Image Colorization using Conditional Generative Adversarial Network (CGAN) with Local Hints

Chun-Hsiang Chan, Qiming Yuan, Yuchu Wang, Zijie Chen
{kenhchan, yuanqm, yuchuw, zijiech}@umich.edu

Abstract

In this work, we propose novel models of Local Hints Networks that combine the CGAN with local hints to improve the quality of generated images. We evaluate the performance of our models qualitatively and quantitatively by using various evaluation metrics. Our experiments show that, in comparison with other models, such as convolutional neural network (CNN), CGAN, and CNN with local hints, our Local Hints Networks perform much better, especially the one where local hints are involved in both the generator input and the generator loss function. And it is shown that our network is evidently useful for complex, real-world images. Besides, we creatively apply our model to colorize videos, which also produces decent results.

1 Introduction

Colorization of images has gained increasing attention and has become prevalent in recent years. It is an easy way to rekindle dormant memories by colorizing someone's favorite black and white family photos or to express artistic creativity by colorizing the world-renowned image from the last century as shown in Fig.1. Image colorization is the technique of taking a grayscale (black and white) image as an input and then producing an output of colorized image that represents the ground-truth semantic colors and tones of the input.

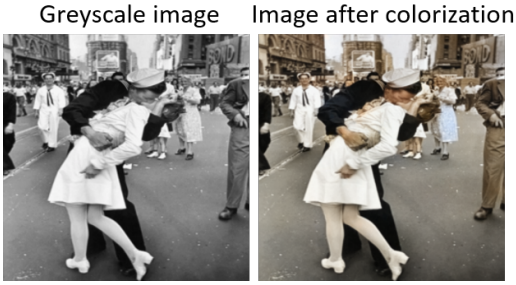

Figure 1: Example of image colorization

In computer graphics, two typical approaches of image colorization are user-guided edit propagation and data-driven automatic colorization. For the user-guided edit propagation, it can obtain impressive results based on the strong intervention or implication of users. However, it requires intensive user interaction and explicit precise selection for different semantic regions which is hard to achieve. For the data-driven automatic colorization, it matches the grayscale photo to an exemplar color image in the database and uses colors from photos or learns the parametric mapping to color from large-scale image data. Several deep learning models have been involved in achieving this goal, such as convolutional neural networks (CNN), generative adversarial networks (GAN), and Conditional GAN. Although the deep learning models are fully automatic and resolve the intensive need for user

interaction, they have some artifacts, such as human hands often colorize with background color and lead to an unrealistic image. This issue has intensively occurred in the CNN-based model and has become a critical issue to resolve in this area. Zhang et al.[1] overcome this issue by combining the user-guided edit approach with the automatic CNN model. The user will provide a few colorized points (local hints) to guide the colorization of CNN model. Besides CNN, GAN have gained more attention in solving image colorization. Due to the adversarial structure, the model can produce a more realistic image and automatically correct the unrealistic colorization on the generative networks. Besides the GAN model that generates the color image from noise, Nazeri et, al.[2] provides a GAN model conditioned on the input with a gray scale image. These conditional generative adversarial networks (CGAN) further improve the performance of image colorization.

In this project, we propose the Local Hints Networks based on the CGAN model to improve the performance of the image colorization from two perspectives. The local hints are little squares adopt the mean of the pixel colors from the ground truth and thus provide guidance for the model. First, with the local hints, we can increase the training efficiency of the CGAN models. Since our hints are sparse, they will not lead to large increase in the computational cost. We examine that our Local Hints Networks have shorter training time compared to the CGAN models without hints. Second, the local hints guide for the realistic image colorization. Since CGAN models guide the generated colorization merely with its adversarial structure, which is least efficient and the generator might lose the realistic considerations during the training. With local hints, the model can learn the partial realistic colorization and automatically correct the generator with right colorization.

