CSCI 1430 Final Project Report:
Color Me This

Team: Nada Benabla (nbenabla), Yipu Gao (bgao9), Ethan Williams (ewilli51)
TA: Joel Manasseh
Brown University

Abstract

Given a grayscale image as input, we attempt to provide an estimation of the image's RGB colors, with the help of user-guided color hints.

We implement automatic colorization using a CNN encoder-decoder architecture combined with a pre-trained Inception-ResNet-V2 model for feature extraction. The automatic colorization is then used to generate a palette of five suggested colors via K-means clustering. The user can then input color hints onto the grayscale image, which are fed into a U-Net architecture to generate a colorized output in real-time in a user interface that we provide. The output of user-hint model is conditioned on an arbitrary number of hints provided.

1. Introduction

Color plays an important role in conveying information. As such, the aim of colorization can range anywhere from making an image more visually appealing, offering a new perspective, giving a glimpse of history, or perhaps simply allowing your grandparents to go on a vivid trip down memory lane.

For this purpose, we have chosen to address a subset of image colorization—flower colorization. In flower colorization, the color channels of a grayscale image of a flower are predicted. By tackling flower colorization, we intend to deliver transferable results with the limited training resources available to us. The hope is that the colorization network can learn basic patterns: that grass is green, the sky is blue, etc. while respecting the multi-modal nature of flower color. Additionally, a model that excels at colorizing flowers may be moderately transferable to other domains by recognizing features shared across all natural images; namely object boundaries and color gradients.

In order to address the above-mentioned multi-modal nature of flower colors, we've chosen to integrate user hints. We condition our output images on points and corresponding colors that users provide. The hope is for the model to propagate the user hints to color entire objects, making it convenient to color general natural images.

Because of our integration of user hints, images with color that's already known can still be converted to grayscale and used as an input with our model. This enables images to be recolored; a task that's normally very time-consuming. Users can provide sparse color hints that do not reflect the ground truth image and still end up with creative colorizations.

2. Related Work

Our approach for user-guided colorization is heavily inspired by [8]. This group tackles general image colorization, while we focus narrowly on flowers. In their model, they have a user hints model, a global hints model, and a palette selection model. Their palette selection model is per-pixel, while we provide a palette for the entire image. Our user hints model is significantly similar to theirs. We have not included a global hints model. The most significant difference between our user hints model and their user hints model is that the specifics of our model architecture are significantly different—we use a different technique to downsample, upsample, and integrate the residual connections. [8] uses subsampling and elementwise addition, while we use strided convolutions. In addition, we sample the user hints in the synthetic user hint generation from a Gaussian centered at the hint point, while [8] takes an unweighted average. In total, this makes our user hint model more sophisticated.

On the other hand, the model architecture for automatic colorization is based on the Deep Koalarization paper by Baldassarre et al. (2017) [1], as well as Wallner's article on Colorizing B&W Photos with Neural Networks [7]

3. Method

Our project is split into two parts: Automatic Colorization with CNN and Inception-ResNet-V2 and User-guided colorization with U-net.

3.1. Automatic Colorization

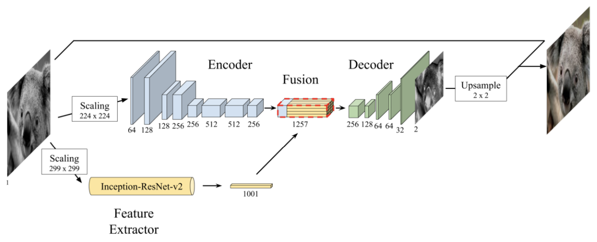


Figure 1. CNN with Inception-ResNet-V2. This architecture is owed to [1]

After preprocessing, a grayscale input image first runs through an encoder which consists of 8 convolutional layers. In parallel, the input image (resized to 299x299) also goes through a pre-trained Inception model (pre-trained on 1.2M images from Imagenet). The Inception model acts as a high-level feature extractor for the image content. We retrieve an embedding of the gray-scale image from the last layer and concatenate it with the encoder's output via the fusion layer. The output of the fusion layer is fed into a decoder model which consists of a succession of convolutional and upsampling layers as shown in figure 1. Each convolution layer uses a ReLu activation function, while the decoder's last layer uses a hyperbolic tangent activation function. We experimented with adding a few batch normalization layers, but this idea was dismissed as they hindered the overall performance during training.

