CS231N Final Project
Line Drawing Colorization

Yuki Inoue
Department of Electrical Engineering, Stanford University
450 Serra Mall, Stanford, CA 94305
yinoue93@stanford.edu

Abstract

Sketch colorization is important but often skipped when drawing illustrations due to its time consuming and monotonous nature of the procedure. However, adding colors to illustrations almost always enhances the drawing, and it is a shame to having to skip such an important step. In this paper, we propose a neural network based method that automates the illustration coloring process. More specifically, we implement models inspired by recent successes in photo colorization problem, which takes in black and white images as the input, and apply them to sketch colorization problem. We find that photo colorization problem and sketch colorization problem share many characteristics, such as classification models outperforming regression models, and allow models to execute the task at a very high level.

1. Introduction

Drawing is a two step process: first, draw the outline of the objects, and then add colors. Of these two drawing steps, many can agree that making the sketch requires more creativity and carries more importance. However, the colorization step is not something to be scoffed off either: colorization immensely enriches the quality of the sketches, and make them more memorable. Unfortunately, colorization step is often skipped, due to its time-consuming and monotonous nature.

In recent years, many Convolutional Neural Network (CNN) based models are proposed to solve the similar task of *photo colorization*, where the task is to add colors to black and white images, and have been shown to perform at a very good high level. Encouraged by this result, we hypothesize that sketch colorization can also be fully automated with CNN's. In this paper, we will implement various CNN models to automate sketch colorization. Much of the first part of the work is dedicated to recreating the network structure proposed in Zhang et. al. [14], and also explore the performance difference between the regression loss and the classification loss. Then we move onto improving the model and the training process to target problems we observed from the baseline models.

2. Related Work

Automatic sketch colorization is a problem not very heavily explored yet. As far as we know, there is no paper published on this specific topic, although there is one notable attempt by a Japanese company Preferred Networks (PFN), which have developed a web-based tool called PaintsChainer [1] [2]. They argue that due to the extreme under-determined nature of the problem, typical autoencoder-type network structures that "reconstructs" the input to output by downsampling then upsampling do not perform well for sketch colorization. They claim that the best such a model can do is to detect which pixels are grouped together, and color them with desaturated colors. So PFN instead bases their model on Generative-Adversarial Networks (GANs) to tackle this problem, and it works very well. In GAN structure, two networks play a game, where the generator tries to create a realistic output from an input and a random noise, and the discriminator tries to decipher between the real and the fake outputs [7]. However, PaintsChainer does have couple problems that GANs typically have- first, it is very hard to train, and second, depending on the input, the network becomes unstable and outputs faulty images.

To circumvent these problems, we will develop non-GAN based models- we believe that the problems mentioned by PFN can be solved by applying techniques developed for the similar field of *photo colorization*, in which one tries to colorize black and white images instead of sketches. Photo colorization and sketch colorization share obviously similar problem structure; they are both trying to add colors to the input, and are solving an underdetermined problem in the process. So we reasoned that techniques developed

for photo colorization can effectively be applied to sketch colorization.

One common sentiment that is prevalent in various literature in photo colorization is that classification models outperform regression models [5]. This is because the squared loss criterion for each pixel forces the network to optimize by simply averaging the pixel values, resulting in sepia-colored outputs [6]. In order to do classification, however, the output pixels need to be binned into discrete classes first, often defined by a range of color values. In Zhang et. al.[14], they go one step further and adds a weight term for the classification loss to account for the rarity of different pixels.

After converting the problem to a classification one, many works have explored other means to help the network to learn to output realistic images. In Iizuka et. al. [8], they take advantage of image tag information in their model. They argue that there are tags such as "outside" and "human," that can be learned to infer the lighting of the image and possible objects in the image. Unfortunately, this cannot be applied to sketch colorization, as we could not find a big enough website that host tagged images. Another approach taken in photo colorization is to predict a part of the image by imputing a portion of a colorized image to black and white region [3] [9]. This is a very unique idea that could be applied to sketch colorization. Finally, in Cao et. al. [4], they applied GAN-based network to tackle the photo colorization problem.

