Coloring of Grayscale Images using Generative Adversarial Network

Aayushi Pandey[1], Abhay Singh Rana[2]
[1,2] *Student, Raj Kumar Goel Institute of Technology/AKTU*

Abstract **- Attempts to colorize grayscale photographs into natural-looking colorful images have been made in the past. The concept of automatic image colorization has piqued attention in the recent decade for a variety of applications, including the restoration of aged or deteriorated photos. There have been various time and energy consuming traditional approaches. Throughout all the research, various deep learning approaches have evolved as a development in the world of technology. The most important reason to colorize a picture is to give it a unique and genuine appearance. In this research, we compare how colorization was handled previously to how deep learning is tackling it now. Many attempts are made linear to get the simplest objectives. In this research, we compare how colorization was handled previously to how deep learning is tackling it now. Many attempts are made in the linear manner to achieve the simplest results, which range from choosing colors from one end to the other, deciding the color palette, and a variety of other metrics. Deep learning user-guided and non-guided approaches have been established as technology has advanced over the last 20 years. This document compares and contrasts all the methods, as well as their benefits and drawbacks. We're working on an image colorization technique that's completely automated. In this work, we compare how colorization was handled previously to how it is being handled by deep learning. To achieve the best results, many efforts have been made in the linear way, which ranges from selecting colors from one end to the other, deciding the color palette and many other such metrics. With the evolution of technology over these 20 years, deep learning user-guided and non-guided methods have been introduced.**

Index Terms **– CIELAB color space, CNN, Colorization, Deep learning, neural network**

I. INTRODUCTION

Black-and-white photographs from the past are deemed priceless and have extraordinary artistic merit. However, because color is a crucial component of visual representation, it is unthinkable to fully visualize the scene only by looking at them. The viewer's attitude changes dramatically when black-and-white photos are colorized. As the past and present temporal differences close, in return it depicts the image as more realistic and acceptable. Nevertheless, accurate color information for early pictures is sometimes absent, making reconstruction problematic. Nonetheless, the goal of colorization is to trick the spectator, convincing him that the colorized image is genuine, and not to rebuild the color precisely.

II. LITERATURE REVIEW

The process of adding color to a monochrome image has a unique and strongly rewarding effect. Black-and-white photographs from the past are considered priceless and of outstanding creative merit. However, because color is such a crucial component of visual representation, it is not possible to properly envision the actual truth just by looking at them. The viewer's perspective is drastically altered when black-and-white photographs are colorized, differences are removed between the past and present, constructing the scene more realistic. However, understanding the true hues of early images is often difficult to come by, making precise reconstruction challenging. Nonetheless, the aim of adding colors is to hoax the viewer into believing in the authenticity of the generated image which will be monochromatic, instead of the reconstruction of the image with accurate colors. Regeneration of ancient monochromatic images, film restoration and are few of the main features of colorization

Automated machine learning, particularly deep learning approaches, have risen to prominence as a result of technological advancements. These methods have been proven to work in many of the image processing applications [1]. Image classification, face detection, handwriting classification, image

resolution, image blending, etc. [2], [3] are few of the different application domains where deep learning has showcased effectiveness, also showcasing further advancements are possible in upcoming time . Large volumes of dataset are easily handled by machine learning and deep learning while detecting hidden patterns and delivering estimations of latent knowledge. Defining a group of rules within the dataset via detecting and uncovering patterns with some prior knowledge is called machine learning, while deep learning is used to discover regularities by employing artificial neural networks. By this method it ensures to achieve colorization advantages and makes it possible.

III. RELATED WORK

Image colorization techniques were different from what we use now before the advent of artificial intelligence procedures like deep learning. Two methods were used: the doodle method and the example-based method. Example-based approaches' outcomes could be obtained both automatically and manually. These methods of obtaining output, on the other hand, have become outmoded. Deep learning has shown to be more successful and efficient than the traditional example-based categorization on considerably bigger datasets.