To achieve these two goals and implement Local Hints Network based on CGAN. We first implement the architecture of the CGAN model introduced by Mirza et, al.[3]. Based on such CGAN backbone, we combine the local hints in either the generator loss function or both generator input and the generator loss function. This local hint approach is inspired by local hints with CNN proposed by Zhang et al. [1]. And note that the most innovative part of our project is to involve the local hints in the generator loss function and generator input. In detail, our CGAN models take the grayscale images as inputs. The generators will create fake images in the RGB format, and the discriminator is a binary classifier to distinguish real and fake images. We compare our proposed models with the previous researches: CNN model, the local hints CNN model proposed by Zhang et al. [1], and the CGAN model [3]. Overall, we found that our Local Hints Network, involving local hints in both input and generator loss function, outperforms all other methods in both qualitative analysis and quantitative measurement with evaluation metric of PSNR and SSIM. As an extension, we also apply our best network to image colorization for videos and the reconstruction looks good.

The report is organized as follows: Section 3 introduces the CGAN structure that we use as a backbone for our proposed methods. Section 4 present the design of our proposed Local Hints Networks. Section 5 shows the experiments and our discussions. We will introduce the datasets, the evaluation matrix, and others set up in section 5.1. Section 5.2 provides the qualitative results and section 5.3 is the quantitative analysis. Besides, in section 5.3, we further show the video colorization results. And finally, we conclude our work in section 6.

2 Related Work

User-guided edit propagation and data-driven automatic colorization are two focuses for image colorization. The former have been intensively studied for the traditional colorization algorithms, and the later are able to accomplish based on the deep Learning methods.

Traditional User-guided Methods. The effective algorithm of user-guided edit propagation was first discussed by Levin in 2004 [4], which used low-level similarity metrics to achieve local control. Local control uses local hints for colorization, which is different from the global control integrating global hints to networks. Later, to reduce users' effort, methods focusing on designing better similarity metrics [5] and long-range connections [6] were introduced. Over the last decade, researchers have been gradually focused on global control instead of local control. Algorithms involving palette-based methods [7] or guiding interactive user edits [8] were developed later. Since the user-guided algorithm relies heavily on the user's selection of colors, semi-automatic colorization was developed to address the limitation. Methods using an example-based algorithm [9] and image analogies [10] worked remarkably well when the examples and the gray-scale images are similar. However, there comes difficulty when the grayscale objects are rare.

Deep Learning Automatic Methods. Recently, deep Learning methods were designed to handle the fully automatic tasks. The neural networks have been proved to be powerful in areas like photo enhancement [11], style transfer in painting [12] and image blending [13], has shown impressive results for image colorization. CNNs and GANs are two neural network models highly involved in image colorization. Large-scale image collections were utilized to train CNNs to achieve realistic results, where direct output colors are assigned to the grayscale images [9]. In addition, Zhang et al. [1] provide a hint based CNN model to improve the colorization quality. In 2014, Goodfellow et al. [14] proposed a new type of generative model: generative adversarial networks (GANs) to generate images from random noise. After that, Unconditional GAN is used to achieve image manipulation guided by user constraints [15], image style transfer [16], and photo-realistic single image superresolution [17]. Later, conditional GANs are applied to the image to image translation [18] and image colorization [2]. Some researchers also start to look into the comparison between CNN and GAN model for image colorization [19].

3 Preliminaries

In this section, we introduce the architecture of CGAN that we reproduced. That is, we reconstruct the conditional generative adversarial networks [3] and use it as a back bone.

3.1 Conditional Generative Adversarial Networks

Generative Adversarial Network(GAN) was proposed by Goodfellow et, al.[2]. The GAN network contain two parts, which are generators (G) and discriminators (D). For image colorization, the generator is trained to generate artificial images and the discriminator is a binary classifier deciding whether the input images are artificial (from generator) or real (from ground truth data). In the traditional GAN, the generator builds the output image from a prior noise distribution and try to fool the discriminator as much as possible.