It is important to note that although the two colorization problems share much in common, the difference in the inputs does make the two problems unique. For example, blindly applying the trained model for the photo colorization problem, shown in Fig. 1 does not result in an appealing output. We claim that the two colorization problems differ in at least 3 major ways. First, due to the difference in the type of the inputs, photo colorization is a 2-channel prediction task, unlike sketch colorization, which is a 3-channel prediction task. Second, because the colors of illustrations do not have to actually represent real life, they tend to be more arbitrary. For example, realistic hair colors are black, blonde, red, and brown, but for the sketch colorization problem, hair colors can literally be anything-green, purple, pink, etc, making prediction harder. Finally, illustrations tend to have more "blank" pixels- backgrounds of illustrations are typically monotonic, which is something rarely observed for real photos. These three points imply that sketch colorization is more underdetermined and prone to desaturated outputs.

Figure 1. Applying a trained photo colorization model to sketches

3. Methods and Technical Approach

3.1. Dataset Aggregation

The main focus of this paper is to create a CNN-based structure that colorize line drawings. Unfortunately, as most of neural network researches today focuses on real-life images and not hand-drawn illustrations, there are no publicly available datasets that only contain illustrations. Therefore, the first step of the project was to create a dataset.

We created the dataset by first crawling for already colored illustrations on a website. We chose *zerochan.net* to crawl from, because it only contains hand-drawn images, and it has over 1.5 million images, mostly colored. After downloading the images, any images that are too small (smaller than 256 pixels in either dimensions) or devoid of colors are removed. Then, a series of computer vision techniques are applied (via OpenCV) to convert colored images to sketches. Block diagram of the colored to line drawing pipeline is shown in Fig. 2. On top of the color conversion, it is during this step that the images are reformatted to 256px by 256px, the input size of the network. For non-square images, they are divided into sliding windows of 256px by 256px.

The examples of the resulting dataset images can be seen in Fig. 3. Before cropping the images, we had 0.6 million images. The dataset was then split into 80/10/10 into training set, test set, and the validation set.

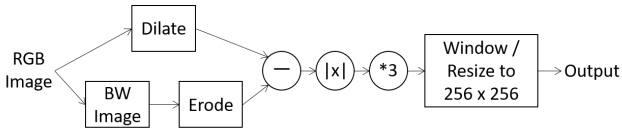

Figure 2. RGB to Line/Binary Image Conversion Pipeline

3.2. Network Structure

As mentioned before, there has been many great models for the photo colorization problem recently, and we would like to base our model off of their success. Therefore, we chose to base our networks on the model described in Zhang et. al. [14]. The model has a typical autoencoder-type structure, first reducing the input image by eight times, and then upsampling it to the output image. The details of the model will be explained in more detail in the proceeding sections as well as the original paper, but the main idea is that the

Figure 3. Example Dataset Images

downsampling procedure extracts the essential features of the input, and tries to "reconstruct" the target image (in our case, the colored image) by utilizing the extracted features. For our project, two different models are implemented, the regression and the classification models, and four variations of the classification models are explored.

3.2.1 Regression Model

A natural way to formulate the problem is to treat it as a regression problem. In regression model, the loss is calculated by taking the pixel-wise L2 distance between the predicted and the ground truth:

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

The model diagram for the regression model is Fig. 4. As mentioned before, the overall structure is based on Zhang et. al. model, but with pass-through layers (indicated by arrows and dotted layers in the diagram) to encourage faster and better learning process. This idea of pass-through layers is inspired by Unet model proposed in Ronneberger et. al. [12], and has also been shown to improve CNN models for many different tasks [13]. After prediction is made, the input layer is merged by multiplying the two layers together. This is omitted from the diagram, but before the multiplication, the input layer is actually divided by 255, so that the input layer acts as a masking layer.

As mentioned earlier, models trained with regression loss do not output realistic colors for the photo colorization problem. This is due to the fact that there are objects that have multiple possible colors. For example, even if two images of flowers have the same shape, they can be very different colors. to account for such a variability, the model takes the mean of all possible colors, typically ending up with some desaturated pixel, a color not valid for flowers. This underdetermined nature of the problem is present in the sketch colorization as well, so we expect the regression model to not perform well. Despite this, we implemented it as a baseline model, and to confirm that sketch colorization is indeed similar to photo colorization.