Following some investigation, the most current ones divided the existing approaches into two categories: guidance and no-guidance. There is more room for error correction and improvement in the findings when users are involved. This user interference was crucial in selecting appropriate reference photographs.

IV. METHODOLOGY

A. User-guided methods

Scribble-based method

The scribble-based method may be traced all the way back to the first attempts at image colorization. Because manual picture segmentation processes were also used, this was an extremely time-consuming process. Many artists suffered as a result of colorization attempts that did not match the true artistic aesthetic of the images. As a result, this strategy was abandoned.

User-created scribbles are placed on certain sections of photographs in this manner. As stated by a framework, the color from the scribble by the user spreads throughout the image to the edges determined by the magnitude. Because we can scrawl in only a few regions of the image, we can add more colors strategically as we see fit.

Because color propagation is limited by intensity values, it eliminates the need for explicit segmentation. We don't even need to look for photographs that aren't accessible.

On the other hand, this method is time-consuming because we must color each and every boundary and color differentiation. It's also crucial to use the right color palette to bring out the image's true beauty. To be able to discern colors, we must first have a suitable source image. In addition, a large number of scribbles are required to replicate the image to our specifications.

Example-based methods

The color information in the source color picture is converted into comparable regions in the destination grayscale image in this process. This helps reduce human interference, but does not completely eliminate it.

Improvements in the internet search engine, as well as the introduction of consolidate and indexed image databases, aided the transfer of a common "context" across photographs, according to Welsh et al. [4]. We need to detect the pixel with that brightness value in the target and reference images using a texture identification similar to the statistical information for surrounding pixels.

The methods provided have been examined, refined, integrated, and concatenated since 2005. Irony et al. [5] apply image segmentation for more effective color assignment. The k-nearest neighbors (KNN) approach is used to create an appropriate feature space for region differentiation. Liu et al analyzed the effects of shadows and light reflections, and changes in lighting conditions. [6]

Since 2010, qualitative and quantitative comparisons of staining approaches have attracted more attention. Chia et al. [7] and Gupta et al.[8] investigate local features and their consequences on color rendering. Pixel groups, such as patches and superpixels, are used for functional studies to take advantage of spatial coherence.

Some of the main limitations are that there may not be a single suitable reference image to verify precise color rendering. Objects in the scene should be similar in both the target and reference images.

As an advantage, the example-based method is marked by its property of ease of use and speed.

B. Deep Learning Methods

Plain Colorization Neural Networks

The architecture of simple colorization neural networks is typically simple: stacked layers simply or may skip connections [9], [10]. This category includes the best known fully automated coloring method that was offered by Zhang et al. [9]. This method is based on the polynomial pixel color classification and the rebalancing of the classes to increase the diversity of the generated colors. For each pixel, possible color distributions are projected. Pixels are classified according to chances of belonging to one of the 313 sampling and quantization plane segments of the CIELAB color space.

It provides a wide range of powerful results. However, small neural networks can have problems reliably capturing color properties in a variety of scenarios with different color styles [11]. Results often include false colors and visible artifacts [2]. It is assumed that using an ensemble of neural networks gives better results than using a single network.

User-Guided Colorization Neural Networks

It necessitates participation by users and is based on deep learning principles. The user contribution is significant since, due to the extensive use of deep neural networks, they are often lumped in with deep learning approaches. User participation in various form like strikes, scribbles [2], [12], or a textual phrase [13], [14], is required for neural network input.

It manages colorization with textual input. This colorization subcategory necessitates language processing and certain textual terms from a limited dictionary. Since the instructions by language and scene area are related, you can reuse the same instructions to color the same elements in different photos. This subcategory's interactivity makes it ideal for children's reading education.

Diverse Colorization Neural Networks

Any design that generates more than one output image with colors that aren't always the same as the original is called a varied colorization neural network[15], [16], [17], [18]. The term "diverse" here is used to describe that results of the colorization methods always differ from the actual images and can also be said to be ground truth yet are considered realistic. GANs are typically used to achieve a wide range of effects. Vitoria et al. devised the GAN architecture.

