# RainbowNet: Color Extrapolation from Grayscale Images

**Emily Ling**
Computer Science
Stanford University
eling8@stanford.edu

**Cindy Lin**
Computer Science
Stanford University
cinlin@stanford.edu

**Owen Wang**
Computer Science
Stanford University
ojwang@stanford.edu

## Abstract

Inspired by digital recoloring of historical photos and Zhang et al.[1], our project aims to train a convolutional neural network to colorize black-and-white images, optionally augmenting with embeddings obtained from Google's Inception ResNet v2 (IRN2)[2]. Given a batch of RGB images, we generate a possible colorization by training and predicting in the CIE Lab color space. We additionally analyze our training data to examine which distributions of colors most often occur together via K-clustering. Our model predicts mostly browns, blues, and reds and performed better without the added complexity of the IRN2 fusion.

## 1 Introduction

Traditionally, the coloring of black-and-white historical photos is as much art as it is functional. Prints could take as long as a month to recolor as artists researched the colors on relevant fabrics, skin tones, and objects with the goal of bringing history back to life for contemporary viewers, who might otherwise feel detached from old black-and-white photographs. Groups such as ColorizedHistory[3] bring together dozens of semi-professional colorizers for this task, and Time magazine has commissioned colorizers to produce a gallery of colored photographs of Abraham Lincoln in the hopes of creating a ”vibrant documentation” of the historical figure.[4]

We would like to explore how well a deep learning approach performs on a task that takes human colorizers months to accomplish. With grayscale images as inputs, we use a convolutional neural network to output a predicted color per pixel. Our goal is to generate a plausible (and hopefully unique and interesting!) colorization to observers.

## 2 Related Work

To begin, we considered papers that were the first to introduce deep learning to this problem space. Cheng et al.[5] proposed a fully automatic approach to grayscale image colorization involving the extraction of various features and the development of small neural networks for different patches of a single image. Iizuka et al.[6] further explored the combination of global and local image features as inputs to their CNN. Here, they use the global features to indicate the type of image (eg. indoor vs. outdoor) and the local features to represent the more minute details of the image (eg. grass, furniture). Iizuka et al. fuse the global and local features together depth-wise, as shown in the image below in section 4.2. Baldassarre et al.[7] augmented Iizuka's fusion by replacing the global features with embeddings retrieved from Google's Inception ResNet 2[2], which we similarly replicated in our project.

Our main reference for this project was from "Colorful Image Colorization" (Richard Zhang, Phillip Isola, Alexei E. Efros). Zhang et al. trained an encoder and decoder CNN to predict colorizations for grayscale images; this paper has two novel caveats to its approach. One was that instead of using the MSE loss and predicting two color channels in their output volume, they chose to discretize their color state space into bins and transform the colorizing task into a (somewhat harder to learn) classification problem. From their experiments, this leads to results with more vibrant colors, as the model is discouraged from being biased towards predictions with low *a* and *b* values. The second caveat concerns their loss function. The empirical color frequency distributions in their training set had many orders more desaturated colors (browns, grays) than ones like red, green, or yellow for example. To encourage the network to learn rarer colors, the softmax loss was multiplied by a weight for every pixel that weighted dull colors with a greater loss. One weakness of this model was its bias for red and blue hues - in uncertain conditions, e.g. dark lighting at night or complicated backgrounds, the local colors would seemingly default to red.

# 3 Dataset and Features

## 3.1 Lab Color Space

For inputs into our CNN, we represent all images in the CIE Lab color space, which contains three channels: L for lightness and a and b for the color components green-red and blue-yellow. This allows for easy separation of black-and-white from color information, allowing us to use the L channel as input and ab channels as the desired output.

## 3.2 *ab* Channel Discretization

We additionally discretize the ab space into 112 buckets and use bucket indices as the desired output instead of raw ab values during training. Our main motivation for this bucketing schema is due to the nature of loss functions based on distance between the predicted and true label. Because these functions are best minimized by mid-range values for a and b, minimizing the loss function does not provide the vibrancy in color we seek in our predictions. Furthermore, since the RGB color space is contained entirely within the CIE Lab color space, we may predict an Lab color that is outside of the possible set of RGB colors, giving an invalid result. Thus, we need some way of restricting the possible output set to only the CIE Lab colors that have a valid RGB mapping. Instead of directly predicting values for a, b, we divide the ab output space into buckets, and compute a probability distribution over the set of discrete colors. We chose to quantize the ab output space, where we limit the range of a and b from -128 to 127, into buckets with grid size of 16, excluding buckets that lie outside the RGB color space. This produced a total of 112 buckets. Using a set of discrete colors allows us to treat this problem as a multinomial classification problem, where for each pixel, we predict a probability distribution over the 112 buckets and use the bucket with the highest probability. These 112 buckets are plotted in CIE Lab color space in figure 1, with L=50.