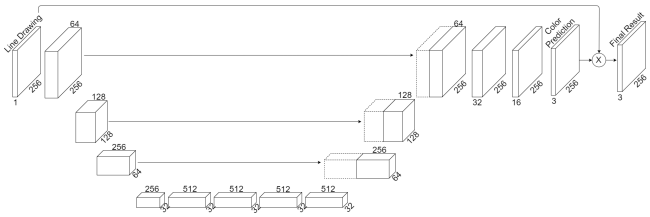

Figure 4. Regression Model, based on *Unet*. The arrows and the dotted layers indicate pass-through layers.

3.2.2 Classification Model

Instead of using regression loss to train the model, many successful photo colorization models are trained using classification loss:

$$L_{cl,wei}(\hat{Y}, Y) = -\sum_{h,w} v(Z_{h,w}) \sum_{q} Z_{h,w,q} \log(\hat{Z}_{h,w,q})$$

where h, w, and q represent the height, width, and channels of the pixel, $Z_{h,w,q}$ denotes the ground truth soft-encoded probabilities, $\hat{Z}_{h,w}$ denotes the predicted log-probabilities, and finally, $v(Z_{h,w})$ denotes the class weights explained later. For the simple classification model, $v(Z_{h,w})$ is set to be 1.

In the classification model, the prediction is still made pixel-wise, but instead of predicting the value of the 3 channels, the model predicts which class the pixel is likely to be in. Here, the pixel class is defined by the ranges of each channel. For example, class 3 may correspond to the pixels with R value between 32 64, G value between 0 32, and B value between 0 32, and so forth. For the project, each channel is equally divided into 8 bins, making the total class number $8^3 = 512$. Also, following the implementation mentioned in Zhang et. al., we decided to encode the pixels using *soft-encoding scheme*, in which each output pixel is not encoded as a one-hot vector of the closest class centers, but a distribution of the classes, weighted according to closeness to the class centers. The "distances" between the ground truth pixel and the class centers were measured using a Gaussian distribution, with $\sigma = 0.25$.

Classification loss works better for underdetermined systems because under classification loss, the model does not average all possible color options. Going back to the flower example from the previous section, if red, blue, and green are all possible colors of flower with an equal likelihood, a well trained classification model assigns 0.33 to each of the three options. At sample time, the model may output any of those 3 colors, and they are all respectable colors for flowers.

The network diagram of the classification model is shown in Fig. 5. This is almost an exact carbon copy of the model proposed by [14], except that the input layer is

merged with the prediction layer via multiplication instead of addition, and there are 512 classes instead of 313 classes. One detail to note here is that because there are 512 classes, the last layer of the model becomes too enormous for the GPU memory that we have. So instead of fully upsampling the input to 256 by 256, the model only upsamples to 64 by 64 and extrapolates by 4 times to 256 by 256 using linear extrapolation, which is ok, but suboptimal nevertheless. This especially hurts sketch colorization, because the lack of details cannot be masked when merging with the input layer like it can be for photo colorization. For the rest of the report, we will call this model *classification model*.

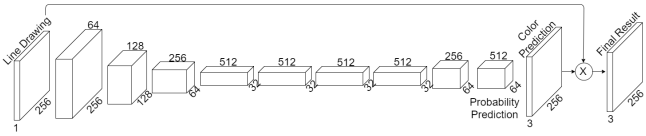

Figure 5. Classification Model, based on Zhang et. al. paper

After the simple classification model was implemented, a few more improvements are made. First, as suggested in [14], class rebalancing term, denoted as $v(Z_{h,w})$ was implemented. This term makes sure that rarer pixel classes are weighed more than common pixel classes. More specifically, they are calculated as follows:

$$w \propto \left((1-\lambda)\tilde{p} + \frac{\lambda}{Q}\right)^{-1}, \quad \mathbf{E}[w] = \sum_q \tilde{p}_q w_q = 1$$

where $\tilde{p}$ is the sampled distribution of the pixel classes in the dataset, $Q = 512$ and $\lambda = 0.5$. The first equation calculates the unnormalized weights to be assigned to each class from the sampled rarity of the classes, and the second equation normalizes the weights so that the cross entropy loss stays at a similar level as before applying the weights. $v(Z_{h,w})$ is then defined by the normalized weight value for the class closest to the pixel at h, w. For the rest of the report, we will call this model *rebalance model*.

