Back to the Future: Building Upon Image Colorization via Classification

Annika Dahlmann dahlmana@umich.edu
Cameron Davis camdav@umich.edu
Cole Hudson colehud@umich.edu
Kyle O'Laughlin kolaug@umich.edu
Nathan Tseng tsnathan@umich.edu

1. Introduction

The task of image colorization involves producing colored images from grayscale images. When implementing colorization, the plethora of plausible colors from a grayscale image poses a challenge to figure out how to properly train the network to produce colorful images. Our group attempts to address these difficulties to allow the network to perform colorization, which would allow for legacy images taken in black and white to be visualized in color. This is especially useful in the film industry, where colorization was recently used to restore World War II footage and other vintage footage. Not only that, colorization can be extended to provide realistic image results from sketches and semantic segmentation maps [1]. Given the variety of use cases, our group focuses on implementing a colorization model that is able to produce plausible colors for black and white images, by treating the problem through the lens of a classification problem.

When attempting colorization in deep learning, the model receives the L channel and attempts to reconstruct the AB channels. Previous work with a regression approach encounters difficulties with the task definition [2]. Treating the colorization as a regression problem involves directly taking the L2-distance between the ground truth and output colorization. While this intuitively makes sense, the range of plausible colors make the task ill-suited for regression. The results tend to end up with desaturated colors because the minimization tasks ends up colorizing using the most common colors that appear in an image dataset [3].

Our approach is instead inspired by Zhang et al.'s work, which converts the task of colorization into a classification one [4]. This is done through quantizing the AB color space into bins through a palette, where each bin is treated as a separate class. Working in a discrete space subsequently allows for weights to be introduced for each class. By using classification, the model is able to estimate the probabilities of different plausible colors, and rare colors can be weighted to encourage the network to keep the images vibrant and saturated in color.

We therefore attempt to make a lightweight model that performs classification on a quantized LAB color space. For the model training, we implement a similar weighting scheme and weighted per-pixel cross-entropy loss to the Zhang et al.'s method. This is done mostly from scratch, with credit for borrowed code in the Implementation section. However, we also introduce our own new contribution, which is a loss term that measures the difference in output and dataset color distributions to keep plausible colors consistent with the original dataset colors. As a result, we are able to produce colorful results on a lightweight model that trains within a few hours on Google Colab.

2. Approach

2.1. Data Processing and Weighting

We begin by quantizing the image based on the 313 possible AB pairs that are in-gamut from the color space of the dataset. This means that we quantize the original RGB image pixels into distinct bins where each pixel value corresponds to one of the 313 bins. For each pixel, we perform quantization by converting the image to the LAB space, and then placing it in the bin that minimizes the L2 distance between the AB values at the pixel and the palette. Once we have quantized the dataset, we compute the weighting scheme from Zhang et al.'s approach [4]. This weighting scheme gives a weight to each bin based on the rarity of the bin in the dataset. More specifically, we compute the weights w from their prior probability $\tilde{p}$ of the bin appearing in the training dataset from Q non-zero bins. The following equation from their paper is as follows:

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

We additionally use a value $\lambda = 0.5$ as suggested from their paper, which produces the best results. The weighting scheme is important since it assigns a higher weight as a particular bin becomes more infrequent in the dataset. This avoids the desaturated output of regression methods

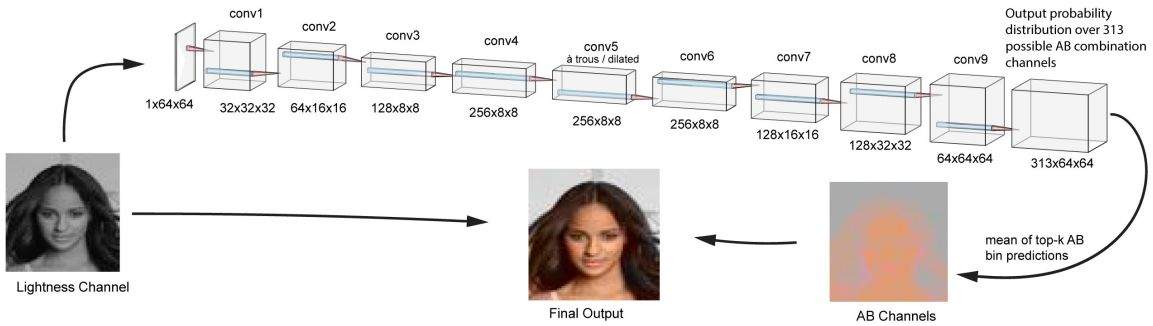

Figure 1. Model architecture visualization. Each convolution block consists of a series of 2d-convolutions followed by ReLU and Batch-norms for the respective input/output depths. The output is converted back to the AB color space and concatenated with the L-channel for the final output.

and non-weighted methods and results in a wider range of bins used in the model outputs.

