# Analysis of Image Colorization Techniques ECE 6254 Final Project

1st Tara Poteat
*Electrical and Computer Engineering*
*Georgia Insitute of Technology*
Atlanta, USA
tpoteat3@gatech.edu

2nd Supriya Sundar
*Electrical and Computer Engineering*
*Georgia Institute of Technology*
Atlanta, USA
ssundar47@gatech.edu

3rd Rohan Banerjee
*Electrical and Computer Engineering*
*Georgia Institute of Technology*
Atlanta, USA
rbanerjee45@gatech.edu

***Abstract*—This body of work aims to study, analyze and present a conclusive comparative study of two machine learning techniques used in the field of image colorization. The implementations detailed in previous research will be studied to gain a comprehensive understanding of the algorithms used. Then, using the exact method detailed in the research, an attempt will be made to reproduce the results. The algorithms from past research will each be tested on the same set of test images to better understand how well the algorithms generalize to an out-of-sample dataset. The unique features of each algorithm will be analyzed to identify the modules associated with positive outcomes and the ones which cause it to fall short of target accuracy. Any areas where optimization could be possible will be identified and discussed. Additionally, any new or different approaches found in other papers will be highlighted.**

***Index Terms*—image colorization, self-supervised learning, deep learning, generative adversarial networks, convolutional neural networks, comparative study**

## I. Introduction

Image colorization is a technique that involves taking a grayscale image and adding color in a way that is plausible and realistic. The colored image produced using such techniques should ideally match the brightness, shade, and color of the image had it been a colored image to begin with. Vitoria P. et al, in their paper *"ChromaGAN: Adversarial Picture Colorization with Semantic Class Distribution"*, mention that "The colorization of grayscale images is an ill-posed problem, with multiple correct solutions [1]." Given a black and white image, even the human brain has difficulty imagining colors. For example, take a grayscale image of a car, outdoors, with the sky, grass, and tress in the background. A human would conclude with a high probability that the sky in the image might be blue and the foliage might be green. However, imagining the color of the car is not such an easy task for the human brain - it could be any given color. Red, blue, green and yellow would all be reasonable guesses. Often times in images, there are many possible color variations for objects, which makes this an interesting problem to study. Image colorization algorithms also have their own shortcomings, but they can produce colors that would be reasonable for us to expect. Some algorithms produce better vibrancy and some better accuracy. In order to compare these algorithms on their own merit, one must be familiar with the method and motive at the heart of the implementation. A comparative study of a body of past research, the techniques and algorithms used lend deeper insight into this problem with multiple correct solutions.

## II. Methodology

### A. Motivation

At the onset, we were familiar with image colorization being an active area of current AI and deep learning research. This would be an interesting problem to explore and gain more in depth knowledge about and implement ourselves. In the past, image colorization had been a difficult problem that would take hours of human effort, requiring human inputs at every stage along with lots of necessary hard-coding. Machine learning turns this into a much simpler problem. Not being too adept at the ambit of the topic at the beginning, we were unsure as to the exact techniques and methodology or the course of research that had been done in order to come up with an effective solution to this problem. Because of this we set out to explore multiple implementations and methodologies that had been utilized in the past.

### B. Algorithm Set

We started by researching the colorization problem and becoming more familiar with general work that had been done in the field. We then picked out a volume of research that described a set of deep learning implementations which seemed replicable with the resources we had in hand. These implementations would ideally be similar in terms of a common data set for training but in with varying approaches to the colorization problem in terms of the core algorithm and neural network architectures used. This would give us a colorized result set based on multiple unique implementations, which we could then compare and qualitatively and quantitatively validate. With this in mind, we found two unique implementations of image colorization that could be compared against each other in order to further our understanding of colorization using AI. The first implementation [2] uses a modified Convolutional Neural Network with a weighted loss function for color rebalancing to "*hallucinate*" plausible colors given an input black-and-white image. The second implementation [4] we looked at, similar to the first one, utilizes a modified Convolutional Neural Network. It, however, has a hidden "*fusion*" layer

that merges information from local image pixels dependent on small image patches with global priors computed across the entire image to achieve more accurate colorization.

