Automatic Sketch Colourization

Yuanfang Li*
Stanford University
yli03@stanford.edu

Xin Li*
Stanford University
xinli16@stanford.edu

Abstract

We present an application of conditional GANs to colourizing landscape linesketch images, based on the Pix2Pix model. We experiment with using various pixel loss functions as well as data subsets to produce realistic colourings. The final model is able to generate acceptable colour images that represent a promising base upon which further processing can be performed to produce more artistic images suitable for practical applications like animation.

1. Introduction

Sketches have various applications in our daily life. Besides being a form of art, it was also used historically to record scenes and individuals. Nowadays, sketching is still a good way to graphically record and demonstrate an idea. Colourizing black-and-white sketches is challenging and time-consuming, but it can help make the sketches more vivid and dynamic, therefore providing more information.

This project aims to develop a model to automatically colourize line sketches of landscape images. We choose to focus on landscape images for two reasons:

1. Natural images are easier to colourize accurately due to standard expected colours for natural objects, i.e. we expect trees to be green and the sky to be blue, whereas a non-natural object like a car can realistically be any colour.

2. Automatic landscape colourization can be applied practically to fields like animation to reduce the amount of time spent on colouring backgrounds and allow artists to focus on characters.

Our final model takes as input a black and white sketch of a landscape along with a label, which is then put through a conditional Generated Adversarial Network (GAN) to produce a colour image as the final output.

*Equal contribution. Author ordering determined by reverse alphabetical order.

2. Related Work

Prior to the advent of deep learning, image colourization was largely achieved through one of two methods. In the scribble-based method, colour scribbles are applied to each region of the greyscale image which are extrapolated to colour the entire image[15]. This method relies on the concept that neighbouring pixels with similar luminance should also have simialr color, so the colorization problem can be solved by minimizing the color difference between neighbouring pixels, subject to the scribble constraints. In example-based methods, colour is transferred from a similar sample image to the target greyscale image[10]. These methods usually aim to minimize pixel loss in the form of L1/L2 or total variation loss[1] However, both these methods require considerable effort on the part of the user in providing reference scribbles and images. Further, because a single scribble/sample image cannot account for all possible colorings of the target image, multiple images from a large database need to be filtered to obtain a collection of reference images[4].

Deep learning approaches to the image colourization problem are able to overcome this issue and achieve better results with minimum effort from the user by leveraging the use of CNNs applied to large datasets[3]. These networks typically consist of a series of convolution and deconvolution/upsampling layers that can be thought of as an encode/decoder system. Unlike classification problems however, it is difficult to quantitatively evaluate the accuracy of the colouring, thus much of the current work focuses on developing a suitable loss function. A typical L1/L2 loss leads to desaturated colours in the output due to the averaging effect of these losses [24]. [2] proposes using a KL-divergence loss to predict the probability of each colour bin at a pixel instead of the colour itself, which can be thought of as a colourfulness loss. This method is applied to the CNN framework in [14] and to the VAE framework in [6].

[12] introduces a classification loss in addition to the pixel loss by jointly training a colourization network and a classification network. The two networks share low-layer weights, and the features extracted from the fully-connected layers of the classification network are used as input to later

layers of the colourization network. This increases accuracy of the final colourized image by introducing a prior on the type of image. Similarly, [9] applies a feature loss between the output and target images in addition to the pixel loss to obtain a realistic colour output from a line sketch portrait.

The loss problem is further exacerbated when using line sketches as the network must not only generate a correct colouring, but also luminance values and additional edges so that the textures and boundaries of objects in the image are acceptable[19]. This problem is not as evident in [9] due to the focus on faces, which have less variation in texture. The model in [19] shows that the use of feature loss on more varied categories leads to images that minimize the mean squared error but are not visually plausible. A weighted sum of adversarial, feature and pixel loss is required to generate realistic images, where the adversarial loss is used to constrain generated images to a natural image prior, and the pixel loss is used to stabilize training[8]. This method is applied by [19] on sketches with colour scribbles, with a heavier weight placed on the feature and pixel loss.

[13] also uses a conditional GAN to generate a realistic image given some input. This model is able to perform image translation on a variety of tasks including greyscale colourization, sketch colourization and inpainting. Whereas [19] conditions only the generator on the input, [13] also conditions the discriminator. Further, colour scribbles are not used, so there is no control over the colour palette used, but less user effort is required.