After that, Conditional Generative Adversarial Network (CGAN) is proposed by Mirza et, al.[3]. CGAN train both the generator and discriminator conditioned on some extra information. In our task, we input the grayscale images as the extra information. That is, the generator builds the output image from a gray-scale image instead of noise distribution. In particular, the generator takes gray-scale images as input and generates the colorized 3-channel RGB images. The discriminator is a binary classifier trained with both ground-truth images and the generated RGB images, and aim to distinguish the input sources. The objective function of our CGAN is

$$\min_{\theta_g}\max_{\theta_d}[\mathbb{E}_{y\sim p_{data}}(\log D_{\theta_d}(y)+\mathbb{E}_{x\sim p_{data}}(1-\log D_{\theta_d}(G_{\theta_g}(x)))+\lambda||G_{\theta_g}(x)-y||_1)], \quad (1)$$

where x represents the grayscale image and y represents the colored image. θ_g and θ_d are the parameters for generator G and discriminator D respectively. The objective of generator is to minimize the function, and the objective of discriminator is to maximize the same function. That is the adversarial identity of GAN. Besides, in order to constrain the modification we made, an extra regularization term is added to the generator, a L1 distance between the generated image and ground truth is added to generator's loss function with a weight λ.

Generator We use encoder-decoder style architecture to build the generator. In addition, we utilize the technique of UNet [7] to let the decoder retain the details form in easier way. We add skip-connections between the encoder and the decoder. This is implemented by adding the feature map between the encoder layers and the corresponding decoder layers. In the encoder, each block contains a convolutional layer (with stride = 2), a batch normalization layer and a Leaky ReLU activation function. In the decoder, each block contains a transpose-convolutional layer (with stride = 2), a batch normalization layer and a Leaky ReLU activation except the last block. The last decoder layer has a transpose convolutional layer and a tanh activation function to gives $[-1, 1]$ values in the prediction. The ground truth RGB images and input images are also normalized to $[-1, 1]$. More detail please refer to the Fig 2.

Discriminator The discriminator is a binary classifier (1-channel output indicating whether the given input is real or fake) with only the encoder and a sigmoid activation function. Each block in the encoder contains a convolutional layer (with stride = 2), a batch normalization layer and a Leaky

ReLU activation function. The final sigmoid activation function is used to bound the output to [0, 1] range. More detail please refer to the Fig 2.

4 Method

In order to improve the performance of basic CGAN and make it feasible to do colorization for some unusual objects. We build two different Local Hints Networks to provide extra information for the CGAN: one with the local hints only involved in the generator loss; the other one uses the local hints as a part of the generator loss as well as the input.

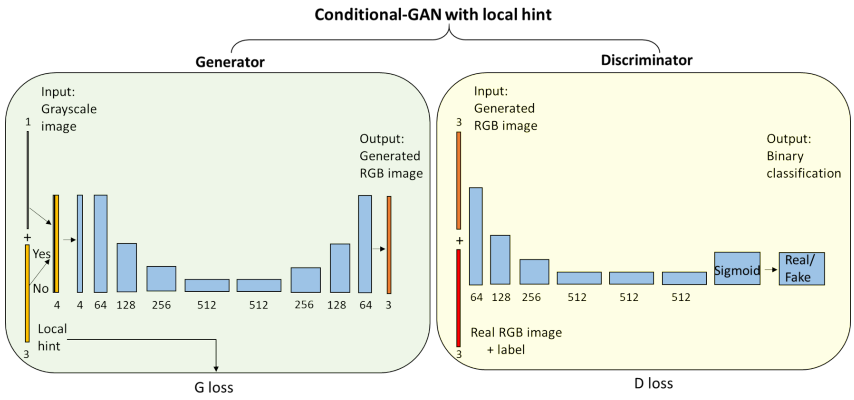

Figure 2: Schematic of conditional-GAN with local hint model

4.1 Local Hints

In this study, the local hints are defined as sparse, small and square points with the size of 2×2 or 4×4 pixels. These little squares adopt the mean of the pixel colors from the ground truth and thus provide guidance for the CGAN. The corresponding mask are filled with ones and zeros, which one indicates the region with local hints and zero without local hints. The number of local hints introduced ranges from 20 to 25, and they are randomly distributed without overlapping. During the pre-processing, the RGB format mask containing local hints will be generated, which has 3 channels with the same size as the ground truth image.