### C. Insights

In most of the colorization attempts we found online, the training is not done on RGB images, but rather, on the Lab color space. The CIELAB color space, also referred to as L*a*b*, is a color space defined by the International Commission on Illumination (abbreviated CIE) in 1976. It is a standard where the initials stand for L* - Lightness, a* - Red/Green Value, b* - Blue/Yellow Value. This 3D space, used to denote colors, is commonly used and especially popular in computer vision applications. It makes intuitive sense as to why this is; the colorization task becomes simpler. If one were to use the RGB color space, the algorithm would receive a black-and-white image, and would have to predict the red, green, and blue values of each pixel. All of this would have to be based on the input image's black and white contrast and shadowing. This task involves prediction of 3 values from a single input value, a 3D prediction from a 1D input. This creates a harder prediction with more room for error. For training in L*a*b*, the input image is first converted from RGB to L*a*b*. The L component of the input image, which represents the lightness of the image, is the input to the network. This is the black and white factor and from here, the algorithm only has to predict the a and b values. Through this method, a 3D prediction for the RGB case can be constrained to a 2D prediction for the L*a*b* space. This proves to be an easier task and would not be as severely under-constrained as the RGB case. After the algorithm has predicted the a and b channel for the incoming L channel values, it superposes the two, resulting in the final Lab colored image. This image is converted back to the RGB space for the final output.

Another challenge that AI-based colorization needs to address is the ability to make the colored images as realistic and plausible as possible. The approach used in the "Colorful Image Colorization" paper uses what they call the "colorization Turing test" [2], where they have real human participants try and differentiate between real colored images and the same image colorized by their model. It claims that it manages to fool humans in 32% of the trials, which is significantly higher than other implementations. However, it also discussed the short-comings of the model when it comes to predicting colors for man-made objects, such as a car on the road. Given a black and white input image, the model could predict the car being of any number of colors, which is, understandably something that the human brain is likely to do in such a scenario, as well. The implementation explained by Isola [3] tries to quantify the accuracy of the prediction using a "discriminator" that assigns a number to the output image based on whether it thinks it is "*real*" or "*fake*". [4] does not quantitatively try and ascertain the accuracy of colorization but uses a human study to determine how *natural* the output image appears to the subject. It claims to have achieved a 92% **naturalness** based on their study.

As part of our study, we use Python scikit-image package in order to quantify how much of an error the same image incurs when put through the different models.

### D. Dataset

Both of the implementations discussed above have a trained model on the ImageNet dataset. Even though the second model was trained, originally, on Places, a scene recognition database from MIT, it does have a sub-implementation which was available for us which was trained on ImageNet. It is to be noted that the algorithms mentioned in the papers were trained over all 1.3 million available images in the ImageNet dataset. It is valuable that each of the models was trained on the same image dataset because that makes our test results more dependent on the algorithm itself rather than the training input. The algorithm and architecture are what we are interested in comparing. For testing, we feed in completely unseen black and white images to the networks in order to test the algorithms on absolutely out-of-sample data. This dataset is from Kaggle rather than ImageNet and contains over 700 black and white images along with the colored versions for testing error. Using these different test images would be more representative of how well these algorithms generalize in presence of unseen data, and therefore ascertain which algorithms gives us the most accurate results.

### E. A deeper dive

The colorization architectures used in "Colorful Image Colorization" [2] and "Let There Be Color!" [4] are based on CNNs but they have specific features which set them apart.

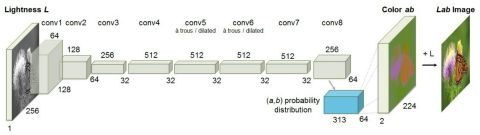


Fig. 1. Network Architecture in "Colorful Image Colorization" [2]

In the above architecture, each convolutional layer refers to a block of 2 or 3 repeated conv and ReLU layers followed by a BatchNorm [5] layer. There are no pooling layers for downsampling and resolution adjustments are done through spatial upsampling or downsampling between conv blocks. In this implementation, the error between the predicted and ground truth images were computed using the following equation:

$$L_2(\hat{Y}, Y) = \frac{1}{2}\sum_{h,w}\|Y_{h,w} - \hat{Y}_{h,w}\|_2^2$$

However, this was deemed inadequate as it resulted in unsaturated, bland colors. Following this observation, the authors quantized the ab color space and approached this as a multinomial classification problem with class rebalancing for the rarer colors as per the following equation:

$$L(\hat{\mathbf{Z}}, \mathbf{Z}) = \frac{1}{HW}\sum_{h,w} v(\mathbf{Z}_{h,w})\sum_{q}\mathbf{Z}_{h,w,q} log(\hat{\mathbf{Z}}_{h,w,q})$$

In the above equation, $v(\mathbf{Z}_{h,w})$ is the weight term that helps learn the rarer colors making the final output image much more vibrant with rich, saturated colors.