Another improvement made is the colorspace conversion from RGB to CIE-LCH. Though discussed in more detail later, the outputs of the classification loss with class rebalance term were still somewhat desaturated. We hypothesized that the reason for this is because of the extreme underdetermined nature of the sketch colorization problem. Even though classification loss may be more robust against color variation problem, desaturation could still occur if the possible colors for a pixel span across many classes. So we further hypothesized that for the same object, the hue information may vary greatly, but the brightness and the saturation information may be more consistent. If this is true, by converting the dataset to a colorspace such as CIE-LCH that concentrates the hue information in one channel, the class distribution of pixels will only span across one axis (i.e. the hue dimension) at most, even for objects that can take on various colors, making the model even more robust against underdeterminedness of the problem. For the rest of the report, we will call this model *LCH model*.

The third and the final improvement made on the model is adding another loss term. As we started to train the models and collected the outputs, we saw that the biggest problem was that the output is desaturated. So we decided to specifically shape the loss function to punish the model if it predicts classes that have vastly different saturation values. More specifically, we created a loss matrix M_{sat}, which is 0 for any element that correspond to predicting classes that have the same saturation levels from that of the ground truth, and 1 otherwise. The loss function is updated as follows:

$$L_{class} = L_{cl,wei} + \lambda_{mix} Z^T_{h,w,q} M_{sat} \log \hat{Z}_{h,w,q}$$

where $L_{cl,wei}$ denote the original cross entropy loss, and $Z_{h,w,q}$ and $\hat{Z}_{h,w,q}$ are as defined earlier, and λ_{mix} is the mixing term for the 2 losses. For the rest of the report, we will call this model *chroma loss model*.

3.3. CIE-LCH Color Space

For some models in our project, the dataset was converted to CIE-LCH color space. CIE-LCH is a color space with **L**ightness, **C**hroma (saturation level), and **H**ue, much like HSV and HSL color spaces. Unlike those two color spaces, however, CIE-LCH color space is designed to approximate human vision, and enjoys a better perceptual uniformity, which is a desirable characteristic when binning channels into equal-sized classes.

4. Experimental Results and Discussion

In order to save space, sample outputs are aggregated in Fig. 6. For all models, the AdamOptimizer is used, with the learning rate initially set at 10^{-4}, and dropped by a factor of 10 every time the loss function stagnated around the same value. All models took about 1.5 days to train.

4.1. Regression Model

As expected, the regression model output (third row of Fig. 6) is very desaturated. Fig. 7 shows the average chroma (saturation) values for each model, and the regression model has the lowest chroma level at 5.86. If we look at the filters for the model, something interesting is discovered: Fig. 8 shows the filters for the filters for the last layer of the regression model. Since the last layer converts a 16 channel input to a 3 channel output, there are 48 3x3 filters, and we discover that the filters corresponding to the same input channel are roughly the same for all three output channels.