4.2 Local Hints as the Generator Loss

For the first Local Hints Network, we include the local hints only in the loss of the generator. As a result, the objective function of the generator would be changed to the following:

$$\min_{\theta_d}[E_{x\sim p_{data}} \log(1 - D_{\theta_d}(G_{\theta_g}(x))) + \alpha||G_{\theta_g}(x) - y||_1 + \beta||G_{\theta_g}(x)\, m_{hint} - y_{hint}||_2] \quad (2)$$

As is shown above, one new term is added. Specifically, y_{hint} is the mask containing local hints, while m_{hint} is another mask having the same size as the gray-scale image but filled with zeros and ones. There are ones only in the region corresponding to the region with local hints in y_{hint}. So, the multiplication of the generator output $G_{\theta_g}(x)$ and the mask m_{hint} would give us the generator prediction of colors for local hints. As a result, the newly-involved term will regularize the model by penalizing the difference between the predicted value of local hints and the real local hints. In this research, we use L2 regularization in stead of L1 regularization because L2 is computationally efficient and stable. Because the local hints are only used as supervisory information and included in the loss function, the input is still a one-channel matrix representing the gray-scale image. The input will not include any information from the local hints. And finally, we will get a model from the generator similar to that from basic CGAN. And this could limit the use for further application like colorization based on user-guided local hints.

4.3 Local Hints as the Generator Input and Loss

Different from Sec. 4.2, in this part, we not only include local hints in the generator loss, but also in its input. Introducing this change will not change the loss function of the generator, so the objective function would be the same as that in Sec. 4.2. But the input matrix would have 4 channels instead, with one gray-scale channel and 3 RGB-format channels representing the local hints. From this Local Hints Network, finally, we can get a different model from basic CGAN and our first model in Sec. 4.2. And it can pave the way for further application of colorization using user-designed local hints.

5 Experiments

5.1 Experimental Setups

Dataset. In this research, we only need the colorful images as the input and we don't need the labels for the images. As a result, the only requirement for the datasets is that they contain colorful images. In detail, we use two different datasets for the colorization task and analyze their prediction separately. One of the datasets is the SpongeBob dataset extracted from the animation. This dataset contains 3000 images having the size of 224×224. Because this dataset comes from an animation, the images in this dataset have simple structures and color composition, and the brightness and saturation do not change a lot compared to the photos of real world objects.

The other dataset used in this study is the famous ImageNet ILSVRC 2012 [20], which contains real world photos. Also, this dataset is a huge one with more than 1.3 million images in the training dataset and 100,000 images in the testing dataset. During the training, the images are resized to 224×224 to keep the same as that in SpongeBob. Because the images in ImageNet2012 are from the real world, the structure, brightness and saturation are more variable, which can increase the computational cost and instability of the models.

Evaluation Metrics. Besides evaluating the results with our eyes, we provide several numerical analysis to measure our results. First, we compute the peak signal-to-noise ratio (PSNR) between the generated images and the original images. To calculate PSNR between two images, we calculate the mean square error for three RGB channels of the RGB image and then use it to calculate PSNR. That is, the MSE of each channel x is

$$MSE_x = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{3}$$

The total MSE is the sum over each channel's MSE:

$$MSE_t = \sum_{x \in \{R,G,B\}} MSE_x, \tag{4}$$

and the PSNR is equal to

$$PSNR = 10 * log(\frac{MAX_I^2}{MSE_t}), \tag{5}$$

where MAX_I is the maximum value of a pixel.