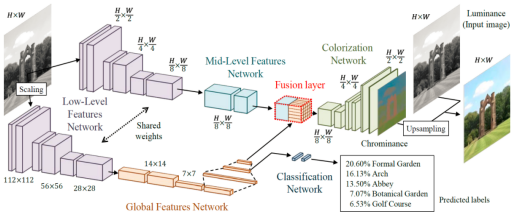


Fig. 2. Network Architecture in "Let There Be Color" [4]

There are standard neural network layers as well as convolutional layers as part of the architecture in the second implementation. The output layer consists of a convolutional layer with a Sigmoid transfer layer instead of a ReLU transfer layer, as conventionally used. Upsample layers consist of using the nearest neighbor approach to increase resolution of the output by a factor of 2. The interesting thing to note is the fusion layer in-between, which combines macro features trained on the whole image and takes into consideration features more local to an image patch so as to give higher accuracy.

To train the above network, the simple MSE formula is utilized. a*b* components are normalised to feed into the sigmoid transfer layer at the output. The model parameters are updated by back-propagation of the loss through the hidden layers. Hidden layers simultaneously train with the global features.

### F. Challenges

As per the initial plan, we set out to train the given models with completely new data sets and obtain results from a previously unseen test set in order to accurately qualify the models and compare them against each other. This proved to be a more daunting task than previously imagined by us. Data and package dependencies with the implementation in our first model from "Colorful Image Colorization" [2] hindered us, as the implementations were no longer supported by the authors. Training the second model, "Let There Be Color!" [4], seemed like an insurmountable problem from the beginning as the paper talks about the dataset requiring roughly 3 weeks on a single-core NVIDIA Tesla G80 GPU. This training corresponds to 11 epochs. The last implementation we found, "Image-to-Image Translation" [3], is something we would like to explore in the future. We briefly started training this model on Google Colab and managed to train for 18 epochs before we ran out of resources. However, the implementation recommended that we trained for at least 20 for suitable results. Because of these issues, we had to look for a workaround.

### G. Implementation

In order to compare algorithms with different colorization implementations, we found the source code for the "Colorful Image Colorization" [2] and "Let There Be Color!" [4] papers. We imported this code into a google colab notebook. To start we tried running the code and determined what dependencies we needed, approximated the time it would take to finish running, and tried setting up the commands to run the code in a way to run all the steps explained in the Readme for each.

The Colorful Image Colorization paper's github contained branches for a pretrained model and then a second one that could be trained using the users' own input images. After facing difficulties with the code for training the model, we decided to use the pretrained model. This pretrained model is trained on approximately 1.3 million images from ImageNet, which is more than what we would have been able to use given the timeline we had. The "Let There Be Color!" source code additionally contained a pretrained model that we could download and use. This model is trained on ImageNet as well. Again, this model is well trained and takes over 3 weeks to complete, so we decided to use the given model. This enables us to get the most accurate results for the architecture. Once we identified the two models we wanted to compare, we used the corresponding papers to walk though the process of obtaining the models and tested to make sure we could run our own chosen images through them.

### H. Error Comparison

To compare the models from the two papers, we first chose a test dataset that contained images that neither model had been trained on. The dataset we decided on was an image colorization dataset that contained jpeg images in both color and black and white versions. The test set contained 737 images. For each of the black and white images, we automated the process of feeding the images to the model and getting a colored image as the output. We then used the mean squared error and percent pixel difference to calculate the averages for each of the models using python scikit-image library functions.

## III. Results/Discussions/Findings

### A. Numeric Results

After running the model obtained from the "Colorful Image Colorization" paper, we found that the mean squared error was 68.83. In order to get the mean squared error, we first found the mean squared error for each test image by taking the difference in pixels between the original color image and the predicted color image produced by the model and squaring them and dividing by the number of pixels. After this average was found, each image had a corresponding mean squared error value and so then another average was taken to get the final value for the overall mean squared error for the model. This same process was repeated on the same set of test images for the model from "Let There Be Color!". The mean squared error for this model was slightly better at 61.22. To gain a better understanding of what these numbers mean and how accurate the predicted colored images are, we also found the percent error for each of the images. These results aligned with the mean squared error values, as the Colorful Image Colorization model proved to have a slightly higher accuracy

compared to the Let There Be Color! Model. The color pixel percent error was 45.64 and 44.61 respectively.

