Colorizing grayscale images using a Convolutional Neural Network

Emilie Klefbom
klefbom@student.chalmers.se

Sara Mjöberg
saramj@student.chalmers.se

***Abstract*—Given a grayscale picture as input, this project looks at the problem of generating a plausible color version of the image. This is a widely researched field and some existing models can generate believable color schemes for grayscale images. However, this project aims to observe how models susceptible recolorization models are to data bias and if we can use biased models to generate more vivid and more believable predictions. We created a blue, yellow, and green dataset and trained three models with the same model structure on these three datasets. We then combined datasets and models separately to create an additional four models and compared the results. We found that the combined models were able to generate more believable output images than the models that were trained on the combined datasets. The models were evaluated by visual inspection and by comparing the mean squared error loss. We also compared the results to a baseline model that was trained on an unbiased dataset.**

***Index Terms*—Colorization, Vision for Graphics, CNN**

I. Introduction

Not too long ago colorizing black and white images would be a tricky task and would have to be done by hand using digital editing software. This work is quite tedious and time-consuming and requires skill from the people working with it. But since black and white images were the norm for so many years, the demand for tools to recolor these images has inspired the development of recolorization models that we have today. With the rise of convolutional neural networks, researchers were able to train these on grayscale images with the original colored image as the true label and develop networks that can recolor images in a matter of seconds.

The naive solution to this problem is to think of recolorization as a regression task where the model tries to predict the color value for each pixel of a grayscale image. We can implement this using CNNs and a regression loss such as mean squared error (MSE) to train the model. One of the more prevalent issues with this type of implementation is the difficulty to generate vibrant predictions. Images tend to get desaturated since regression losses like MSE punish vibrant colors more harshly if they are incorrect. Zang et.al [1] proposed an alternative solution that helps minimize this issue, by using a tailored loss function. To tackle the multi-modality of the colorization problem they instead predict the distribution of possible colors and reweight the loss at training time to emphasize rare colors. They then predict the color for each pixel as a classification task.

In this project, we have chosen to use the more naive implementation and explored if we can encourage models to predict more vibrant colors that are perceptually realistic to a human being. To do this we trained models on biased datasets that favor certain colors and then created the new predicted image by taking the average of the biased models' predictions.

We hypothesized that the biased models will tend to learn much more vibrant colors and roughly where to place them. When combining the models we hoped that the different models would dominate in certain areas, i.e. the blue model will dominate the sky patches of an image and the green model might dominate the grass patches while still generating a vibrant and realistic prediction on the image as a whole.

II. Methodn

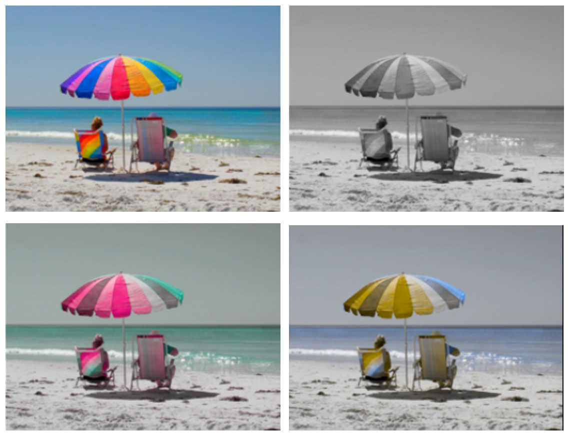

Fig. 1. Top left: Original RGB image. Top right: Lightness. Bottom left: A-channel. Bottom left: B-channel. To plot the lightness channel all A and B channel values were set to 0. When plotting the A and B channels we combined the respective channels with the lightness channel as well as a filler channel with 0 values for the unused color channel. An illustration of only a and b channels can be seen in Fig.3.

To be able to generate vibrant colors we chose a landscape dataset from Kaggle [2] which contained mostly vibrant images. For our model structure we used the first layers of ResNet18 [3] [4], illustrated in Fig.2 to encode our image and then decoded it with a CNN consisting of convolutional layers and upsampling, illustrated in Fig.4. We trained the model using adam as our optimizer with a learning rate of 0.001 and no weight decay. This model structure has been used in several hobby CNN projects [5] [6] and it is the model structure we used for all different datasets. Since ResRet18 is optimized for images of size 256x256, we resized and padded the images with black borders to keep the full image content.

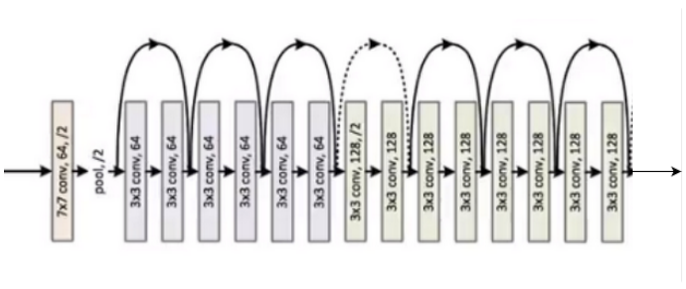

Fig. 2. ResNet layers used. Image taken from [7]. Note that the network takes in images of size 256x256x1 where values ranges from 0 to 1.