The input images were represented in the CIE L*a*b color space. This choice, as outlined in [1], separates the color characteristics from the lightness component and has the benefit of increasing the level of detail in the output images.

To aid learning, data augmentation was implemented. These augmentations include shear transformations, zooming, rotations, horizontal flipping, width shifting, height shifting, and a nearest neighbor fill mode.

Because the results of automatic colorization are known to be rather desaturated and slightly faded, image enhancement was added to the output images.

Gamma correction [5] was used to adjust the brightness of the image. This was done by building a lookup table that maps each pixel value to its adjusted gamma value. In addition, the RGB colors were lightened by a coefficient of 20% before being displayed to the user, by multiplying the value of the image's red, green and blue channels by that coefficient.

Finally, K-means clustering was used to extract the top five most dominant colors in the output picture. These colors were used to populate a palette of suggested colors, meant to guide the user in choosing color hints.

Figure 2. Color palette with top 5 suggested colors

3.2. User hints model

After a user chooses colors from the automatically generated palette and specifies the spatial location, the remainder of the colorization task is carried out by the user hints model. The input to the user hints model is a grayscale image, a Python list of image coordinates and their associated list of RGB colors of the same length. The output of the user hints model is the predicted RGB image. The greyscale image is downsampled to a convenient size, chosen to be 224x224. The next task of the user hints model is to encode the hints into a format that can be processed by a convolutional neural network. Three channels with the same dimensions as the grayscale image are generated: a binary hint mask, and two color hint channels. The hint mask takes the value true within a radius (which is a tunable hyperparameter) of each of the hint points. The RGB colors are converted into the YCbCr format, and the two color channels (Cb and Cr) are taken. The color hints are plotted on each of the respective color hint channels. The two color hint channels have the value zero where the hint mask is false. The output of the model is the two predicted Cb and Cr channels, which are combined with the input grayscale, converted to RGB, and delivered to the user. The convolutional neural network is a purely convolutional U-net, inspired and tuned from an example used for image denoising [3].

The training was conducted using flower images taken from the Oxford Flowers 102 [4] dataset. These images were converted to grayscale (the Y channel of the YCbCr image encoding), and synthetic user hints were picked. The number of hints was taken from a geometric distribution (whose parameter is a tunable hyperparameter of the model; 0.10 was chosen), and the hint locations were sampled from a normal distribution with the mean being the image center. The color at each hint location was sampled around each hint point, weighted using a normal distribution (whose standard deviation is a tunable hyperparameter; 2 chosen). Occasionally (the frequency is a tunable hyperparameter; 5% chosen), the entire image is revealed as a hint, with the same justification as [8]. A 90%-10% training-testing split was selected for the dataset. For every training example, we hope to learn the entire Cb and Cr channels of the input image.

3.3. User Interface

We incorporate the above-mentioned elements into an Interactive User Interface that allows the user to upload an image (in colors or grayscale), input color hints from either the suggested color palette (2) or an RGB color picker, and see colorized outputs in real-time.

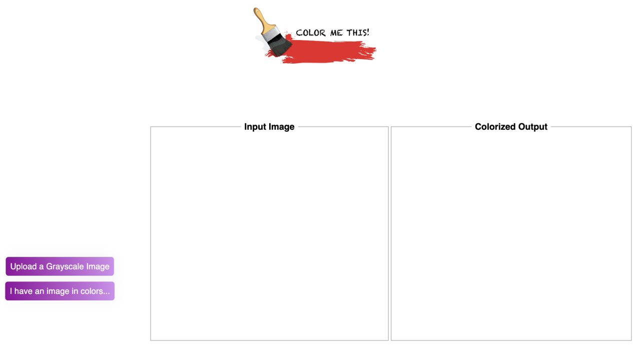


Figure 3. UI welcome page. The user is prompted to upload a grayscale image or a colored image. If they choose the latter, they are redirected to the page shown in figure 4 which displays the original ground truth image, along with its grayscale version.