Although it seems like these results are not super accurate based on the error numbers, the images themselves look like they are realistic from the human perspective. When viewing the results, one can see that the results from both of the models are similar to the original color image as shown in figure 1. However, as shown in figure 2, not all predicted images align exactly with the original. In this example, both of the predictions show a yellow car when the car was blue originally. However, the colors are realistic and plausible and anyone would be inclined to believe that either of the three images could be representing the real color of the car.

Fig. 3. Colorization of snow scene. The first image is the original and the two on the right are generated by the colorization models. All three scenes look very similar and individually seem realistic. It would be hard to tell which one was the original.

Fig. 4. Colorization of cars. The first image is the original and the two on the right are generated by the colorization models. Although the original is very different in color, each would look realistic.

### B. Discussion

This test image here provides a great example showing why image colorization is such a hard problem. Grayscale images are very hard to translate accurately into the exact original color because there is an inherent ambiguity as to what the actual color of the object is. A grayscale image only contains the L channel and does not have any of the a and b channel information. Since this can vary so drastically, colorization algorithms are only able to venture a realistic guess and cannot pinpoint what the actual colors could be. With parts of an image that are invariably mostly the same color, such as the sky and foliage or a banana and an orange, the models are able to predict accurately. In other scenarios, such as with man-objects like cars, the models will not necessarily be able to output the same color as the original image, but it can predict a realistic color for the object. As algorithms become more advanced, machine learning models will improve at this prediction. However, as of now, the best models can only venture realistic guesses and nothing above and beyond that. Overall, for the test images, the models we tested did prove to give us relatively accurate results. When manually choosing images and comparing them visually, the output images from both of the models and the original objects looked similar. To determine the cause for the difference we dived deep into the details of the implementations.

### C. Explanation

In the first implementation, "Colorful Image Colorization" [2], we were able to identify a step in the color re-balancing process where the rare colors were lent more weight to account for the fact that they are seen less often in images in the ImageNet dataset. This resulted in more vibrant colors at the output as seen in figures 3 and 4; the vividness of color is apparent in the images. This could be a possible reason for higher error because some colors could be exaggerated. In "Let There Be Color!" [4], by separating local and global pixel information, the model is able to customize the color output based on the image as a whole. The implementation lends perspective to the micro-features in the image based on the whole scene. This makes the second model perform marginally better and have lower error values as compared to the former. The model could predict the sunset scene in Fig 3., and hence, was able to subdue the colors. As for the cars, the first model performs better better the focus is on a single object and not much contextual information is required. Figures 5. and 6. quantify the error for the snow scene, the car and presents the overall average.

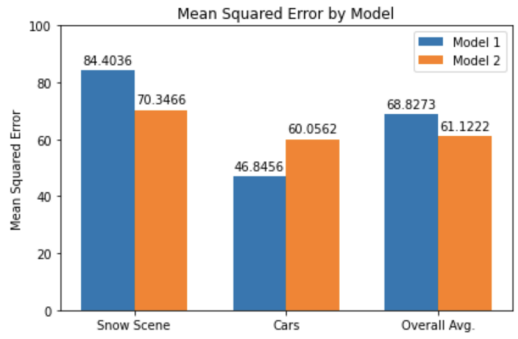


Fig. 5. Comparisons of mean squared error between the two models. For the snow scene, the second model had a lower MSE, for the cars, the first model had a lower MSE, and overall the second model had a lower MSE.

## IV. Conclusion and Future Work

Even though the models are based on a CNN at the core, unique design decision lend singular character to each model, accounting for different error numbers. One of the methods used a weighted loss function for color re-balancing to generate realistic colors. The other method uses a hidden fusion layer which merges local and global information about the pixel colorization to determine the color. Hence, the output image is colorized with the whole scene in mind.

A third approach that was found implemented colorization with a Conditional Adversarial Network[3], optimising an L1 loss, essentially making it a regression task, followed by the

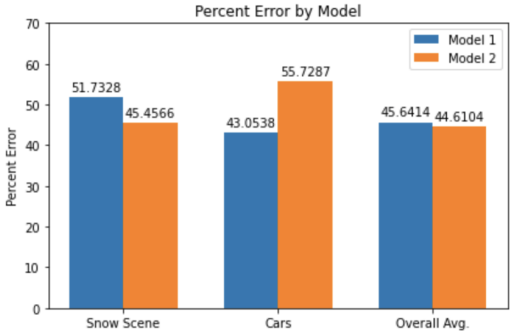


Fig. 6. Comparisons of percent error between the two models. For the snow scene, the second model had a lower percent error, for the cars, the first mdoel had a lower percent error, and overall the second model had a lower percent error.