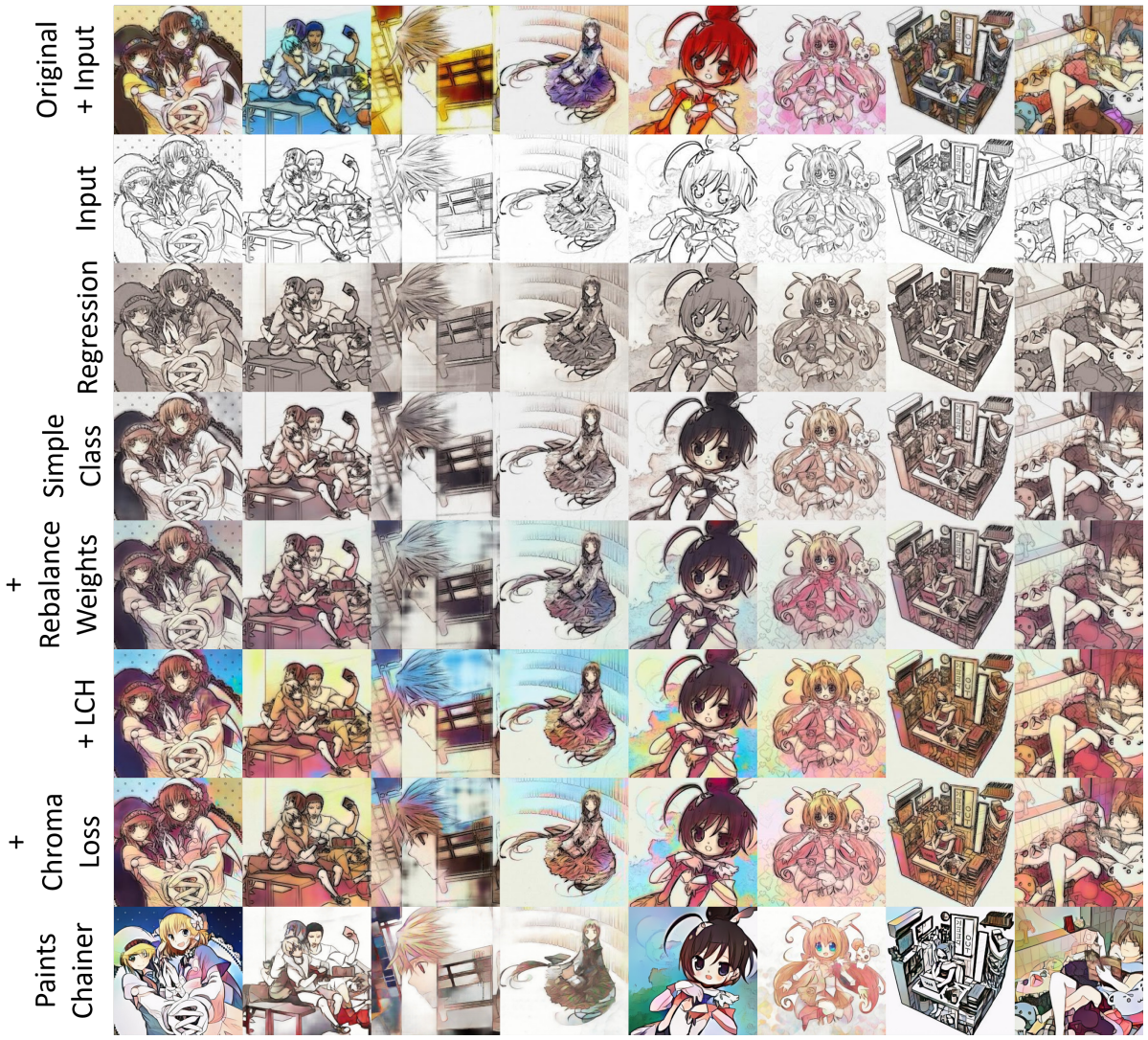

Figure 6. Sample output comparison

In other words, the filters enforce the model to only output monochromatic colors.

Though the model only outputs monochromatic images, it does not mean that the model does not learn anything. For example, it does seem to figure out which pixels belong to the same object, as can be seen from the fact that those pixels are colored the same way. It also seemed to have figured out that human skins are light color.

4.2. Classification Model

As discussed in [14], the predicted class probabilities are converted to pixel colors with "temperature," which is equivalent to applying the softmax function with temperature (shown below) to the predicted probability and taking the mean of the resulting distribution. As in [14], $T = 0.38$.