Color information is generated by the two-part generator, which also identifies cognitive content. The discriminator is supposed to learn and tell the difference between actual and hoax data. The colorization method employs the CIELAB color space. The model is trained using a fully self-supervised technique, which produces high-quality, vivid outputs.

V .GENERATIVE ADVERSARIAL NETWORKS (GANS)

Generative adversarial networks were introduced by Goodfellow et al in 2014, as a new kind of generative model (GANs). The generator and discriminator are the two smaller networks that make up a GAN. As per the name, the job of the generator is to produce outputs that are same or almost similar as the real data. The job of the discriminator is to differentiate if a picture vector or illustration is from the original data distribution or from the generator's model distribution. The generator and discriminator are simultaneously trained until the generator can provide a reliable output. The structures of both the smaller networks (i.e. The generator and discriminator) are based on the architecture of a multilayer perceptron model. The generator and discriminator are both CNNs, because colorization is a type of image translation problem. The generator is represented by the mapping G(z; G), in which z is a uniform distribution of noise variable that represents the input to the generator. The discriminator is represented in a similar way by the mapping D(x; D), which produces a scalar between 0 and 1, in which x is the image color. The output that was produced by the discriminator can be interpreted and seen as the probability that the input came from the training data. The optimization problem for training the subnetworks ((i.e. The generator and discriminator) is specified by these D and G structures: G is taught to lessen the chance of the discriminator making the right prediction in generating data, while D is trained to improve the probability of the correct label being assigned. Mathematically, it can be stated as:

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

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

This function provides the cost functions that are needed to train a GAN are provided by the above equations. The two cost functions are often defined as a single minimax game issue along with value function V (G, D):

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

In the model that we have built, we used a different cost function in the generator. The discriminator's cost function in equation 1 is defined by minimizing the probability that it is correct. However, the two problems with this strategy are: 1) The generator will have a near-zero gradient if the discriminator has a good performance during the training stages in back propagation. There will a delay in the convergence rate because the generator will continue to provide comparable outputs while it is being trained. 2) The initial cost function can be defined as a strictly decreasing or declining, infinite function. The cost function will diverge in the minimization phase. The possibility of the discriminator being incorrect or inaccurate has increased since we have changed the cost function of the generator rather than minimize the likelihood of the discriminator being accurate in order to address the mentioned issues. It was proposed by Goodfellow at the NIPS 2016 Tutorial [19] that the new cost function as a heuristic, non-saturating game , and it is written as:

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

And can also be defined as the minimization problem.

The figure below represents the comparison between the cost functions in equations 1 and 4 through the blue and red curves. Also, the cost function has been further changed by using the 1 -norm in the regularization term [20].

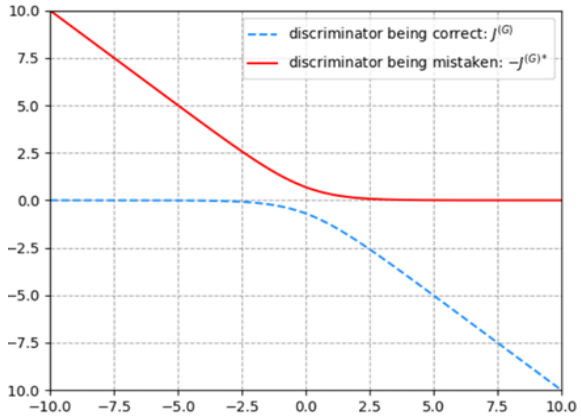

Figure 1 Comparison of cost functions J(G) (dashed blue) and −J (G) ∗ (red)

This forces the generator to provide pictures that are comparable to the ground truth. This should retain the original image's form and hinder the generator from "fooling" the discriminator by assigning random colors to pixels. The cost function looks like this: where y represents the ground truth color labels and is a regularization parameter.