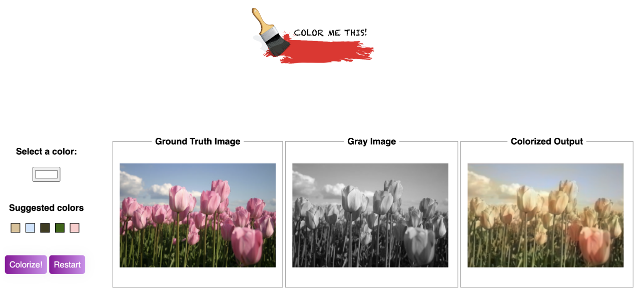


Figure 4. UI for uploading image in colors. Result of automatic colorization (no color hints)

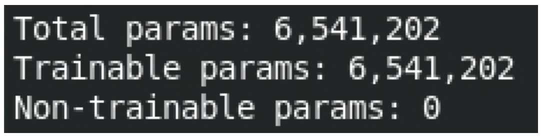
The front-end was built with a combination of JavaScript and HTML/CSS. Frontend/Backend communication was established with API calls serviced through a Flask app and JavaScript's Fetch API.

Initial automatic colorization is presented through clicking the "Colorize" button, while the user-guided colorization is delivered in real-time. Whenever the user inputs a color hint onto the gray-scale image, an API call is made to retrieve the colorized output and display it to the user.

4. Results and Discussion

The CNN model for automatic colorization was trained on the Tensorflow Flowers dataset [6], which consists of 3670 flower images. Training was conducted for about 18-20 hours, for 800 epochs with 8 steps per epoch, followed by 250 epochs with 20 steps per epoch, with a batch size of 20. A learning rate of $1e-5$ was maintained through training. An Adam optimizer was used initially with a Root Mean Square Error (RMS), before switching to Mean Square Error (MSE). It is to be noted that the switch of loss function did not result in a significant performance improvement.

A testing accuracy of about 78% was reached with a loss at 0.025.



```
Total params: 6,541,202
Trainable params: 6,541,202
Non-trainable params: 0
```

Figure 5. Number of parameters for CNN model

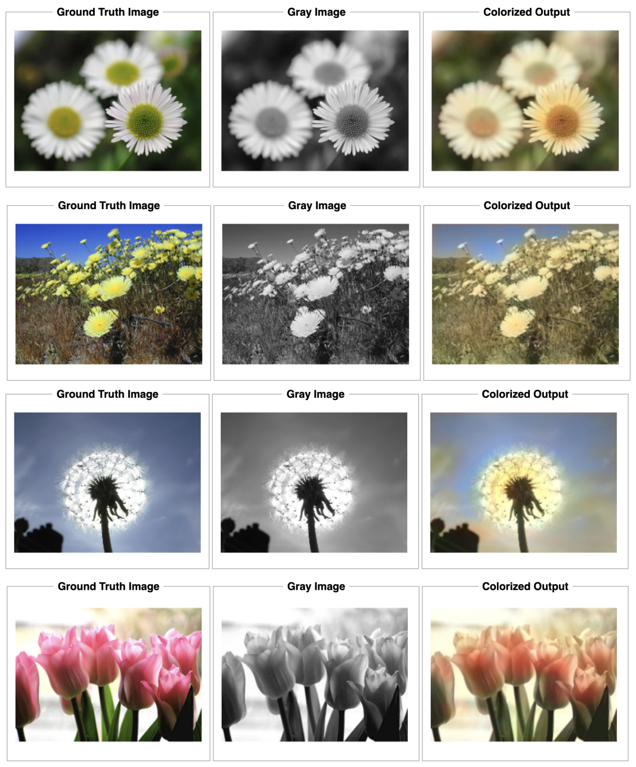


Figure 6. Some results of automatic colorization with CNN/Inception-ResNet-V2.