Finally, [7] performs colourization of greyscale landscape images using regression and demonstrates the importance of scene information as performance can be improved by learning a separate regressor for each category.

Our work applies the Pix2Pix model[13] implemented on Tensorflow[11] and trained on our collected dataset of landscape sketches using various pixel losses.

3. Methods

3.1. Conditional GAN

A Generated Adversarial Network (GAN) consists of two components: the generator, G, is trained to generate realistic images from some random noise vector, z, with the goal of fooling the discriminator, D, which is trained to distinguish between the target image, y, and the generated image, $G(z)$. In a conditional GAN, both the discriminator and the generator also receive an input image, x. The goal is to use x to direct the image generation process[17]. Then the generated image is $G(x, z)$ and the discriminator output is $D(x, y)$ for the target input-output pair and $D(x, G(x, z))$ for the generated input-output pair, where $0 \leq D \leq 1$ is a measure of the probability that the image pair is real and not generated.

During training, the generator attempts to minimize $\log\left(1 - D(x, G(x, z)\right)$, the log-probability that the discriminator classifies the generated image as fake, while the discriminator tries to maximize $\log\left(D(x, y)\right) + \log\left(1 - D(x, G(x, z)\right)$, the log-probability that the discriminator correctly distinguishes between real and generated images.

The model also adds a pixel loss, $\mathcal{L}(G)$, that acts as a measure of the similarity between the generated and target images and helps to stabilize training. Thus the final objective function is:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}(G)$$

where

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y\sim p(x,y)}[\log D(x, y)] + \mathbb{E}_{x\sim p(x),z\sim p(z)}[\log(1 - D(x, G(x, z)))]$$

and λ is the ratio between the pixel loss and the cGAN loss.

3.2. Pixel Loss

We experiment with using L1, L2 and Huber loss as the pixel loss:

$$\mathcal{L}_{L1} = |y - G(x, z)|$$

$$\mathcal{L}_{L2} = (y - G(x, z))^2$$

$$\mathcal{L}_{Huber} = \begin{cases} \frac{1}{2}(y - G(x, z))^2, |y - G(x, z)| \leq \delta \\ \delta(|y - G(x, z)| - \frac{1}{2}\delta^2), |y - G(x, z)| > \delta \end{cases}$$

From a colourization perspective, we expect that the L2 loss will lead to blurry images with less detailed colouring, as it favours many small, non-zero residual and is greatly affected by outliers, leading to application of a single average colour. We expect that the L1 loss will produce sharper, more detailed colourings, but will be more unstable to train as it penalizes residuals linearly. Finally, as the Huber loss follows an L2 loss for small residuals and L1 loss for large residuals, we expect that the resulting colouring will retain the detail of the L1 loss while remaining stable.

3.3. Architecture

Figure1 shows the generator and discriminator, each of which is a separate CNN, and the layer dimensions.

The generator takes as input the black and white landscape sketch x, and passes it through a series of convolution/encoder layers, followed by a series of deconvolution/decoder layers, to produce the coloured generated image, $G(x, z)$. Each layer consists of a convolution followed by batch/instance normalization and leaky ReLU activation, except for the last layer, which uses a tanh activation[5]. Dropout is applied during both training and test time to introduce random noise, z, to the model. A key feature of the generator in the Pix2Pix model is the connection from each

encoder directly to its corresponding decoder. These skip connections allow the model to bypass encoder/decoder layers that are unnecessary.

The discriminator takes as input the black and white landscape sketch x, and an unknown image (either the target or the generated image), concatenates them together, and passes the result through a series of convolution/encoder layers to produce a guess as to whether the unknown image is real or generated. Instead of producing a single probability value, the Pix2Pix model uses a PatchGAN, where the output is a 30x30 image, with each pixel representing the guess for a 70x70 patch of the image pair. The final discriminator output is an average over the patch outputs. Like the generator, each layer of the discriminator is a convolution followed by batch normalization and leaky ReLU activation, except for the last layer, which uses a sigmoid activation.