Second, we use structural similarity index (SSIM), since PSNR relies strictly on numerical comparison and does not actually take any level of biological factors of the human vision system into account, such as the structure of the image. Specifically, SSIM is correlated with the quality and perception of the human visual system (HVS color model). Instead of using traditional error summation methods, the SSIM models image distortion as a combination of three factors that are loss of correlation, luminance distortion, and contrast distortion. Given two image x, y, the SSIM is as follow:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{6}$$

where μ_x is the average of x; μ_y is the average of y; σ_x^2 is the variance of x; σ_y^2 is the variance of y; σ_{xy} is the covariance of x and y; $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator; L is the dynamic range of the pixel-values, and $k_1 = 0.01$ and $k_2 = 0.03$ by default.

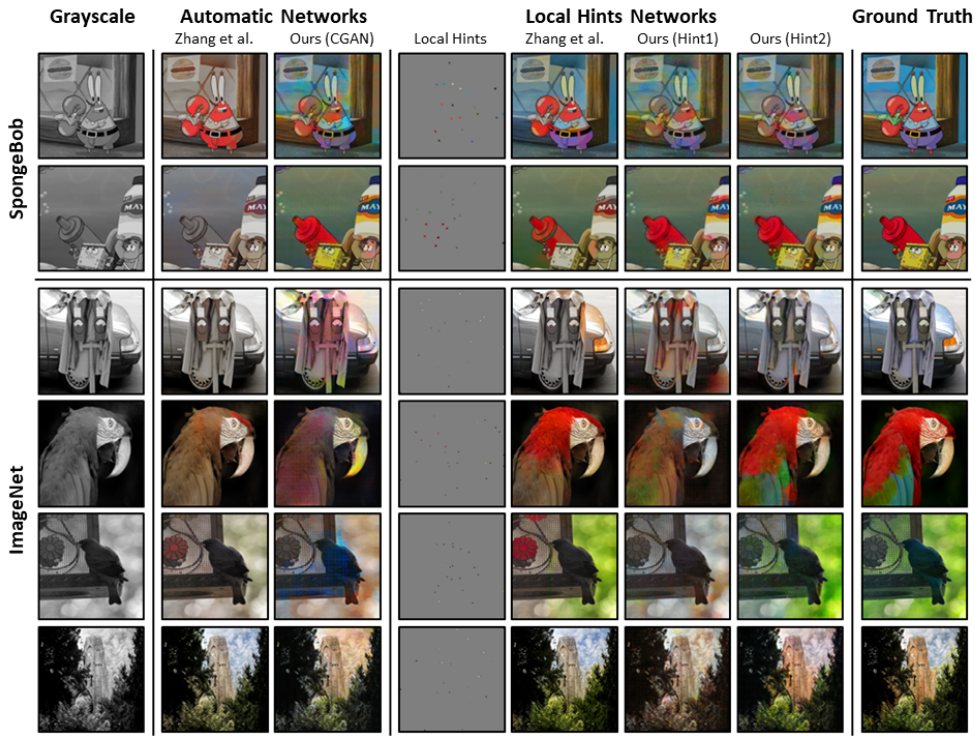

Figure 3: **Selected examples of colorization** The results come from five different colorization methods. The first column shows the grayscale inputs. Column 2 and 3 are the automatic models, including previous model using CNN and our model using basic CGAN. Column 4 shows the local hints used in three Local Hint Networks. Column 5 shows the seminal model of (Zhang et al. 2017), combing local hints with CNN. And our models are show in Column 6 and 7. Our first Local Hint Networks Hint1 includes local hints in the loss function of the generator, and our second model Hint2 involves the local hints in both the generator loss function and the input. And the final column shows the ground truth. Among these examples, Row 1 and 2 come from the experiment using the Spongebob dataset, while Row 3-6 from the experiment using the ImageNet dataset.