The results of automatic colorization (6) are of varying quality depending on the image. In general, the model could successfully recognize the sky and discern flowers from grass. We noted that the model yielded the most realistic results when the ground truth image contained shades of yellow. This could be attributed to the prevalence of yellow flowers in the training dataset. Some of the results could also be explained by the small size of the dataset, and the limited GPU resources for training. Nonetheless, despite the imperfections (e.g. graphic artifacts, color splotches) in the colorized results, these images can serve as a valuable starting point for guiding users in choosing how to (re)colorize their image.

As for the user-guided model, its primary use is to accurately recolor images of flowers. A flower with a flat background was selected for this demonstration, and user hints were added in locations of the image that were colored particularly inaccurately 7. After iteratively selecting 4 points and their corresponding colors, the model was able to reproduce the dominant objects in the image. Adding additional hints increases the accuracy of the reproduction, for example.

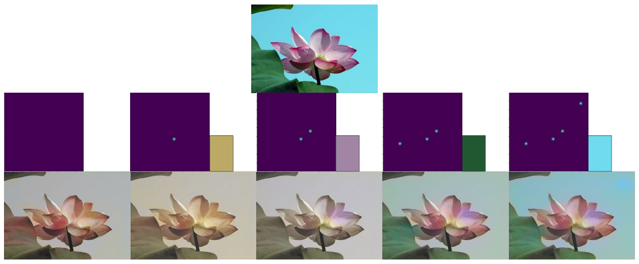


Figure 7. The model's response to multiple hints.

One of the stated goals of the model is image recolorization: the ability to produce images with colors that are different from the ground truth image. A flower with a relatively flat grayscale profile and black background was chosen for demonstration purposes, and 6 colors were chosen to test the model's ability of the model to produce different colors 8. Subjectively, the model is able to integrate a single hint into the image of the flower in a way that looks realistic, despite there being few green flowers in the training dataset, for example.

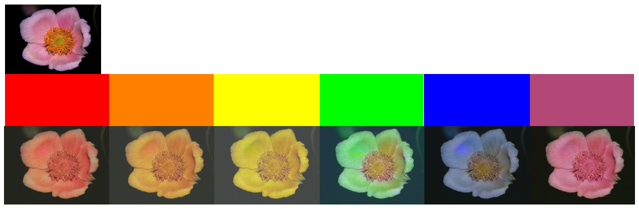


Figure 8. Providing a single color hint to the model. A range of colors were used for the hint. Note, the leaf in the background is occasionally colored green (even without being hinted), and the center of the green flower is yellow, closely matching the ground truth image.

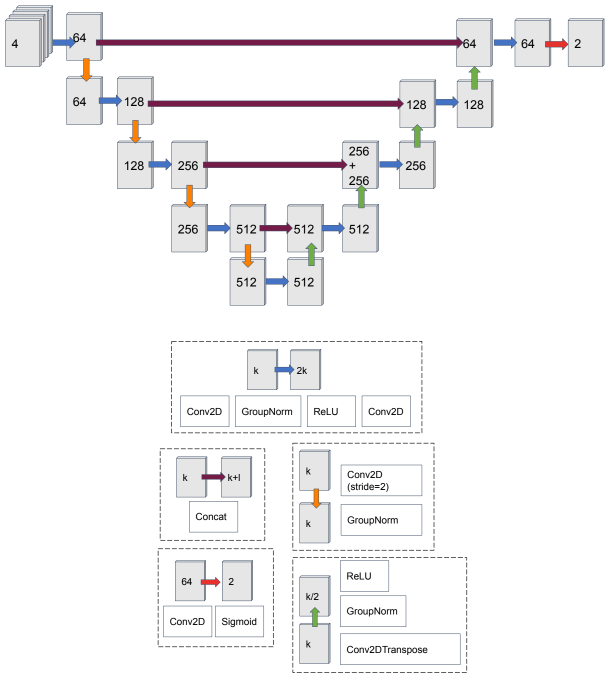


Figure 9. User hint model architecture overview. The key to the arrow meaning is provided below.

4.1. Hyper-parameters tuning

Researchers also conducted a series of testing based on different values of hyper-parameters of the model. The result is summarized in Tables 1 and 2. Figure 11 is an image generated by Tensorboard to show the performance of the model with different hyper-parameters in 20 epochs.

