\|G_{\theta_g}(x)-y\big\|_1\Big]$$

With GAN, if the discriminator considers the pair of images generated by the generator to be a fake photo (not well colored), the loss will be back-propagated through discriminator and through generator. Therefore, generator can learn how to color the image correctly. At the final iteration, the parameters $\theta_g$ will used in our generator to color grayscale or edge-only images.

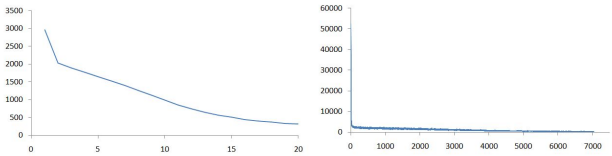
## 4. Dataset

To provide us with more distinctive color characteristics and wide variety of objects, we use a popular animation that contains myriads of creatures: Pokemon. We obtain 75 episodes of Pokemon which totals to 15 hours of videos from legal sources, then sample images per 50 frames of the videos with MATLAB as our data so that we don't have identical frames. To create one-to-one correspondence between features (gray-scale images or edge images) and labels (RGB), we convert the frames that we sample to grayscale images with MATLAB and generate edge-only images using Canny edge detection with OpenCV. Fig. 6 shows a sample of our data.

We shuffle all the sampled images and divide the dataset into training set, validation set and test set with a proportion around 8:1:1. In this process, we gather 22543 training examples, 2818 validation examples, and 2818 test examples, all of which are of size 360 pixels x 640 pixels. Each example includes the original RGB image, the grayscale image and the edge detected image. These frames are of low resolution, hence we down-sampled the images to 180 pixels x 320 pixels by zooming with spline interpolation to enhance the efficiency of training since it allows larger batch size and speed up the training process.

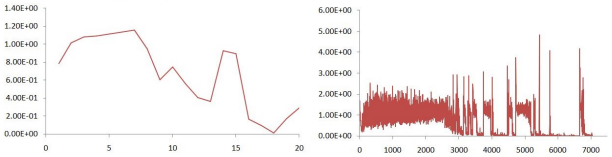
Grayscale Image Edge Image Ground Truth Image

Figure 6. Sample of Dataset



ConvNet loss per epoch ConvNet loss per batch

Figure 7. ConvNet loss



D loss per epoch D loss per batch

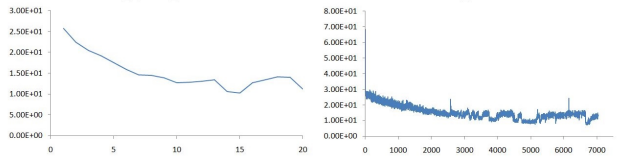
Figure 8. GAN D loss

## 5. Experiment and Discussion

To make the two models more comparable, we train both models on both types of input data. i.e., we train the ConvNet model for the two tasks on the grayscale-RGB training set and on the edge-RGB training set respectively and reiterate the same process for the GAN model. Furthermore, most hyper-parameters are shared amongst the two models. The number of epochs the model were trained on are the same: 20 epochs. The batch size are both 64 since larger batch size like 128 can't fit in the memory of the GPU for the GAN model. The learning rate for the ConvNet model, the generator and the discriminator of the GAN model are the same, which is 2e-4. The GAN model has an extra hyper-parameter: $\lambda$, the weight of L1 distance in the generator loss function. We set this value to 100, which gives the generator huge incentive to learn generating images close to ground truth. We use Adam optimizer for both models, but the beta1 for the discriminator and the generator of GAN are set to 0.5 instead of the default value 0.9. According to [14], decreasing beta1 is effective in training GANs to prevent the discriminator learn too fast and the discriminator loss drop to zero soon leading to the failure of learning for the generator.

### 5.1. Training Loss

To monitor the learning process of our models, model losses were plotted against epochs and batches. We ana-



G loss per epoch D loss per batch

Figure 9. GAN G loss

lyze the loss curves for training on grayscale images but the curves are similar for the edge-only images.

#### 5.1.1 ConvNet

The ConvNet loss curves are shown in Fig. 7, which depict steady decreases in both epoch losses and batch losses. It can be seen from the graphs that ConvNet model converges fairly quickly, especially for the batch loss. This is because of the simpler and non-adversarial nature of the model, as opposed to the cGan model. One thing we note during the training process of ConvNet is that, despite the batch loss seems to be converging at early stage, the quality of the prediction images still improve with more epochs by having sharper and brighter colors, and eventually converges at around 15th epochs.