The motivation behind the PatchGAN approach is that instead of penalizing any difference between the generated and target images, it only penalizes differences within each patch of the image. As the model only considers local patches of the image, it is able to accurately model high frequency structure within the image, while the pixel loss is used to model low frequency structure.

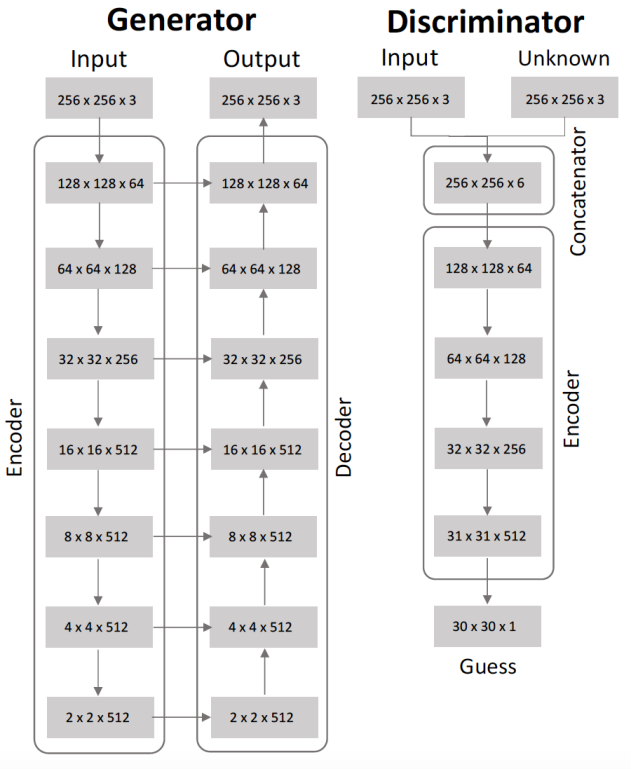

Figure 1. Architecture

3.4. Instance Normalization

Typical batch normalization normalizes across a batch of feature maps by some factor proportional to the mean and standard deviation across the batch during training. At test time, each feature map is normalized using a running average taken across each training batch. However, in instance normalization, a batch size of 1 is used and normalization with respect to batch statistics occurs at both training and test time as follows[20]:

$$\mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}$$

$$\sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - \mu_{ti})^2$$

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}$$

The goal of this method is to normalize the input with respect to the contrast of the image and has lead to improved results compared to batch normalization when applied to generator networks.

3.5. Activation Functions

[18] show that using leaky ReLU with a slope of 0.2 in GANs leads to better performance than regular ReLU, especially for images with higher resolution. Additionally, use of the bounded and symmetric tanh activation compared to unbounded ReLU at the last layer of the generator helped the model to learn more quickly to saturate light and dark colours equally. Finally a sigmoid activation is used at the last discriminator layer in order to output a binary class probability.

3.6. Evaluation Method

Evaluating the results is a difficult problem. Computing peak signal-to-noise ratio (PSNR) or mean squared error (MSE) are traditional methods to evaluate the similarity of two images. However both PSNR and MSE results have proven to be inconsistent with human perception. The methods used by [24] and [13] involve a Turing test requiring human participants, which is not realistic for our project.

Another evaluation method is structural similarity (SSIM)[23][21], which measures the perceptual correctness by comparing contrast, luminance and structure of two images[9]. SSIM is a computationally efficient way to measure the similarity between images. It is invariant to image scaling, rotation, and also insensitive to luminance and contrast change. Further, like the adversarial loss, SSIM is a structural loss and so is well suited to evaluating GAN

outputs[13].

$$S(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

4. Dataset and Features

In order to train our model, we need a dataset of images and their related sketches. Few studies have developed methods of transforming images into sketches. [9] uses a large log-scale face dataset as well as hand-drawn face sketches of famous artists. However, it is difficult to find a dataset with landscape sketches.

We developed an efficient method of preprocessing a large amount of images into sketches by using fuzzy logic image processing to perform edge detection[16]. The results generated from edge detection resemble pencil sketches well.

The general idea is as follows: To obtain the sketch, the input image is first converted to greyscale, and then converted to double-precision data. We use image gradient to locate breaks in uniform regions. If the gradient is not zero, then this pixel is located on the edge. Fuzzy inference system is defined and evaluated for edge detection. Finally, a image that resembles pencil sketch is generated based on the result.