2.2. Architecture

We adapt Zhang et al.'s model architecture [4] for use on 64x64 images, with the full model specification given above in Figure 1. Downsampling of the width and height dimensions in the network occur through the final 2d-convolution with stride=2 of the block. While this largely follows the original paper's specifications, there are a few important changes. In particular, we reduce the depth of the network outputs at each layer and remove one of the atrous convolution blocks to reduce the amount of trainable parameters in the model. Atrous convolutions are used to increase the receptive field of the network without modifying the spatial dimensions [5]; however, we removed one of the atrous convolutions since we were working with low resolution inputs. The final difference between Zhang et al.'s model and ours' is an additional two transpose convolution blocks at the end to upsample the output back to the original height and width.

The model takes in the L (lightness) channel, from an image in the LAB color space. The output of the model is a 313 channel output, with each channel corresponding to a discrete bin as described in the previous section. The output therefore contains probability logits for the likelihood of each AB bin. We use these output logits during training for the loss computation. At inference time however, we need to recover the AB channels from the model output to produce the full LAB image. At this stage, the top-k values are chosen channel-wise. This represents the indices of the k most likely bins that the model predicts for each given pixel. We then convert each of the top-k bins into their corresponding AB values and take the mean of the k AB values. While the Zhang et al. used an annealed-mean, we found the top-k bin conversion to be capable of producing colorful results. Our recommended value for the final model is a value of $k = 2$, which is further explored in the experimentation section.

2.3. Loss

As mentioned in our methodology, we compute weights based on the prior of the bin distributions in the dataset. We denote per-pixel weight as v, which is computed from the weight of the highest predicted bin, shown in Eq 2. We then utilize those weights in a weighted cross-entropy loss where the per-pixel contribution to the loss is weighted by the correct class, as shown below in Eq. 3. Both equations are from Zhang et Al.'s loss formulation where Z is the ground-truth bins for each pixel and $\hat{Z}$ are the model predictions. [4].

$$v(Z_{h,w}) = w_{q^*} \quad \text{where } q^* = \arg\max_q Z_{h,w,q} \tag{2}$$

$$L_{cl}(\hat{Z}, Z) = -\sum_{h,w} v(Z_{h,w}) \sum_q Z_{h,w,q} log(\hat{Z}_{h,w,q}) \tag{3}$$

This follows the idea behind Zhang et al.'s classification loss as it allows for a weighting term to rebalance the cross-entropy classification loss. We find similar results in our datasets with respect to the color distribution. Desaturated colors are far more common than saturated ones, so using a higher loss weighting term for uncommon classes, or bins, encourages the model to choose saturated colors for the probability outputs. On the other hand, the model outputs should remain realistic in the amount of rarer colors it uses. The model begins to heavily favor the rare colors if only the weighted cross-entropy loss is used. This in turn leads to oversaturated images that have strange colorization, which we describe in-depth in the following experiments section. As a result, we remedy this by introducing our own loss component.

We formulate our additional loss component, which we call the distribution loss, to penalize deviations in the distribution of color between the dataset and the produced images. This helps constrain the model into not overusing the

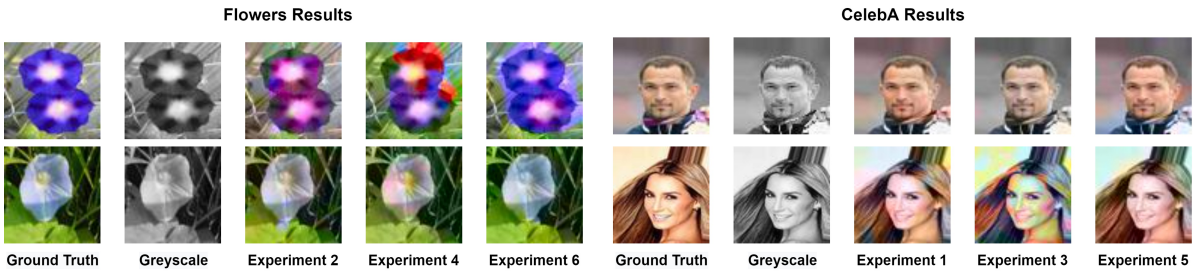

Figure 2. Visualized RGB results from the model output for each experiment. It can be observed most prominently with the flowers that the saturation increases when we add class rebalancing weights. The experiment with the additional distribution loss term helps avoid unnecessary rare colors in the output.