GAN loss, which quantifies the output colored image as "*real*" or "*fake*" [3]. Incidentally, this bit of research was directly responsible for Pix2Pix, a Generative Adversarial Network designed for general purpose image-to-image translation.

Each model explored has its own merits and demerits. Compared to ground truth images, the output color images incurred an significant error margin. However, it is very apparent that each model can predict and colorize an image in such a way that it is completely plausible and these colored images are able to fool subjects most of the time. In case there is no ground truth image to begin with, such as B&W images from the past, and even B&W films made in the past [5], these models could perform quite effectively. Leaving the error incurred aside, the models could colorize an image realistically and could even be used to colorize old B&W films. They could ascribe plausible colors to a B&W image or video, helping bring the past to life. This has been demonstrated multiple times with images from the past that have been colored using any of these algorithms. This is quite an exciting proposition to consider and a lot of very exiting research to look forward to as colorization models continue to be improved upon!

## REFERENCES

[1] Patricia Vitoria, Lara Raad and Coloma Ballester, "*ChromaGAN: Adversarial Picture Colorization with Semantic Class Distribution*", 2020.
[2] Zhang, Richard and Isola, Phillip and Efros, Alexei A. "Colorful Image Colorization", 2016.
[3] Isola, Phillip and Zhu, Jun-Yan and Zhou, Tinghui and Efros, Alexei A, "*Image-to-Image Translation with Conditional Adversarial Networks*", 2017.
[4] Satoshi Iizuka and Edgar Simo-Serra and Hiroshi Ishikawa, "*Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification*", 2016.
[5] Zhang, Richard and Zhu, Jun-Yan and Isola, Phillip and Geng, Xinyang and Lin, Angela S and Yu, Tianhe and Efros, Alexei A, "*Real-Time User-Guided Image Colorization with Learned Deep Priors*", 2017.

## INDIVIDUAL CONTRIBUTIONS

The following is the list of tasks that were completed in order to accomplish this project and indicates who was responsible for the task The tasks accomplished through the project are listen below, along with the responsible individual.

1) Find research papers that focus on image colorization with unique implementations that could be analysed and compared. 2 papers were chosen for the reason that they had an available working implementation and source code that we could attempt to train and run test images on and ultimately be able to replicate the results ourselves. A third paper implementation was considered and is mentioned briefly in the discussion on future work. (Rohan, Supriya, Tara)
2) Gain a deeper understanding of the colorization implementations that were picked out from the set of numerous past implementations. This includes a brief of the deep learning network architecture of the models, usage of relevant datasets and the algorithm at the heart of each implementation. (Supriya, Rohan)
3) Find a new dataset to test both of the implementations. This dataset needed to contain images that were new and that the training models had not seen before. This would help validate the implementations as well as being indicative of the ability of each to generalize to out-of-sample data. Both of the models were trained on images from ImageNet and so the chosen dataset was an image colorization dataset from Kaggle. (Tara)
4) The first paper, Colorful Image Colorization, contained an implementation and a model that was pre-trained using the ImageNet dataset. Understand the colorization implementation so as to be able to use it for ourselves. Generate an output colored result dataset using the Kaggle dataset mentioned before. (Tara)
5) The second paper, Let There Be Color!, contained an implementation with a model pre-trained on ImageNet. Understand the colorization implementation so as to be able to use it for ourselves and replicate the results. Generate an output colored result dataset using the Kaggle dataset mentioned before. (Rohan)
6) Study the implementation of an alternate model that used cGAN that potentially performed better than the CNN implementations analysed before. This model proved much more challenging to implement given limited resources. However, we were able to gain a deeper understanding of different approaches possible to the colorization problem. (Supriya)
7) Determine how best to qualify the results from each colorization implementation. Finally selected Python scikit-image library and relevant library functions to compute MSE and loss. (Rohan)
8) Quantify the error that each result image incurs with respect to the ground truth colored image for each implementation. Calculate MSE and pixel percentage error for each pixel in each image in the 700+ image dataset and repeat the process for each implemented colorization algorithm. Obtain error numbers per image and over the whole dataset. This is to quantitatively determine which model performs best. (Tara)
9) Final Report. (Rohan, Tara)
10) Final Presentation. (Supriya)