Figure 2. A sample from the dataset

We created our own dataset of 5000 images from flips and crops of 2500 images. We further split this into 3500 training images, 1000 validation images and 500 test images.

These images were obtained through Google Image Search by modifying a crawler script to download the images returned. The method provided by [22] allows us to download 100 images of each keyword. We modified it to download all images returned by Google Image Search, which is about 700 images of each keyword. To ensure that our dataset included images of different landscapes, we varied the key terms in our search. We then manually went through the images to remove any irrelevant or broken images. We further augmented the data by cropping and horizontally reflecting these images. Our final images are 256x256 for both the linesketch input and photo target. A sample dataset image is shown in Figure 2.

5. Experiments and Results

5.1. Experiments with Different Pixel Losses

We used a pixel loss to penalize the difference between the colourized outputs and the ground truth images, hence forcing the outputs to be similar to targets. We experimented with different kinds of loss to find the one with the best performance.

We obtained preliminary results by training the model on the full dataset for 10 epochs with a learning rate of 0.0001 using SGD optimizer for the discriminator and Adam optimizer[5] with $\beta_1 = 0.5$ and $\beta_2 = 0.99$ for the generator, chosen through hyperparameter search. Batch size of 1 was used and dropout with probability of 0.5 is applied to the generator layers during training and testing phase[13]. We optimize over the weighted sum of the GAN loss and the pixel loss for different choices of loss and weight ratio λ. For the pixel loss, we tested L1 loss, L2 loss and Huber loss with λ values of 10, 100 and 1000.

Several sample outputs from the validation set are shown below. The input line sketch is on the left, the network generated output is in the middle, and the ground truth target image is on the right.

Figure 3. Result: using L1 loss, SSIM = 0.398

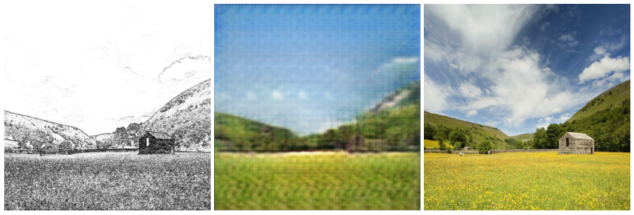

Figure 4. Result: using L2 loss, SSIM = 0.389

Compared to the target image, the output image using L1 loss 3 is blurry, but the network has learned the correct colouring for mountains and grass compared to sky and the image is reasonable. There are even white patches in the sky that could be clouds even though they are not obvious in the line sketch. Using L1 loss, we obtain a reasonable result.

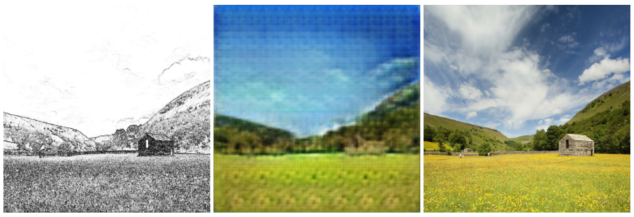

Figure 5. Result: using Huber loss, SSIM = 0.385

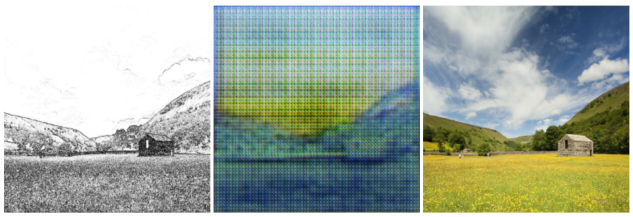

Figure 6. Result: using no pixel loss, SSIM = 0.066

Loss Type	Relative improvement
L1-loss, λ=10	0.056
L1-loss, λ=100	**0.471**
L1-loss, λ=1000	0.294
L2-loss, λ=10	0.080
L2-loss, λ=100	0.450
L2-loss, λ=1000	0.236
Huber, λ=100, δ=0.25	0.235
Huber, λ=100, δ=0.5	0.252
Huber, λ=100, δ=1	0.154

Figure 7. SSIM results

The output image generated using L2 loss 4 is blurrier than the L1 loss result, but is still acceptable. This is expected, as the L2 loss is more sensitive to large residuals and tends to ignore small residuals and the output is more sensitive to outliers. Therefore the model produces a non-sparse residual, which results in blurring from a human perspective. Because of the averaging effect, the model colourizes the image with much fewer colour details. As we can see from the outputs above, the background (sky) is colourized with the same light blue and the clouds have almost disappeared, compared to the L1 loss result.