5.2 Qualitative Analysis

Fig. 3 above shows qualitative results of our methods and the previous models. In this study, we first train our models on the SpongeBob dataset. Among a large range of hyperparameters, we select the optimal ones. We set the batch size to be 50, and the learning rate to be 1e-4. For the two Local Hints Networks, we introduce 20 random distributed hints of the size of 4×4 for the training on Spongebob dataset. These three models (our basic CGAN model and two Local Hints Networks Hint1 and Hint2) are trained for more than 500 epochs. We have also trained two previous models as the reference. One is the automatic colorization using CNN and the other one combines local hints with the CNN algorithm. For the second algorithm, the local hints only appear in the input, which is different from involving local hints in the loss function. And as we can see from the Fig. 3, for the SpongeBob dataset, all of our 3 models get good colorization results. And the previous automatic model fails while the local hint-involved model performs well. Comparing our three models, it's hard to tell which network performs best simply by looking the images, which is because of the simple color distribution and structure of the animation images.

As a contrast, in Row 3-6 of Fig. 3, there are examples from models using the dataset of ImageNet2012. After probing the hyperparameters, we set the batch size to be 50, while a changing learning rate is used in this part. The initial learning rate is designed to be 1e-3 but it will decrease linearly until 1e-4 within 15 epochs. And for the Local Hints Networks, there are 25 random distributed hints of the size of 2×2 used in the training on this dataset. As we can see from Fig. 3, when using the ImageNet2012, the difference among our three models is more distinguishable. Even with our eyes, we can find that, our second model Hint2 can achieve much better results than the basic CGAN and

the first Local Hints Network Hint1. Taking the two previous models into account, we can still find that the Local Hints Network can do much better compared to others. And if looking at some details of the images like the feather of the bird in Row 4 of Fig. 3, we can argue that our second model Hint2 in Column 7, using hints in the input and the loss, performs better than the baseline Local Hints Network in Column 5 qualitatively.

5.3 Quantitative Analysis

Model / Dataset	Zhang et, al CNN	Our CGAN	Zhang et, al local hint CNN	Our Hint1 CGAN	Our Hint2 CGAN
SpongeBob	25.08/ 0.90	28.61/ 0.88	30.61/ 0.89	29.08/ 0.87	30.08/ 0.89
ImageNet 2012	20.83/ 0.88	23.83/ 0.89	26.08/ 0.89	25.56/ 0.89	31.08/ 0.91

Table 1: **PSNR and SSIM for each model on two dataset** The results come from five different colorization methods, which have been introduced before. In each block, the left denote the PSNR and the right denote the SSIM.

According to Table 1 above, We discuss the PSNR and the SSIM measurement separately.

First, we can observe that the Local Hints Network indeed performs better than the model without hints in PSNR. For the CNN model, the PSNR is 25% higher than the un-hinting-based model. For the GAN models, the Hint2 model performs the best with PSNR = 30.08 and 31.08 in two data set. Taking two dates set to score on average, we have the Hint2 model have 19% higher PSNR than the CGAN model, and the Hint1 model has 5% higher PSNR than the CGAN model.

Now, we make our discussion on different datasets. For the SpongeBob dataset, three Local Hints Network is much the same. Thus, we can not tell which models are the best. We think those models have similar PSNR because the SpongeBob dataset contains relatively simple images and either model is easy to learn the colorization pattern. On the other hand, Imagenet2012 is complex enough for us to examine the power of our Hint2 model. The model Hint2 outperforms the rest of the models, which is the same as we argue in sec. 5.2. The PSNR score of the Hint2 model is 20% higher than the rest of the models. This shows that involving the local hint in both the G loss and the input is a nice solution for CGAN design on image colorization, and such architecture can serve for real-world image colorization and produce good quality images.