4.1.1 Connecting layers

Type of Connection	best acc	best loss
Only Upper	0.7983	0.0022
Upper and Middle	0.8053	0.0019
Full Connection	0.8104	0.0018
Non Connection	0.7640	0.0028

4.1.2 Regularization Techniques

Regularization used	best acc	best loss
Dropout with 0.25	0.8024	0.0021
Dropout with 0.5	0.7526	0.0039
Max-pooling	0.7477	0.0057
Non regularizer	0.7640	0.0028

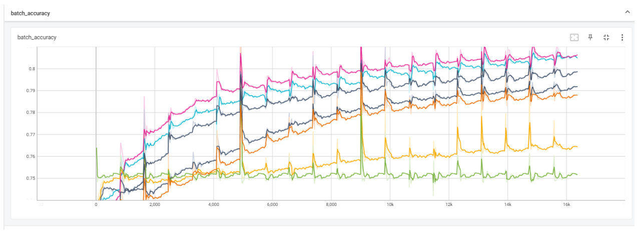


Figure 10. Performance for the model under different hyper-parameters.

4.2. Transfer

As explained previously, our models were exclusively trained on flower images. That being said, our dataset has significant overlap with general natural images. One of the biggest technical problems that the model has to solve is propagating user hint information from the small hint point to the boundary of the object the user intended to hint at. Because of that, our model is forced to learn object boundary detection, a skill that's transferable to any natural image. One image 11 shows a particularly strong example, in which the model has learned to segment the head of a dog. In this example, no user hints were provided, so naive colorization is presented. While the dog's head does not match the ground truth, it is clear that the model was able to segment the image.

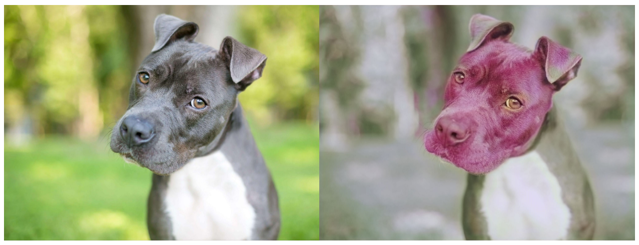


Figure 11. Dog colorized, no hints. It appears that the model analogized the dog's head to a flower's head. Amusingly, the eyes are colored accurately, and the dog's leafy ears are greenish at the tips.

With hints, our model can successfully be used to colorize images of dogs 13. One suspected reason that our model has any success in colorizing images that are so far outside of the domain is that these images are relatively flat in color and feature significant grass, like the training data. Images outside of the training distribution, particularly those with subjects that are not flowers, are not colored accurately without hints. In general, with a dozen hints, our model can colorize simple natural images in a way that looks convincing.

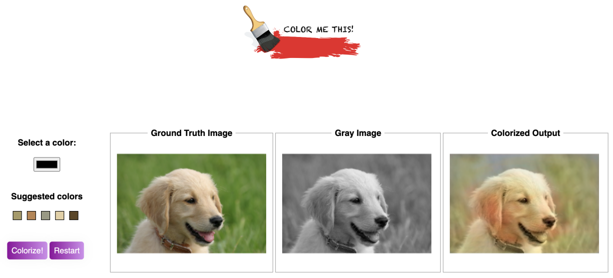


Figure 12. Golden retriever colored automatically without hints.

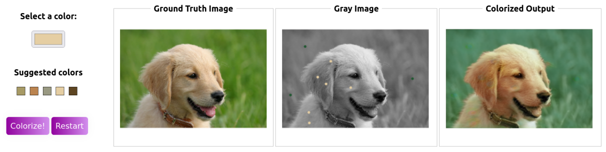


Figure 13. Golden retriever colored with hints.

4.3. Socially-responsible Computing Discussion via Proposal Swap

A concern the authors received about our colorization system is that recoloring images could reveal sensitive information that is hidden in the true colorization. It is true that this is a common problem in photo editing systems – by brightening an image or changing the contrast, fine transitions between dark shades can be come visible and reveal text or other image details that were thought by the photographer to be in absolute shadow. Our system is unlikely to reveal these details. We duplicate the luminance channel of the input image in the output. That means our result images have the same perceptual brightness and luminance contrast as the input image. Any color contrast in our output is entirely fabricated by our model and does not reflect detail from the ground truth image.