rare, saturated colors. To do so, we compute the sum of the predicted bin values for each bin into $B \in R^Q$, where B_i is the i^{th} bin. This is used to form a probability distribution of output predictions for each bin over the entire batch of size N outputs, given below in Eq.4.

$$B = \frac{B_i}{||B_i||} \quad \text{where } B_i = \sum_{N,w,h} \hat{Z}_{h,w,q} \tag{4}$$

Since the distributions are discrete, we measure the difference in distribution with the L2 difference between our computed output distribution B and the dataset priors $\tilde{p}$, which is shown below in Eq.5.

$$L_{dt} = ||B - \tilde{p}||_2 \tag{5}$$

Finally, our group combines the two losses into the overall objective loss function that the network uses to train, given by Eq. 6. We fixed the value of α to 0.4 in training. Our distribution loss acts as an adaptive training penalty that penalizes based on the model output per batch. The further the output distribution strays from the dataset during training, the higher the distribution loss will be. Intuitively, the discrete distribution of the output prediction logits across the batch should be similar to the set of prior probabilities for each bin in the training dataset. This helps enforce the model to continue to produce representative distributions of colors while encouraging the model to correctly predict colors with high rarity.

$$L_{obj} = \alpha L_{dt} * (1 - \alpha) L_{cl} \tag{6}$$

3. Experiments

3.1. Loss Comparisons

In order to get an accurate depiction of how our model performs, we decided to perform a series of comparison studies to measure how the network performs without the model weights and loss additions. For all the experiments, we use an ADAM optimizer, fix the learning rate to $2e^{-4}$,

Experiment	Weighting	Loss	Dataset	PSNR	SSIM
1	None	L_{cl} only	Celeb-A	23.26848	0.873627
2	None	L_{cl} only	Flowers	19.94664	0.7724886
3	Eq. 1	L_{cl} only	Celeb-A	21.08575	0.809523
4	Eq. 1	L_{cl} only	Flowers	19.509997	0.7613506
5	Eq. 1	L_{obj}	Celeb-A	26.6429	0.9072
6	Eq. 1	L_{obj}	Flowers	19.8056	0.7702

Table 1. Details of each experiment run with the specific weighting, loss, and performance metrics.

train for 100 epochs, and reconstruct the AB image channels from the maximum bin prediction (reconstruct with the top-1 bin). We chose to fix the hyperparameters and model design throughout the experimentation so we could observe the effects of changing the loss formulations. The specifics of each experiment ID can be seen in Table 1. The two datasets we chose to analyze were a dataset of celebrity faces [6] and on a dataset of colorful flowers [7]. These two datasets were chosen since there was a salient object to colorize (either a person or a flower) in the foreground, that we could compare colorization results on. Finally, experiments 1 and 2 serve as the baseline since it is the simplest approach with no weighting scheme or distribution loss.

To assess our performance with quantifiable metrics, we used two metrics: structural similarity index measure (SSIM) and peak signal-to-noise ratio (PSNR). Both of these measures are used to assess the similarity between two images, with higher scores representing a closer similarity. These scores across our six experiments are also shown in Table 1. While Zhang et al. achieved high-quality results with the weighting scheme used in Experiment 3 and 4, we found the introduction of weighted loss to result in poorer performance relative to the baseline. This could be due to a variety of reasons, since we use a modified simpler model architecture, train for far less (1-2 hours instead of days), and don't employ their annealed mean reconstruction that can be found in their paper [4]. With the experiments using our revised objective loss function however, we were able to get better scores for Celeb-A dataset and similar scores for the flowers dataset compared to the baseline.

We look at the qualitative results, as displayed in Figure 2, to further compare the model performance. We suspect the drop in quantitative performance was due to the nature of the scheme, since the rarity of the color is inversely proportional to its weight. The rare colors end up with the highest weights; thus, the model largely penalizes incorrect classification of these rare colors. Rather than risk incorrectly classifying a rare color when it appears in the ground truth, the model could learn to use rare colors liberally. Consequently, this also means that incorrectly predicting a false positive for a rare color on a ground truth pixel of a common color will contribute minimally to the loss since the common color carries a low weight. As seen in Figure 2, this causes splotches of incorrect rare colors throughout the images in experiments 3 and 4. These experiments with only weighted cross entropy loss tended to produce saturated, but often incorrect results.