A. CONDITIONAL GAN

The generator's input is randomly generated noise data z in a typical GAN. However, because grayscale photos, rather than noise, are used as inputs to our problem, thus it is not a suitable approach to automated colorization tasks. This problem was solved through conditional generative adversarial networks [22] Because there is no noise injected, the generator's input is interpreted as zero noise with the grayscale input as a prior, or G(0z|x) formally. The discriminator's input has been changed to accommodate for the conditional network, Our final cost functions as a result of these modifications are::

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

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

Both of the colored images generated by discriminator and raw black and white image are collected by discriminator, and it involves in finding the true colored picture.

VI. BASIC REQUIREMENTS OF THE SYSTEM

A. DATASET

One of the most important phases in every GAN-based project is to find an adequate dataset. The data collection we use can have a big impact on what our model learns. Fortunately, we may use any existing picture dataset including RGB photos for our image colorization work, or we can quickly obtain a considerable amount of data. However, to achieve better results, that dataset needs to be broadened. Tone (theme) or lighting conditions in RGB photographs may provide a lot of information that isn't present in grayscale images. We'll need a lot of verities in this dataset to make an effective model. For the time being, we are not using real-world images for the sake of simplicity. We're getting our training data from some well-known animated films, TV shows,

and YouTube videos. To capture this data, we created a CV2 script that converts films into photos and reduces superfluous data by removing comparable neighboring images.

B. PRE-PROCESSING AND CLEANING

While converting videos to photos, we removed redundant data, thus our dataset is already clean. We'll aim to supplement our data in this stage so that our model doesn't overfit. Create a pipeline that takes all the picture paths as input and outputs RGB images that are 512*512(px) in size, together with their grayscale input images. We utilized random crop, a TensorFlow image processing function, to shrink the image size. This function returns a randomly picked portion of the input photos of a certain size. Since then, we've gathered images in "Full HD" and "4k" resolutions, respectively 1920x1080(px) and 3840x2160(px). Any of these resolutions can create nearly infinite unique images of 512x512(px) and make sure that the model will barely train on the same segment of any image twice.

C. MODEL BUILDING

We modify our generator and discriminator architectures for this project. Convolution-BatchNorm-ReLu modules are employed by the generator as well as discriminator. Phillip Isola and colleagues describe the architecture in their paper Image-to-Image Translation using Conditional Adversarial Networks. The Image-to-Image Translation model is trained through the data which was acquired in phase one.

Using the architecture presented in Figure 2, we train a Convolutional Neural Network(CNN) to map from a grayscale input, i.e. black and white image to an RGB or colored image. The model is freely available, and details of its architecture are shared in the additional materials on our project GitHub page. The following sections focus on the goal function's construction and our method for deducing color point estimates from the predicted color distribution.

D. ARCHITECTURE

The architectures consist of generator and discriminator. Convolution-BatchNorm-ReLu [23] modules are engaged by the generator and discriminator The architecture is described in detail in the online additional materials, with significant aspects discussed below.

Generator with skips

Problems with image-to-image translation are classified as the fact that they transmit it into the input grid of high clarity, outgoing grid with high resolution. The appearance of the input and output of the problems look different too, still both are a translation of the same basic structure. As a result, the input structure is adjacent to the output structure. The generator architecture is based on these concepts. An encoder-decoder network [30] has