To implement our model we converted our images to the LAB color space [8]. Similar to RGB each pixel color gets represented by three dimensions but in the case of LAB these are, lightness (L): which corresponds to the grayscale value, the A channel: which represents a green to red scale and the B channel: which represents the blue to yellow scale. In Fig. 1 one can see an example of an RGB image and the corresponding lightness, A and B channels. Transforming the images to the LAB color space allowed us to use the L channel as input. The size of the input was therefore reduced to a one-channeled grayscale image but more notably the model can predict only the A and B channels. The predicted AB result is then concatenated with the original L input to create the final predicted image. This is illustrated in Fig.3. Since the L values remain unchanged the predicted image loses no information during prediction, unlike an RGB prediction where the lightness can change when we change the color.

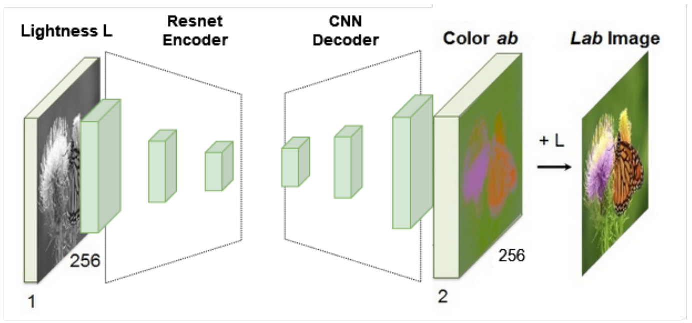

Fig. 3. Overview of model and image format at different stages. Part of the image taken from [1]

To train our biased models we created three datasets which consisted of 24 blue, yellow, and green images. These images were selected by observation. These are notably small datasets, this was mostly due to the fact that selecting the images was time-consuming and due to the large number of models, larger datasets would have taken too long to train.

After this, we created two additional datasets, the BY dataset containing all images from the blue and yellow datasets, and the BGY dataset containing all images from the blue, green, and yellow datasets. Lastly, we selected 864 images at random from the full dataset for our baseline model. From this, we trained six models using the model structure previously described based on each of these datasets and evaluated them on a validation set with 32 images of mixed colors that had no overlap with any of the training datasets. Since the datasets were of different sizes, each model ran for 2000 batches with a batch size of 4 in an attempt to make the evaluation more fair.

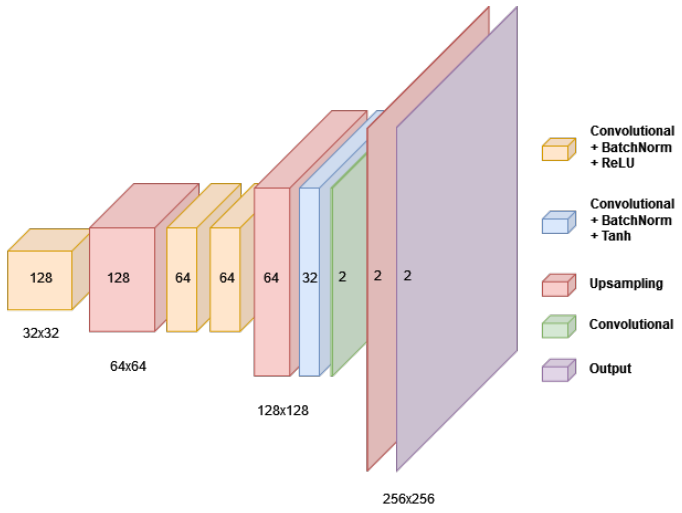

Fig. 4. Upsampling network. Before training the tensors were normalized. Output tensor value, therefore, ranges between -1 to 1 since A and B values typically range between -127 to 127. This is why our final activation is tanh instead or ReLU. For upsampling the 'nearest' upsampling algorithm is used.

Following this, we created two combined models. Note that these models were not trained but instead created as a mathematical combination of single color models mentioned above. The BY-combined model averages the predicted values from the blue and yellow models while the BGY-combined model averages the predictions from the blue, yellow, and green models. The color models were all weighed equally in the combined models. We evaluated these models on the same validation as the other models.

III. Result and Discussion

As previously mentioned each image was padded and reduced in size. This was done instead of for instance cropping to keep the image subject intact which made it easier for a human to inspect the result and see if they were believable or not. This method of resizing the images did not seem to have any significant effect on the results.