For the SSIM ratios, all the models have high SSIM scores. Since our models involving grayscale ground truth images as the input and generate the color distribution to color the image, This indicates that our models do not produce much noise during the image colorization process, which is a good property that we aim to maintain.

5.4 Video Colorization

After getting decent results from our model using local hints, to extend our work to a higher level of novelty, we also conduct the colorization for a short video using hints from frame to frame. There is a gap between image colorization and video colorization because there are little information depicting the connections between different frames if we use the automatic algorithm. However, in this study, we treat the video as separate frames and consider using the Local Hints Networks, where local hints from the ground truth may provide information of relationship between different frames. The example of our experiment is a short movie clipped from the movie *Avengers: Endgame*. About 400 frames are extracted from the video in sequence. We applied our second model Hint2 in Sec. 5.2, which involves local hints as the generator input and loss for image colorization. And then, we stitch these frames together to recreate the video with color.

When training, we use a batch size of 50 and a decreasing learning rate starting from 1e-3 and ending at 1e-4, which is the same as those in Sec. 5.2. However, given the existence of unusual colorization in this video, for example, the unusual colors of Hulk and infinity stones, we increase the number

of hints to provide more details for the Local Hints Network. We introduce 50 random distributed local hints with size of 4×4 for the training and this network is trained for 15 epochs using the ImageNet2012 training dataset with 1.3 million images. As is shown in Fig. 4, the experiment result shows that we achieve colorization in general for some large distinct objection like Hulk and the six color infinity stones. However, since our local hints is randomly generated from each images, the network may not get information from the stones for some frames, so the colors for six infinity stones would keep changing in the reconstructed video.

Figure 4: One selected frame as the example of video colorization based on CGAN with local hints model

In this experiment, it is shown that the main challenge for video colorization is to achieve temporal consistency while remaining faithful to the reference style. This experiment has also provided us with some inspiration for a future improvement that the selection for the local hints position is extremely essential. In the future work, we will attempt to introduce a recurrent framework that unifies the semantic correspondence and local hints selection which will provide better guidance for the colorization of every frame, thus ameliorate the inconsistency issue.

6 Conclusion

In this study, we propose two new networks based on CGAN with local hints for image colorization. One involves the local hints in the loss function of the generator, while the other one both in the loss and the input. In our experiment, it is proved that adding local hints can greatly improve the performance of colorization. And we have also shown that for a complex dataset like ImageNet2012, our second Local Hints Network Hint2 outperforms all the rest of the models based on various evaluation metrics. This shows that including the local hints in both the generator loss and the input would be a good idea for model design. Apart from the colorization of images, we have also extended the application of our Local Hints Network Hint2 for video colorization. Although the current results need to be strengthened, we think our second model Hint2 is a possible solution to video colorization because the Hint2 model has higher efficiency compared to the basic CGAN, and it generates the best quality images according to our evaluations. From this experiment, it is also proposed that a recurrent framework, which can guide the selection of local hints, would be meaningful for enhance the quality of video colorization and we believe that a research into this topic would be interesting future work.

References

[1] Zhu J.Y. Isola P. Geng X. Lin A.S. Yu T. Zhang, R. and A.A. Efros. Real-time user-guided image colorization with learned deep priors. 2017.

[2] Eric Ng Kamyar Nazeri and Mehran Ebrahimi. Image colorization using generative adversarial networks. 2018.

[3] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. 2014.

[4] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. pages 689–694, 2004.

[5] Xujie Li, Hanli Zhao, Guizhi Nie, and Hui Huang. Image recoloring using geodesic distance based color harmonization. *Computational Visual Media*, 1(2):143–155, 2015.

[6] Kun Xu, Yong Li, Tao Ju, Shi-Min Hu, and Tian-Qiang Liu. Efficient affinity-based edit propagation using kd tree. *ACM Transactions on Graphics (TOG)*, 28(5):1–6, 2009.