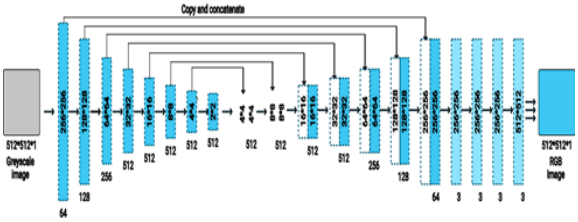

Figure 2 Architecture of the "U-Net" model

been employed in several earlier solutions [24, 25, 26, 27, 28] to difficulties in this field. A series of layer is taken, and the input data is passed through that to progressively downsample the data until a bottleneck layer is reached when the process is reversed. The entire flow of information must flow through all the layers, along with the bottleneck in these networks. There is too much minor information shared between the output and input grid that has high resolution, and it exists for many image translation difficulties. For e.g., in image colorization, the input and output have a common arrangement of eminent edges. By adding a skip connection that follows the usual "UNet" structure [29], the generator can bypass bottlenecks for information like this. A skip connection is placed between layer I and layer n, where n equals the number of total layers, specifically . 3.2.2 Markovian discriminator simply merges every channels present in layer I along with all channels of layer n. Loss of L2 (see Fig. 4) and L1 leads to fuzzy and blurred outputs [32]. They may not contribute to high frequency intelligibility, but captures the low frequencies accurately in many conditions.

If this is the case, an entirely new system is not required for the model to provide low frequency accuracy. L1 is sufficient. This motivates the use of the L1 term to ensure low-frequency accuracy by limiting the GAN discriminator. It then only models high-frequency structures. (Eqn 4). The local fragments of the image are paid enough attention in order to represent the high frequencies. As a result,

we develop a discriminant architecture called PatchGAN that penalizes the structure using only the scale of patches. An attempt is made by the discriminator to know if the NN labels in the picture are fake or genuine. The discriminator is convolutionally run on the images, averaging all responses to get a final result of D. This section shows that the overall image size can be much bigger than N and still give high quality output. This is user more often because there are fewer parameters in the smaller PatchGAN, is a lot faster and can be used on images with huge size. The image can successfully be modelled as a Markov random field by the discriminator, thinking that the independence that exists in the middle of pixels is separated by something that is more than the patch diameter. So our PatchGAN can be defined as the loss of style and structure.

Patch GAN - Markovian discriminator

It is notable that the losses, L01 and L02, produce incorrect and blurred outcomes while generating images [30]. Although the high recurring crispness are ignored by the losses, they precisely catch the lower frequencies most of the time. For issues where this is the situation, we do not need to bother about a total new structure to uphold the lower frequencies correctly. L01 will do it. This limits the GAN discriminator to model the high frequency structure, depending on a L01 expression that drives the accuracy of the lower recurrence. It is sufficient to limit attention to the design in the near region of the image for the higher frequencies. Conclusively, we envision a discriminator architecture called PatchGAN. This architecture penalizes the architecture only by patch size.

To check if all the N*N adjustments are on target, the discriminator groups on all the off chances.

Optimization and inference

For network optimization a standard approach is followed [31] : We toggle between a gradient descent step in G and a gradient descent step in D. As provided in the first research paper on GAN, as opposed to training D, we limit log G(x, D(x, z)) [31] and train to maximize log(1 − G(x, D(x, z)). Also, when optimizing G, dividing the target by 2 reduces the learning rate of G compared to D. SGD is used for the mini-batch and Adam solver [42] is applied with a rate of learning of 0.0002. . , and the energy boundary β1 = 0.5,

β2 = 0.999. During withdrawal, start the generator network in the same way as in the preparation phase.

This protocol differs from the normal protocol in a way that it applies dropouts during testing and relies on normalization of the package test application [24] using

Figure 3 Generated dataset

test package data instead of to remove sizes somewhere in the range from 1 to 10. Different. Total insights from your training game. This method of handling cluster normalization is known as "instance normalization" if the value of batch size is equal to 1, and has proven convincing in image generation problems [32]. The test uses a b batch size of from 1 to 10, depending on the experiment.

Training Data

We started with a collection of YouTube videos and Movies with different genres (animated, action, frictional etc.) for our dataset. Then we created a custom python script to break down the videos into frames. We created a function to reduce redundancy by eliminating similar frames based on scores calculated by Frame Subtraction Method [-]. The remaining images are converted to grayscale which acts as input data and their colored versions are referred to as target data.

E. WEB APPLICATION

Create and Deploy APIs

Our model is going to be computationally heavy. So we cannot run it on the client site. To generate the colored image, we will create a flask API through which input i.e. raw picture will be provided, and the result will be a converted picture which will be a colorized image and will return it to the web application.