A second concern the authors received is that the output of our model could reproduce whatever bias is present in the training set. In particular, the concern is that users may feel that they are not represented in their race or setting is not reproduced accurately. The suggestion provided in the concern is to notify our users of known biases in the result of the model. Since our models were trained exclusively on images of flowers, bias from human skin in the training data will not be present. However, conceivably, our models could coincidentally color some images of humans less accurately than others. The behavior of our models on these images outside of the training distribution (particularly images featuring humans or man-made settings) is unpredictable. In order to address this concern, it is essential that before deploying this model, experiments be conducted feeding our model a diverse set of images. Consistent patterns or egregious colorizations could be identified. If need be, our model could be fine-tuned on a diverse set of images in order to improve the accuracy in these scenarios. A disclaimer explaining the training dataset and limitations is warranted.

A third concern the authors received is that in obtaining our dataset and training our models, we may waste a large amount of energy. There was an insignificant energy cost in obtaining the dataset, as we downloaded datasets [4] [6] constructed by others. The concern with training is warranted, as our models are particularly large: some with several dozens of millions of parameters. Because of their large size, a significant amount of time and energy was spent training. The authors are emotionally divorced from this cost because the training was done remotely on machines owned by the Brown CS department, Amazon, and Google. While training our models has undoubtedly caused untold environmental damage, there is a silver lining. Since our models are particularly good at recoloring natural scenes, we can assist in producing images that are effective at communicating the inherent beauty of nature and the need to keep fossil fuels in the ground. For this reason, the authors believe that the net impact of their work is positive for the environment.

5. Conclusion

In conclusion, we produced a dynamic image colorization system that has moderate success in colorizing images with no hints and is able to produce realistic results conditioned on provided user hints for a variety of natural images. We conclude that some degree of human intervention in the form of user-guided color hints serves the task of colorization well, and yields promising results.

References

[1] Federico Baldassarre, Diego Gonzalez Morin, and Lucas Rodes-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2, 2017. 1, 2

[2] Fastai. Fastai/imagenette: A smaller subset of 10 easily classified classes from imagenet, and a little more french. 6

[3] Rina Komatsu and Tad Gonsalves. Comparing u-net based models for denoising color images. *AI*, 1(4):465–486, 2020. 2

[4] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 2, 6

[5] Adrian Rosebrock. Opencv gamma correction. https://pyimagesearch.com/2015/10/05/opencv-gamma-correction/, 2015. 2

[6] The TensorFlow Team. Flowers, jan 2019. 3, 6

[7] Emil Wallner. Colorizing b and w photos with neural networks, Aug 2019. 1

[8] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016. 1, 2

Appendix

Team contributions

Ethan Williams Ethan was responsible for architecting the user hints model. He found flower datasets, wrote code to sample the image color around user points, generated a user hints encoding to feed to the neural network, and created a dataset from flower images and synthetic user points. After Yipu implemented the model from the paper, Ethan reduced the size to be trainable on the available hardware.

Yipu Gao Yipu was responsible for building and training the fundamental model and comparing model performances under different hyper-parameters. In order to find the best model under the computational bound, Yipu tried different types of regularization terms, tried different methods of downsampling, and tested out the performance for connecting layers. Last but not least, Yipu tried to use the pre-trained weights from the original paper to reduce the number of trainable parameters needed for the model, but the conversion between pth to pb (from PyTorch weights to TensorFlow weights) was just too difficult to accomplish.

Nada Benabla Nada was responsible for implementating the User Interface, and setting up the APIs to establish a connection between the front-end and the back-end colorization algorithms. She implemented and trained the CNN encoder-decoder model with the pre-trained Inception-ResNet-V2 model for automatic colorization, and used K-Means clustering to generate a suggested color palette. She first trained the model on ImageNette and ImageWoof [2] (both subsets of ImageNet), before switching to a smaller flowers dataset. She used some image enhancement techniques such as gamma correction to improve the suggested color palette.