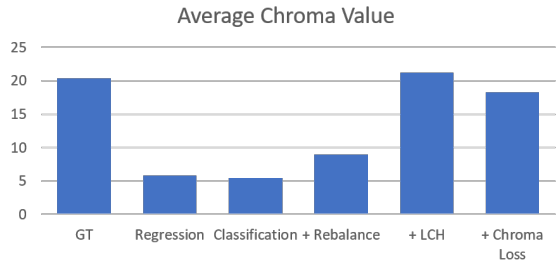

Figure 7. Average Chroma (Saturation) Values for Different Models

$$f_T(z) = \frac{\exp(\log(z)/T)}{\sum_q \exp(\log(z_q)/T)}$$

As expected, the simple classification model (forth row

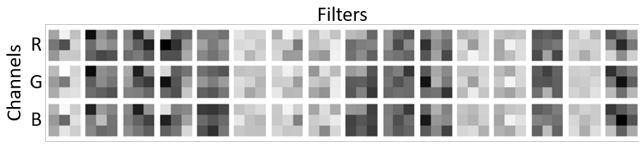

Figure 8. Filters of the last layer for the regression model

of Fig. 6) works better than the regression model. Fig. 9 shows the MSE for the RGB channels, and we see that the simple classification model is able to halve the MSE for every channel from that of the regression model. Also, Fig. 11 shows that the brightness prediction vastly improved as well- this is likely a result of the model not having to average all possible color choices for the same object. However, the images are still very desaturated- Fig. 7 shows that the chroma level for the classification model is 5.43, a little lower than that of the regression model. However, Fig. 10 shows that although the classification model outputs may be more desaturated than that of the regression model, its prediction on the saturation level is more accurate.

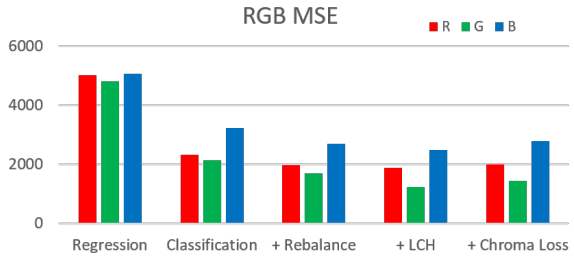

Figure 9. MSE of the RGB channels for Different Models

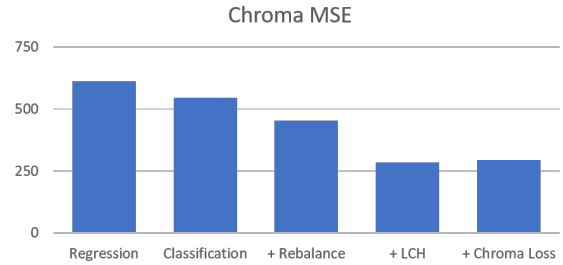

Figure 10. MSE of the Chroma channel for Different Models

By adding the class rebalance term (fifth row of Fig. 6), the output images became more vibrant, again as expected. Fig. 7 shows that the chroma level of the model improves to 8.94, roughly a 64% increase from the previous models. Qualitatively speaking, looking at the result images in [14], the effect of adding the class rebalance term seems to have benefited more for the sketch colorization problem than it did for the photo colorization problem. This makes sense, as illustrations often have "empty" regions of all white or black pixels. This implies that the class representation is much more skewed for illustrations than it is for photos. So without the class rebalance term, the model reduces the loss function by outputting colors that are more common, less vibrant background pixel colors. However, even with the reblance term, the model still struggles to output vibrant pixels, and limits itself to shades of red for the most part.

Changing the colorspace to CIE-LCH (sixth row of Fig. 6) seems to have greatly improved the image quality. It definitely improved the chroma values, as can be seen in Fig. 7, reaching 21.22, which is more than double of that of the previous models, and also on the same level as the ground truth images. Fig. 10 and Fig. 11 also show that the colorspace conversion improved both the lightness and the chroma predictions accuracy, nearly halving the MSE for both. On the other hand, Fig. 12 shows that the hue prediction did not improve much. In fact, if we measure the MSE for the RGB channels, it shows that the prediction did not improve much (Fig. 9). This confirms our hypothesis about brightness and saturation channels to be much easier to predict than the original RGB values.