Creating the Web App

The purpose of a web app is to give an easy interface to navigate around the project. Our project is an

integrated solution of AI and Web development. The main motive to create a web app as an initial step is to increase its availability. As the first step, we have decided to make a website offering the features we provide. This will be in the beta testing phase as long as the bugs and features are fixed. We will be releasing the first version as pilot testing and will be maintaining it with continuous patch updates.

As a future potential project, we plan to make it an android and iOS compatible application which will serve the purpose of easy access. The website will continue to provide all the features for free. Until then, the web app will be responsive for different screen sizes and will be convenient to use on any device.

The application website will inculcate the use of MERN stack. That is, the website's front end will be handled using React.js and Tailwind CSS. The reason for choosing React as our front-end technology is the mere light-weightedness and easy accessibility. React comes in with many free features, which makes it easy to scale the website later and make changes without worrying about inconsistencies in the web app. This also makes our web app render and load faster than the traditional bootstrap framework, which takes a lot of time in loading components that are not really required.

As per the backend of our website, we will be sending pictures uploaded by the user as a JSON object to contact the backend services. This will be done using APIs. Since the user data is not very large and can be stored in the local storage, we will refrain from using public databases. The image is previewable and downloadable at the click of a button.

Once you visit the URL of "Colorizing Images", you will be presented by the homepage of this project. The homepage consists of a navigation bar that consists of basic menu options. These menu options will give you options to select from. You can read about the members, the project, instructions, and the final working project page.

The "instructions" section consists of the guide to use our project. Right from uploading an image to applying different filters to it, each and every piece of information will be provided. The main "project" section will consist of an option to upload your image that is to be converted to a colored picture from grayscale. You will also be provided a dropdown menu that will have other options to choose from. These options will provide different filters which you can apply to your picture.

As soon as the image is processed and the chosen filter is applied to your choice of picture, a side-by-side image collage will be generated. In this, on one side you can see the original image, that is the image that you uploaded and on the other side you will see the filtered image, i.e., the resulting image generated through colorization. This way, you can compare the accuracy of the generated image.

Conversion of grayscale images to colored images is our primary goal, but we also provide other filters.

The whole website is very intuition-based and uncomplicated to navigate. Hence, with the instructions and guide, users will find it easy to use the website without any specific technical skills.

VII. CONCLUSION

We have learned that image colorization is a huge and iterative task if done manually. In this study, we have shown that we can colorize multiple images at once using deep learning methods such as GAN. The colorization may not be pixel perfect, but is close to reality. Our method is useful to change color themes in photos and movies. It can colorize empty sketches with real world colors. It can turn grayscale and monochrome pictures into colored versions to match the reality. Some results did have an unsatisfying result, but our method outperforms some state-of-the-art methods in visual quality.

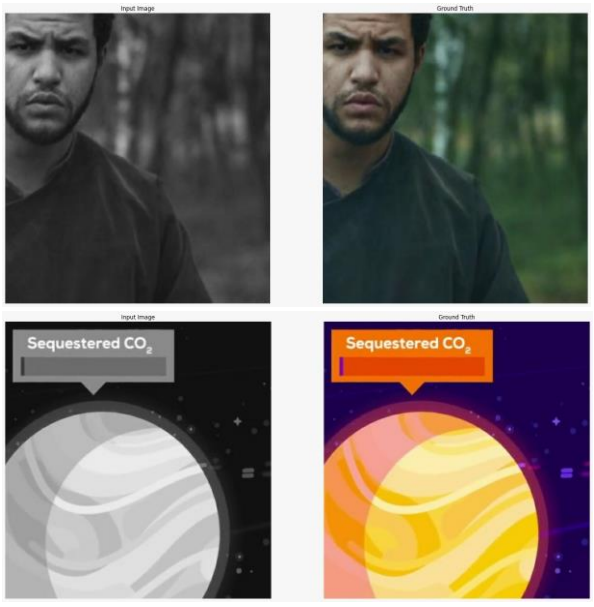

Figure 4 Comparison between grayscale image and generated image

VIII. REFERENCES

[1]https://medium.com/@ODSC/wonders-in-image-processing-with-machine-learning9c6f2e070e99

[2] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, et al., "Real-time user-guided image colorization with learned deep priors", ACM Trans. Graph., vol. 36, no. 4, pp. 1-11, Jul. 2017.