For this report, we selected two images from the validation set. The first will be used to show when the model performs well and the other to illustrate a case where the biased model performs poorly. The first image can be seen in Figure 5 below. In this image, we can also see an example of the issue mentioned throughout this report, that is that the image predictions typically get desaturated which does happen for the baseline prediction of this image.

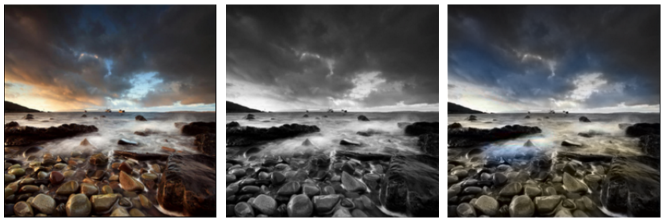

Fig. 5. Left: original image, Middle: Grayscale version, Right: Baseline model prediction.

The result from the models trained on single color-focused datasets can be seen in Figure 6. One can see how these models have learned to favor the color that they have been trained on and that the resulting predictions are more vibrant than the baseline. However, this seems to come at the cost of believability since for the green and yellow models our water is colored in the wrong color. An interesting observation is that the yellow model still predicts the sky as being blue whereas the green model does not. When looking at our datasets we saw that in the yellow dataset there are still some clear blue skies depicted whereas in the green dataset the sky is mostly clouded and grayish.

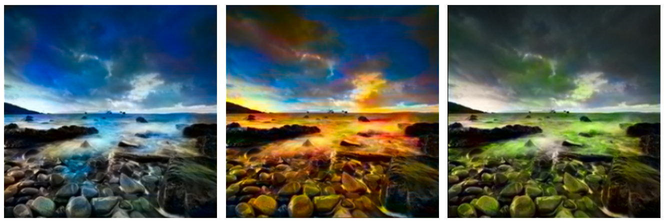

Fig. 6. Left: Blue model prediction, Middle: Yellow model prediction, Right: Green model prediction.

In Fig. 7 one can see that the BY and BGY models are still quite saturated while they produce more realistic results than the single color models. We can see that both images have now learned that the sky is most often blue but the water still looks unrealistic and seems to be mistaken for hills.

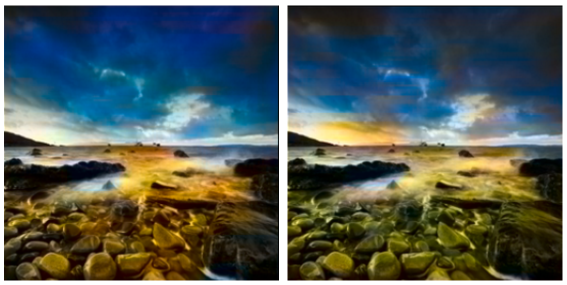

Fig. 7. Left: BY model prediction, Right: BGY model prediction.

Lastly, in Figure 8 we can see the results from the BY-combined and BGY-combined models. Although not perfect we can see a notable improvement from the previous models in how believable the output looks. Even compared to the baseline image the combined image prediction might be interpreted as being a more believable prediction. On the other hand, the model still differs from the original image but compared to the BY and BGY models which have access to the same data it looks more realistic while still being quite saturated.

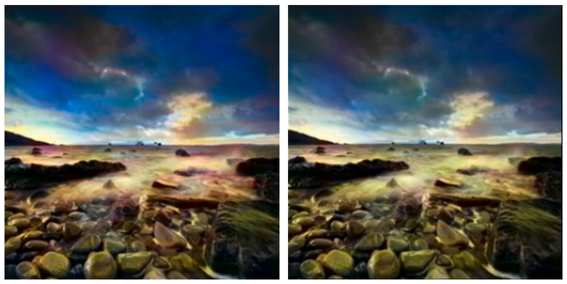

Fig. 8. Left: BY-combined model prediction, Right: BGY-combined model prediction.

For this image we considered the BY-combined prediction to be the most realistic. Another interesting result can be seen when comparing the losses of the models which can be seen in Table I.

TABLE I

Model	Loss
Blue	027795305009931326
Yellow	0.03527559107169509
Green	0.025983819738030434
BY	0.023837938671931624
BGY	0.019840738968923688
Baseline	0.13707154616713524
BY-combined	0.015555263729766011
BGY-combined	0.014605537406168878

The combined models significantly outperform the BY and BGY models which technically had access to the same amount of data and training time. But here one can also observe that even though the baseline model had access to magnitudes more data the combined models manages to get a loss close to that of the baseline.