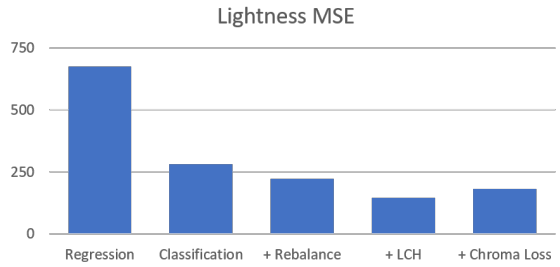

Figure 11. MSE of the Lightness channel for Different Models

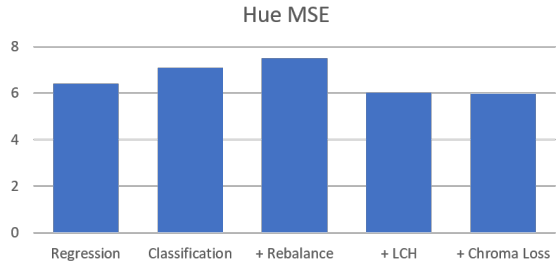

Figure 12. MSE of the Hue channel for Different Models

Finally, adding the loss that specifically target the chroma level (seventh row of Fig. 6) did not seem to improve the result very much. Although from Fig. 6 we see that the model now values hue prediction less from the fact that there are more color inconsistencies compared to the model without the chroma loss, it does not seem to have improved the desaturated images, such as in the second and the eighth columns of Fig. 6. Fig. 7 and Fig. 10 also show that the chroma values did not improve by adding an extra

loss term.

4.3. Survey Result

Evaluating the goodness of colorization can be tricky. Since the goodness of colorization is a very subjective quantity, it is hard to come up with a metric that succinctly summarizes it [11]. Some of the common metrics used in photo colorization literature are object classification accuracy and surveys [10]. In the first of the two metrics, one measures the difference in the object classification accuracy between the ground truth and the colorized images. If the prediction accuracy does not change between the two, then that means that the colorization is successful. However, this is only possible if there is a widely distributed, off the shelf object classifiers available. Unfortunately, such a model does not exist for illustrations (typically, "illustration" is a category for object classifiers to begin with, rendering them useless), so we cannot use this metric for our purpose. So we relied on surveys to measure model performance. In a way, the survey acts as a Turing test for the sketch colorization problem.

To measure the model performance from multiple angles, we created two surveys. First, the surveyees are shown some images, including both the ground truth and model outputs from the test set, and are asked to rate between 1 (not plausible colorization) to 10 (great colorization). The result is shown in Fig. 13. It shows that the surveyees agree that our best models (the LCH model and the chroma loss model) can color sketches at a level rivaling that of the ground truth images. For the second survey type, the surveyees are shown the ground truth image, the output of the rebalance model, and the output of the chroma loss model for the same input image. Then we asked them which one they think was human painted. We decided to exclude the LCH model in this survey, because the outputs of the LCH model and the chroma loss models looked too similar to each other, and were afraid that the surveyees would not choose them because they know that there is only one human-painted image. The result is shown in Fig. 14. We see that when the images are put next to each other, surveyees have a little higher chance at deciphering which is human-colored, but the difference is not very significant.

4.4. Comparison with Paints Chainer and Current Problem with the Model

The last row of Fig. 6 is the output of the Paints Chainer. In general, the outputs of Paints Chainer is more vibrant, presumably because PFN model uses GAN. Because our models are trained in a fully supervised fashion, no matter what techniques are applied to avoid underdeterminedness of the problem, there will be ambiguity in the prediction for at least 1 channel, showing up in the end result as desaturation. On the other hand, GAN's can circumvent the desaturation problem, because the model is trained in a semi-supervised fashion- prediction for GAN's are critiqued by a classifier that predicts if it is realistic output or not, not by how close the output is to the ground truth. In other words, a classifier is better suited for the problem than the classification loss is, because this learning process better models human perception. This is one of the reasons why GAN models outperform other models in generative tasks. Also, the outputs of Paints Chainer are more precisely colored, presumably because their model outputs images that are the same size as the inputs, unlike our models that rely on linear extrapolation at the end.