[3]https://eprints.sztaki.hu/9292/1/Varga_1_3306455_ny.pdf

[4] T. Welsh, M. Ashikhmin and K. Mueller, "Transferring color to greyscale images", ACM Trans. Graph., vol. 21, no. 3, pp. 277-280, 2002.

[5] R. Irony, D. Cohen-Or and D. Lischinski, "Colorization by example", Proc. Eurographics Symp. Rendering, pp. 201-210, 2005.

[6] X. Liu, L. Wan, Y. Qu, T. T. Wong, S. Lin, S. C. Leung, et al., "Intrinsic colorization", ACM Trans. Graph., vol. 27, no. 5, pp. 152-1-152-9, 2008.

[7] A. Y.-S. Chia, S. Zhuo, R. K. Gupta, Y.-W. Tai, S.-Y. Cho, P. Tan, et al., "Semantic colorization with internet images", ACM Trans. Graph., vol. 30, no. 6, pp. 1-8, Dec. 2011.

[8] R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng and H. Zhiyong, "Image colorization using similar images", Proc. 20th ACM Int. Conf. Multimedia (MM), pp. 369-378, 2012.

[9] R. Zhang, P. Isola and A. A. Efros, "Colorful image colorization", Proc. Eur. Conf. Comput. Vis., pp. 649-666, 2016.

[10] Z. Cheng, Q. Yang and B. Sheng, "Deep colorization", Proc. IEEE Int. Conf. Comput. Vis. (ICCV), pp. 415-423, Dec. 2015.

[11] Z. Cheng, Q. Yang and B. Sheng, "Colorization using neural network ensemble", IEEE Trans. Image Process., vol. 26, no. 11, pp. 5491-5505, Nov. 2017.

[12] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, et al., "TextureGAN: Controlling deep image synthesis with texture patches", Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 8456-8465, Jun. 2018.

[13] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, et al., "TextureGAN: Controlling deep image synthesis with texture patches", Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 8456-8465, Jun. 2018.

[14] C. Zou, H. Mo, C. Gao, R. Du and H. Fu, "Language-based colorization of scene sketches", ACM Trans. Graph., vol. 38, no. 6, pp. 233:1-233:16, 2019.

[15] P. Vitoria, L. Raad and C. Ballester, "ChromaGAN: Adversarial picture colorization with semantic class distribution", Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), pp. 2445- 2454, Mar. 2020.

[16] Y. Cao, Z. Zhou, W. Zhang and Y. Yu, "Unsupervised diverse colorization via generative adversarial networks", Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases, pp. 151-166, 2017.

[17] A. Deshpande, J. Lu, M.-C. Yeh, M. J. Chong and D. Forsyth, "Learning diverse image colorization", Proc. IEEE Conf. Com

put. Vis. Pattern Recognit. (CVPR), pp. 2877-2885, Jul. 2017.

[18] S. Yoo, H. Bahng, S. Chung, J. Lee, J. Chang and J. Choo, "Coloring with limited data: Fewshot colorization via memory augmented networks", Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 11275-11284, Jun. 2019

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[20.] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. 2016.

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. 2016.

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

[23] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In ICLR, 2016

[24] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015

[25] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In CVPR, 2016.

[26] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In ECCV, 2016.

[27] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In ECCV, 2016.

[28] Y. Zhou and T. L. Berg. Learning temporal

transformations from time-lapse videos. In ECCV, 2016.
[29]] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixellevel domain transfer. ECCV, 2016
[30] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006
[31] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, 2015.
[32] A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In ICML, 2016.