#### 5.1.2 Conditional GANs

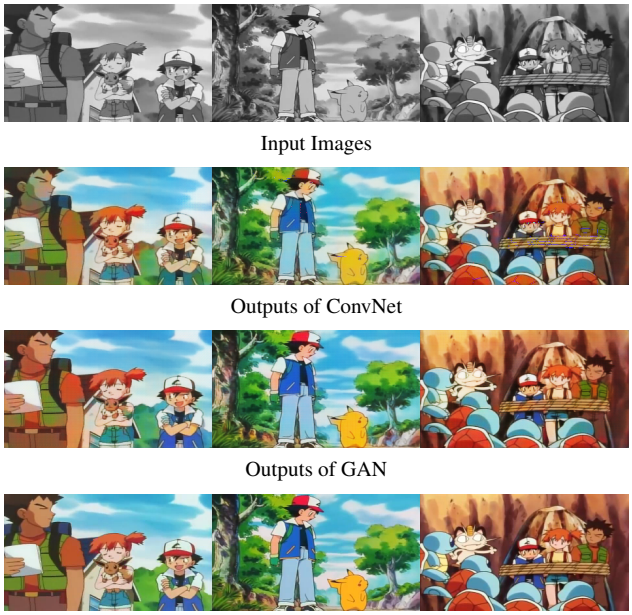
The discriminator and generator loss curves are shown in Fig. 8 and Fig. 9 respectively. From the curves, it can be seen that both generator and discriminator loss are decreasing over time. Generator loss decreases more steadily than discriminator, which has greater variance. It is interesting to note that we can see the adversarial nature of generator and discriminator in their loss graphs. At early epochs when G-loss is dropping quickly, there is a slight rise in D-loss, which means the growth of generator is making discriminator not able to distinguish between real and fake images. At around 14th epoch or 6000th batch, there is a spike in D-loss and it is also reflected by the dip in G-loss at the same epoch, and once the discriminator improves itself, D-loss decrease and G-loss slightly increase. The overall trend of both losses are decreasing and our results confirms that the model is learning well.

### 5.2. Generative Results

Fig. 10 shows some sample results from the test set of the task of colorizing grayscale images. It is noticeable that quality of the images generated by GAN is in general higher than the images generated by the ConvNet with L2 loss. Images generated by GAN are brighter and clearer. Sometimes even brighter than the ground truth image, e.g. the sample in the middle in Fig. 10. For images generated by ConvNet,

| | Grayscale | | Edge | |
|---|---|---|---|---|
| Distance | ConvNet | GAN | ConvNet | GAN |
| L1 | 19.19 | 16.25 | 50.68 | 39.09 |
| L2 | 1219.73 | 747.29 | 4720.35 | 3046.84 |

Table 1. Average L1 & L2 Distance between the generated images on the test set and the ground truth images



Input Images

Outputs of ConvNet

Outputs of GAN

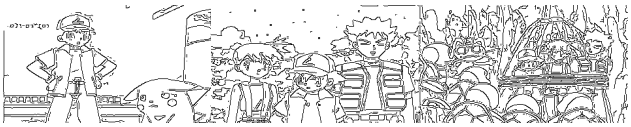
Ground Truth

Figure 10. Colorization of Grayscale Images on Test Set

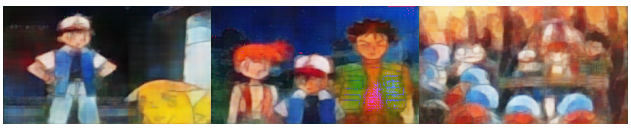
color in the area near the edges spreads across the edge more easily.

Fig. 11 shows some sample results from the test set of the task of colorizing edge-only images. With less input information, the generated images, which are blurry, are inferior to those from the grayscale images. However, it is notable that the images generated by ConvNet are no more comparable to those generated by GAN, which suggests the robustness of the GAN.

To evaluate the performance of our models quantitatively, we calculate the average L1 and L2 distance (per pixel-channel) between the generated images and the ground truth images on our test set. The results are shown in Table 1. Two observation can be made from the table: 1) GAN model generates images closer to ground truth in both L1 and L2 distance metrics than ConvNet model despite its objective is not directly minimizing the distance while ConvNet's is. This further confirms the superiority of GAN over ConvNet model, from both qualitative and quantitative perspective. 2) For both models, using grayscale images as input yields lower distance differences than using edge-only images as input. This is not surprising because grayscale images contains more information than edge-only images.