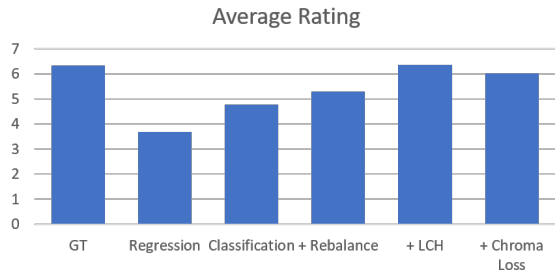

Figure 13. Average ratings of sample outputs

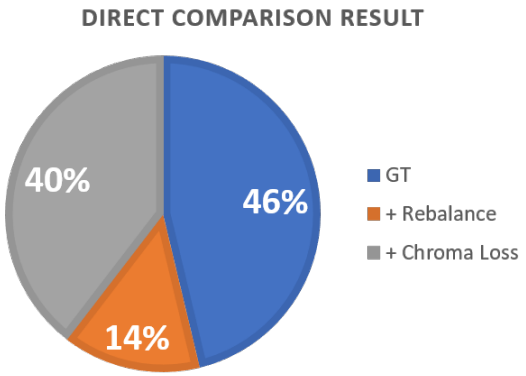

Figure 14. Direct comparison between GT, rebalance model, and chroma loss model

Our model also has a problem in that it tends to do significantly better for the generated input than it does for actual sketch inputs. We hypothesize that this is because the generating algorithm does not threshold the input to 0 or 255 pixel values, while many actually sketches are that way. We chose to not threshold the values to binary input, because the generated sketches appeared more natural to our eyes. However, because of this, our models are very susceptible to the input pixel values, especially to how dark the dark pixels are. This is clearly not ideal, and due to this problem, our model will not be able to accept every sketches like Paints Chainer can.

5. Conclusion

In this project, we explored different models to solve the problem of sketch colorization. We started by implementing the baseline regression and classification models, and incrementally made changes to improve the model performance. We believe that we were able to develop a model that can accomplish the task at a respectable level. Although our model is weak against input variation, and GAN-based models may work better, we are happy to show that non-GAN models can indeed also accomplish the task at a very high level, under right conditions.

For the most part, most of what we developed on the project is based on the successes of photo colorization problem, especially ones by [14]. However, we did also make changes in the model that are more specific to the sketch colorization problem, such as converting the dataset to CIE-LCH color space, and adding a loss term based on the saturation level.

It was also nice to see that some sample outputs showed that the models really learned how to color. For example, Fig. 15 shows an example output that was originally colored with unnatural colors (left). But because the model has learned what a typical face looks like, it colored the image using more natural colors (right).

Figure 15. Original coloring (left) and the model output (right)

6. Future Work

With more computational power and a larger GPU instance, we can train a classification network that can actually upsample the layers back to the input dimensions without the help of linear extrapolation. This will vastly improve the output sample quality. Also, due to the time crunch and the fact that it was a one person project, we did not make any alterations to the model structure described in [14]. So it would be interesting to make changes to the network structure in the future. This may allow for faster learning as well, as it currently takes about 1.5 days to train the models. Finally, we would like to redo the data creation process, so that the input is binary values, to make our model more robust against input variations. Another idea we have is to also train the model with noise for more robustness.

All code for this project can be found at: `https://github.com/yinoue93/CS231N_final_proj`

References

[1] Line drawing colorization using chainer (http://qiita.com/taizan/items/cf77fd37ec3a0bef5d9d).

[2] Paintschainer (https://paintschainer.preferred.tech/).

[3] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

[4] Y. Cao, Z. Zhou, W. Zhang, and Y. Yu. Unsupervised diverse colorization via generative adversarial networks. *arXiv preprint arXiv:1702.06674*, 2017.

[5] G. Charpiat, M. Hofmann, and B. Schölkopf. Automatic image colorization via multimodal predictions. *Computer Vision–ECCV 2008*, pages 126–139, 2008.

[6] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. *CoRR*, abs/1605.00075, 2016.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[8] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4):110:1–110:11, 2016.

[9] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421, 2015.

[10] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2405–2413, 2016.

[11] G. Ramanarayanan, J. Ferwerda, B. Walter, and K. Bala. Visual equivalence: towards a new standard for image fidelity. In *ACM Transactions on Graphics (TOG)*, volume 26, page 76. ACM, 2007.

[12] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[13] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[14] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016.