Huber loss leads to patchy features in the output image as shown in Figure 5. There are black point-like patches in the image, especially in the grass part. This happens in many other test images too. This is unexpected, as we hypothesized that the Huber loss would produce the best results.

We also tested the model without adding any pixel loss. Like the Huber loss, the output image in Figure 6 has grid-like features. The SSIM result is much worse than any of the results above, and even falls below the baseline SSIM between the line sketch and target image, which is 0.311.

A possible explanation for the gridded appearance of the Huber loss and GAN loss only models is that there is too much emphasis on high-frequency structure. The original Pix2Pix model generated similar images when the local patch size of the PatchGAN used was too small, leading to tiling artifacts[13]. Since the model learns low frequency structure using the L1 loss, removing it altogether leads it to focus only high frequencies.

Table 7 shows the SSIM results for the full dataset, which is the average of 500 test images in total. State of the art models using line sketches are able to achieve SSIM values of 0.85 [9]. This result is obtained based on a very large dataset with over 330,000 human face images (combining CelebA, LFW, CUFS and sketches of famous Dutch artists). Clearly our model falls below this. However, there is still some improvement when compared with the SSIM of the original line sketch. Considering that we only have a small dataset of 5000 images, and landscape colourization is a much harder problem than human face colourization[19], our results are still acceptable.

5.2. Experiments with Subsets

One problem with using the full dataset to train a single model is that the ground is always coloured green and the sky is always coloured blue in the output images. One possible reason is that many of the training samples have similar patterns of green ground and blue sky. The similarity of sketches also makes it difficult for the generator to learn colourings from the complete dataset, as shown in Figure 8.

Figure 8. A less successful result: colourize beach as grassland

The generated image appears to be a field while the target image is actually a beach. This is likely due to the fact that from the line sketch it is difficult to determine what the image should be, thus the network colours the ground green as the majority of natural landscape images are green.

In order to tackle this problem, we divided our dataset into several subsets (i.e. city, beach, trees/grassland) and trained 3 separate models. We provide scene information in the form of a label to our test image and choose the trained model based on the category of the sketch as in [7]. Although this requires more work on the part of the user, we

believe this is reasonable since even humans cannot determine between a line sketch of a grassland and a beach. Further, considering possible applications in animation, it is highly likely that the user would know what the landscape image represents.

Number of epochs	SSIM (base: 0.249)	Relative improvement
10 epochs	0.380	0.521
20 epochs	0.432	0.727
30 epochs	0.467	0.877
40 epochs	0.489	0.966
50 epochs	0.468	0.879

Figure 9. Average SSIM results using subsets

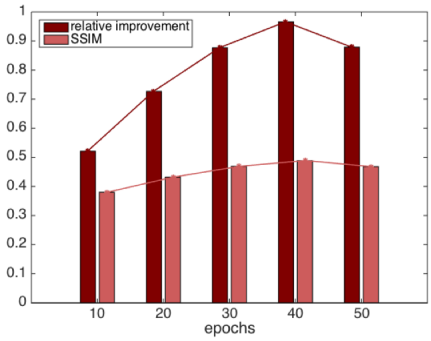

Figure 10. Average SSIM results using subsets

Figures 9 and 10 show the average validation SSIM across the 3 category-specific models trained. Our final test relative SSIM improvement obtained on a model trained for 40 epochs is 0.917. Although the SSIM is still lower compared to state-of-the-art models, it performs significantly better than the single model trained on the complete dataset.

Figures 11, 12 and 13 show sample images generated by our final model for each of the scene categories. Despite the low SSIM, the model is still able to learn some interesting features. Figure 11 shows that the model has learned the general appearance of light/water reflections. In Figure 12, the line sketch has watermarks in the form of diagonal lines on the bottom and words/images on the top. However the model is able to infer that these features are not typically present, so the generated image has no lines in the grass and the words are blurred and appear as clouds in the sky.