Input Images



Outputs of ConvNet

Outputs of GAN

Ground Truth

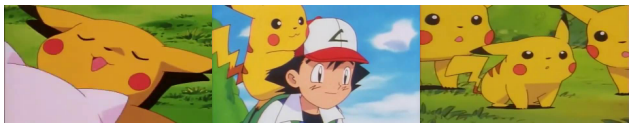
Figure 11. Colorization of Edge Images on Test Set

Generated by GAN

Ground Truth

Figure 12. GAN Common Failure: Exchange of hair color

Generated by GAN



Ground Truth

Figure 13. GAN Common Failure: Large Size of Pikachu

Input Output of ConvNet Ground Truth

Figure 14. ConvNet Common Failure: Colorization of Caption

### 5.3. Common Failures

Even though GAN performs better than ConvNet in terms of brightness and sharp edges, it is not perfect and has some weakness. We discovered some common failures particular to Pokemon, which may also occur in other animations. One of them is that with characters that share similar appearances, GAN may confuse the colors between them. Example of that is with Jessie and James in Pokemon (the two individuals in Fig. 12), the color of their hair is sometimes confused by having patches of hair color switched, since they share similar facial structure, expressions and clothing, and the only thing that differs them is mostly their hair. Another failure mode is that went the object is too large, GAN fails to color the enter object as a whole, and the object might partially blend into the color of the background. This might be cause by not having enough training data with large object or the filter sizes are not optimized. This phenomenon can be seen on the Pikachus at Fig. 13. Note that GAN gets the color of smaller Pikachus perfectly but have color patches either from the background or objects at vicinity on it when Pikachus are large. The third common failure happens in ConvNet model. When there are subtitles like the end in movies, the generated images will be a mess. Perhaps because the subtitles are too thin, causing the model confused about whether it is a huge block or many edges. Thereby, many flashing glitches are generated nearby the white subtitles. Also, we found from Fig. 14 and Fig. 15 that ConvNet has difficulty generating colors if the input image is a batch of black.

### 5.4. Generalizability on Different Animations

To further test the capability and generilizability of GAN model, we used the model trained on Pokemon animation to make predictions on unseen grayscale and edge-only images from different Animations or Manga. The test sets contains manga of similar styles as well as animations outside of Japan like Spongebob. To provide a realistic testset, many of the images are cropped by actual manga which we do not have ground truth of. We evaluate these results based qualitatively. The results are shown in Fig. 15. We can see that GAN with grayscale input can generate better and clearer results while edge input will lead to messy results. Many colors are even drawn outside the edge boundary. As to the color, surprisingly both model successfully color Doraemon blue and astro boy skin color. The shape of Doraemon is similar to squirtle so maybe that is the reason it gets colored blue. Astro boy gets skin color because it is human-like. However, spongebob cannot get colored well because the model has never seen such an object before and the color of Spongebob is very light.

Figure 15. Other manga colorization

## 6. Conclusion

In this work, we compared the quantitative and qualitative results of colorization using ConvNet and GAN to tackle the problem of colorization in Manga. The following conclusions can be drawn in our experiments.

- Although GAN is much difficult to train, the generated images are much better in terms of color brightness and sharpness, whereas images generated by ConvNet have patches of dimmer color caused by averaging the colors at vicinity in L2 loss optimization. GAN is also superior to ConvNet quantitatively by having generated images closer to ground truth in both L1 and L2 distance metrics.

- Grayscale input yields better colorization results than using edge-only input due to the extra information it contains. However, with GAN, the model is still able to produce reasonable colorization with edge-only images.

- The GAN model is able to generalize quite well on manga images when training on grayscale inputs, but not so much on edge-only input. It suggests that grayscale images extracted from color images are closer approximation to manga, and can be used to train the task of manga colorization.

## References


[1] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[3] 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*, pages 2672–2680, 2014.

[4] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE transactions on pattern analysis and machine intelligence*, 17(7):729–736, 1995.

[5] R. Ironi, D. Cohen-Or, and D. Lischinski. Colorization by example. In *Rendering Techniques*, pages 201–210, 2005.

[6] 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.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[9] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *ACM Transactions on Graphics (ToG)*, volume 23, pages 689–694. ACM, 2004.

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

[11] Y. Qu, T.-T. Wong, and P.-A. Heng. Manga colorization. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1214–1220. ACM, 2006.

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

[13] 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*, pages 234–241. Springer, 2015.

[14] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[15] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2010.

[16] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 277–280. ACM, 2002.

[17] L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. *IEEE Transactions on Image Processing*, 15(5):1120–1129, 2006.

[18] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.