We chose the use of L_{obj} (experiments 5 and 6) as the best after considering both its quantitative and qualitative performance. The introduction of the distribution loss helps alleviate this issue because it incurs a penalty for overpredicting rare colors, since this would cause the overall output distribution of prediction logits to skew away from the training distribution. As noted in Figure 2, this allows the colors to remain fairly saturated, but also appear more consistent with the ground truth. Overall, the model choice is able to achieve the highest PSNR/SSIM scores on Celeb-A and while it has a slight drop in performance for the flowers dataset with respect to the baseline, the results tend to choose more vibrant colors, verifying our choice.

3.2. Top-k Bin Selection

Once we had found the objective loss function we wanted to use, we decided to further experiment with the image reconstruction methodology. More specifically, we varied the value of k when selecting the top-k bins using the model trained from experiment 5. Since this reconstructs the original AB channels from the model output, there was no further training needed and we fixed the model 5 weights throughout this experiment. The images shown in Figure 3 show how the images vary across sampling different top-k bin sizes. In both examples, it can be observed that as the amount of bins averaged increase, the images become more saturated. Other predictions that are not the top prediction might favor more saturated colors, thus causing the mean of the top-k predictions to result in a saturated image.

To determine the best selection of k, we measured the SSIM and PSNR scores on the test set for each setting of k from $k = 1, ..., 5$. Additionally, we conducted a qualitative study. We presented 20 ground truth images and their corresponding 5 images for a total of 100 output images to team members and outside individuals. The task given was to order the images from best to worst, where the score for the image was their index in this sorted order (1=best, 5=worst). The final result reported is the average score for each setting of k, with lower being better, displayed in Table 2.

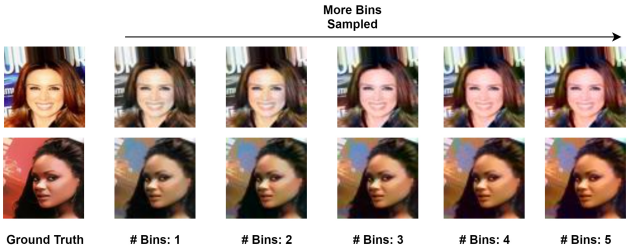

Figure 3. The effect of the top-k bins reconstruction for the AB channels can be seen as the value of k increases. The output image color becomes more saturated but increasingly deviates from the original.

	#Bins 1	#Bins 2	#Bins 3	#Bins 4	#Bins 5
Avg Rating:	2.45	2.25	2.05	3.8	4.45
SSIM	0.9072	0.8973	0.8806	0.8618	0.8432
PSNR	26.6429	25.8000	24.6880	23.6487	22.7339

Table 2. The qualitative user ratings and the quantitiative metrics for each setting of k for the top-k bins.

The SSIM and PSNR scores decrease as we increase the amount of bins sampled, which implies that the image begins to further deviate from the original image as we sample more bins. While quantitatively the best result occurs at $k = 1$, we note that the colors visually appear more saturated as k increases which is reflected by the best qualitative score occuring at occurring at $k = 3$. We therefore chose $k = 2$ as a compromise between the quantitative and qualitative metrics. This minimizes the drop in SSIM and PSNR score and is also rated well in our qualitative study.

4. Implementation

The model architecture is inspired by Zhang et Al.'s model [4] where we adapted it to a 64x64 resolution. We implemented the paper mostly from scratch, with the quantization, loss calculations, and training infrastructure. We also made the following few key changes: we use naive quantization at each pixel value instead of a Gaussian weighting, we add our own loss component, and we use a top-k mean instead of the annealed mean to recover the AB channels from the model output. For borrowed code, we took the .npy files containing the 313 AB bin and priors matrices from the author's Github [1]. To ensure that our weighting values were correct, we also borrowed the priors to weight conversion from a pytorch implementation of the paper [2]. Finally, to load the data as a grayscale and quantized image at each training step, we built on top of an existing image folder implementation [3].

[1] https://github.com/richzhang/colorization
[2] https://github.com/Time0o/colorful-colorization
[3] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

References

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[2] Domonkos Varga and Tamás Szirányi. Fully automatic image colorization based on convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3691–3696. IEEE, 2016.

[3] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. in: Proceedings of the ieee international conference on computer vision. pages 415–423, 2015.

[4] R. Zhang, J. Zhu, I. Phillip, X. Geng, A. Lin, T. Yu, and A. Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017.

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. December 2015.

[7] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.