When inspecting the image predictions in the validation set we saw that when the original image is quite desaturated the combined models perform a lot worse by both over-saturating the image and also becoming more patchy in its color predictions. In these cases, the baseline was in general more believable. This can be seen in Fig. **??**

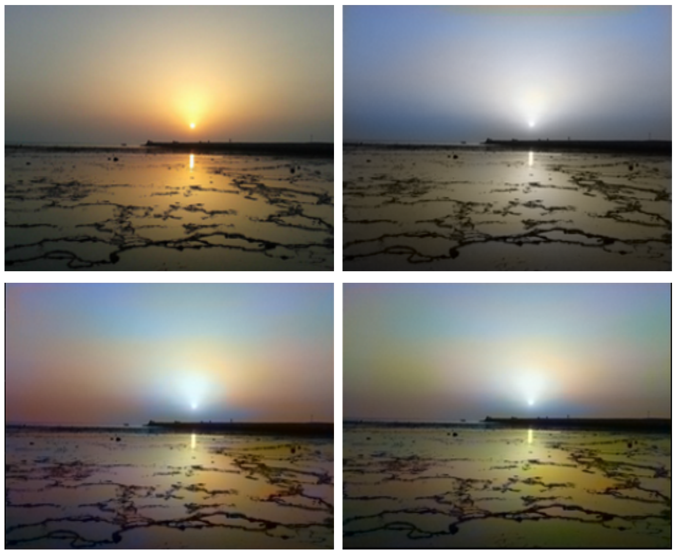

Fig. 9. Top left: Original RGB image. Top right: Baseline prediction. Bottom left: BY-Combined prediction. Bottom left: BGY-Combined prediction.

The patchy predictions may stem from something similar to what we looked at before. In darker areas of the sky, the blue model dominates but in lighter areas where the blue model may have learned to see white clouds the yellow model dominates thinking there might be a sunset. The combination generates these unnatural-looking patches. With this, we can see that even if the model might generate more vivid colors it's not always more believable.

Another important conclusion we can make from this is that the baseline model although looking believable doesn't look like the original image. This is an important conclusion for the recolorization problem. The model learns to identify features of the image and assigns them the most likely color but there may be several plausible colors for the same feature. In this case, we can see that most models learn that the sky is most often blue but we can also see that during a sunset the sky might take on different colors. This multiplicity is part of what makes the colorization problem so difficult. Since a grayscale version of an image can be seen as a compression of the original image there is information loss and the result of the recolorization network isn't always the most accurate but rather a believable version of how the original image could have looked.

Therefore we can't make hard conclusions from the result of these networks of what the original color was because as we've seen the predictions depend on the distribution of our training data and it will predict the most likely color.

IV. Conclusions

During this project, we have been forced to cut some corners in how big our datasets were and for how long we were able to train our model. Because of this, the datasets are very small and the training hasn't fully converged when we stopped training our models. Because of this, the results may shift slightly if one were to redo our experiments for more data and allow longer training but we still believe that the overall results and conclusions would hold.

Our suggestions for furthering this project would be to try and intentionally train on the A and B spectrums separately and see if combining these models could produce better results. For example generate training data that only contained the magenta, cyan, yellow and blue values and then ensemble these models. Another suggestion is to try similar experiments on models using different loss functions to see if the hypothesis about combining models still holds.

We were able to prove the hypothesis and train models to identify certain vibrant color patches and then combine the results to get more vibrant color predictions. In general, this project was more of an exploration into the biases of colorizing models and better loss functions. From our experiment, we have been able to conclude that recolorization models are heavily influenced by bias in the input data, and the multiplicity of recolorization suppresses less occurring options such as sunsets compared to blue skies. We have also seen strong indications that MSE loss may be an unsuitable loss function for the recolorization problem as it can easily generate patchy predictions as well as desaturated images. Today there exists models which better generalized on large amounts of data such as [1] and we believe that this is the direction recolorization should take in the future.

References

[1] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *CoRR*, vol. abs/1603.08511, 2016. [Online]. Available: http://arxiv.org/abs/1603.08511

[2] "Landscape pictures," Available at https://www.kaggle.com/datasets/arnaud58/landscape-pictures?resource=download.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[4] Pytorch, "Resnet18," https://pytorch.org/vision/master/models/generated/torchvision.models.resnet18.html read 21/10/2022.

[5] L. Melas-Kyriazi, "Image colorization with convolutional neural networks," Available at https://lukemelas.github.io/image-colorization.html published 15/05/2018, read 21/10/2022.

[6] E. Lewinson, "Image colorization using convolutional autoencoders," Available at https://towardsdatascience.com/image-colorization-using-convolutional-autoencoders-fdabc1cb1dbe published 17/04/2020, read 21/10/2022.

[7] P. Mahajan, "Understanding residual network (resnet)architecture," ://medium.com/analytics-vidhya/understanding-resnet-architecture-869915cc2a98.

[8] Wikipedia, "Cielab color space," Available at https://en.wikipedia.org/wiki/CIELAB_color_space last edited 15/10/2022, read 21/10/2022.