[7] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Trans. Graph.*, 34(4):139–1, 2015.

[8] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. pages 649–666, 2016.

[9] Guillaume Charpiat, Matthias Hofmann, and Bernhard Schölkopf. Automatic image colorization via multimodal predictions. pages 126–139, 2008.

[10] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. pages 327–340, 2001.

[11] Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics (TOG)*, 35(2):1–15, 2016.

[12] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. pages 2414–2423, 2016.

[13] Jun-Yan Zhu, Philipp Krahenbuhl, Eli Shechtman, and Alexei A Efros. Learning a discriminative model for the perception of realism in composite images. pages 3943–3951, 2015.

[14] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Ian Goodfellow, Jean Pouget-Abadie and Yoshua Bengio. Generative adversarial nets. pages 2672–2680, 2014.

[15] E. Shechtman J.Y. Zhu, P. Krahenb uhl and A. A. Efros. Generative visual manipulation on the natural image manifold. 2016.

[16] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. 2016.

[17] F. Huszar J. Caballero A. Aitken A. Tejani J. Totz Z. Wang C. Ledig, L. Theis and W. Shi. . Photo-realistic single image super-resolution using a generative adversarial network. 2017.

[18] Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros. Image-to-image translation with conditional adversarial networks. 2018.

[19] Mu-Heng Yang Qiwen Fu, Wei-Ting Hsu. Colorization using convnet and gan. 2017.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

A Appendix

A.1 Previous Model: Automatic Colorization using CNN

In this study, we first run the fully automatic approach using CNN for image colorization. The dataset used in this part is ImageNet2012. As is shown in Fig. A1, although the algorithm is quite simple, the colorization results can be of high-quality.

Figure A1: Example of gray-scale images and auto colorization outputs using the CNN algorithm

A.2 Previous Model: Local Hints Networks

With local hints, we also run the previous model that can implement colorization using the gray-scale images and the hints. The dataset used in this part is also ImageNet2012. We tested on several images with different user input. As shown in Fig. A2

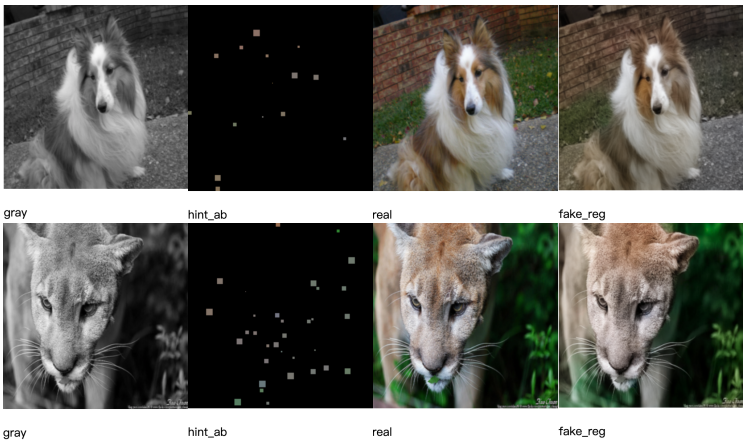

Figure A2: Example of colorization with local hints outputs using the CNN algorithm

A.3 Colorization with Global Hints

Instead of the automatic approach using CNN, in this part, we also train the deep network to colorize, given the gray-scale version and the hint histogram from the global statics of the user inputs. The dataset used in this part is ImageNet2012. As is shown in Fig. A3, a set of examples is shown. The gray-scale image of a car on the left is the input and the three images in the first row serve as the global hints. Clearly, we can find that the three outputs shown are following their hints.

Figure A3: Example of colorization with different global hints

A.4 Video Colorization

The supplementary movie (Movie A4) is enclosed to show the results from our video colorization part. As is discussed in Sec. 5.4, this experiment is designed to enhance the novelty and explore the future applications of our present works.