Finally, Figure 14 shows the generated image when the model is applied to a real line sketch. Compared to a generated line sketch, the input image contains fewer details, i.e. there are no lines indicating water reflections or individual window lights. However, the model is still able to infer these in the generated colour image.

Figure 11. Result: city

Figure 12. Result: forest/grassland

Figure 13. Result: beach

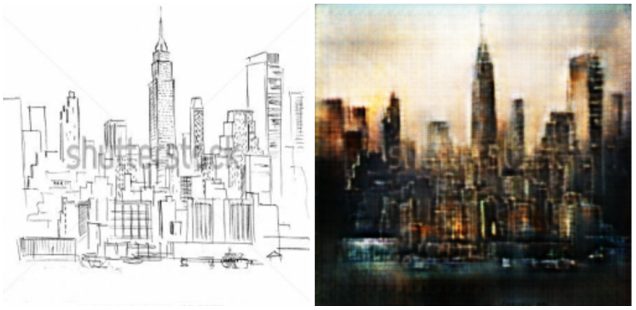

Figure 14. Result: real sketch

6. Conclusion and Future Work

Our experiments demonstrate the potential use of the Pix2Pix model for colouring landscape line sketches. By training separate models for each landscape category, we are able to introduce scene information into the model and achieve relative SSIM improvement of 0.917 when using a combined adversarial loss and L1 pixel loss. Although this does not reach the current state-of-the-art for line sketch colourization as applied to portrait images, landscape colourization is a much harder problem due to higher

variations between images. Despite the low SSIM, the model is still able to learn some useful features.

A large limitation of the model is due to the small dataset used, thus a clear next step would be to train the model on a larger dataset. Beyond this, the current model attempts to generate photo-realistic images from sketches but practical applications like animation usually require artistic images. Thus a possible extension is to use the generated image as a colouring template and apply style transfer to obtain a more artistic image.

References

[1] A. Bugeau and V.-T. Ta. Patch-based image colorization. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3058–3061. IEEE, 2012.

[2] G. Charpiat, M. Hofmann, and B. Schölkopf. Automatic image colorization via multimodal predictions. *Computer Vision–ECCV 2008*, pages 126–139, 2008.

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

[4] A. Y.-S. Chia, S. Zhuo, R. K. Gupta, Y.-W. Tai, S.-Y. Cho, P. Tan, and S. Lin. Semantic colorization with internet images. In *ACM Transactions on Graphics (TOG)*, volume 30, page 156. ACM, 2011.

[5] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu. How to train a gan? tips and tricks to make gans work. `https://github.com/soumith/ganhacks#authors`, 2016.

[6] A. Deshpande, J. Lu, M.-C. Yeh, and D. Forsyth. Learning diverse image colorization. *arXiv preprint arXiv:1612.01958*, 2016.

[7] A. Deshpande, J. Rock, and D. Forsyth. Learning large-scale automatic image colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 567–575, 2015.

[8] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[9] Y. Güçlütürk, U. Güçlü, R. van Lier, and M. A. van Gerven. Convolutional sketch inversion. In *European Conference on Computer Vision*, pages 810–824. Springer, 2016.

[10] R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong. Image colorization using similar images. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 369–378. ACM, 2012.

[11] C. Hesse. Image-to-image translation in tensorflow. `https://github.com/affinelayer/pix2pix-tensorflow`, 2017.

[12] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4), 2016.

[13] 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.

[14] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016.

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

[16] P. Melin, O. Mendoza, and O. Castillo. An improved method for edge detection based on interval type-2 fuzzy logic. *Expert Systems with Applications*, 37(12):8527–8535, 2010.

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

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

[19] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. *arXiv preprint arXiv:1612.00835*, 2016.

[20] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[21] A. Vacavant. A python module for computing the structural similarity image metric (ssim). `https://github.com/jterrace/pyssim`, 2016.

[22] H. Vasa. Google images download. `https://github.com/hardikvasa/google-images-download`, 2015.

[23] Z. Wang and E. P. Simoncelli. Translation insensitive image similarity in complex wavelet domain. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 2, pages ii–573. IEEE, 2005.

[24] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.