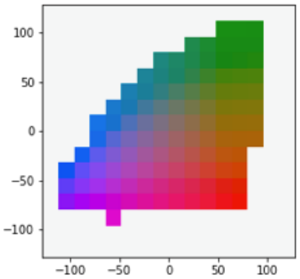

Figure 1: Plot of 112 buckets resulting from discretizing the ab space

## 3.3 Preprocessing

Our dataset contains approximately 10K images from Unsplash[8], which are broken into 9294 training, 247 validation, and 247 test examples. The images are mostly 256x256 portraits and are preprocessed as follows:

1. 256x256 RGB images resized to 224x224
2. RGB images converted to grayscale and fed into IRN2 to obtain an embedding per image
3. RGB images normalized to [0, 1] and converted to Lab color
4. Lab images separated into L (224,224,1) and ab (224,224,2) channels
5. for every pixel, generate a logit of the bucket index that the $(a, b)$ values belong to: size (224, 224, 1)

As a result, we obtain the L channel (224, 224, 1) and IRN2 embeddings (1000,) as our inputs and the discretized ab channels (224, 224, 1) as the supervisory signal / ground truth.

# 4 Methods

## 4.1 v0: Baseline

Our baseline is trained on a toy dataset of size 40 using raw ab values, not buckets, and consists of one encoder and one decoder both using ReLU activation. The result, as seen in figure 2, is a drastically desaturated image, as the loss function trained with raw ab values encourage conservative color predictions.

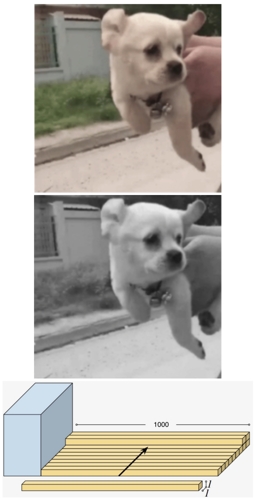

Figure 2:
Top: ground truth
middle: baseline prediction
bottom: depth-wise fusion of encoder output (blue block) with IRN2 embeddings (a single yellow strip of dimensions (1000,)

## 4.2 v1: CNN with IRN2 Fusion

v1 contains three main components:

1. Encoder block: two convolutional layers followed by BatchNorm
2. Fusion layers: combination of encoder output and IRN2 embeddings as described in Iizuka and Simo-Serra et al.[6] and depicted in figure 2
3. Decoder block: two convolutional layers followed by BatchNorm, and 2x upsampling

The model has four encoder blocks with filters of sizes 64, 128, 256, 512. The fusion appends our Inception Resnet v2 embeddings depth-wise with our encoded volume, and then there are four decoder blocks with filters of size 512, 256, 128, 64. We add one more convolutional layer of 128 filters, to ease the scaling up to a 112. The final layer is a convolution with softmax activation of 112 filters. Our final output volume is of shape (224, 224, 112), and interpreted as a softmax over the 112 color buckets for every pixel.

The function for convolutional layers per-pixel is written as the following, where $W$ is a shared matrix of weights, the $k$'s are the kernel height and width, and $x$, and $y$ are the pixel component of the input and output, respectively.

$$y_{u,v} = \sigma\Big(b + \sum_{i=-k'_h}^{k'_h} \sum_{j=-k'_w}^{k'_w} W_{k'_h+i,k'_w+j} x_{u+i,v+j}\Big)$$

$$k'_h = \frac{k_h - 1}{2}, k'_w = \frac{k_w - 1}{2}$$

## 4.3 v2: CNN without IRN2 Fusion

v2 is similar to v1, except it does not involve fusion with the IRN2 embeddings nor the BatchNorm layers, retaining only the encoder and decoder components. After training on v1, we decided to remove IRN2 embeddings in an attempt to decrease the complexity of the model and possibly speed up training. The architecture of this neural network is shown in figure 3.

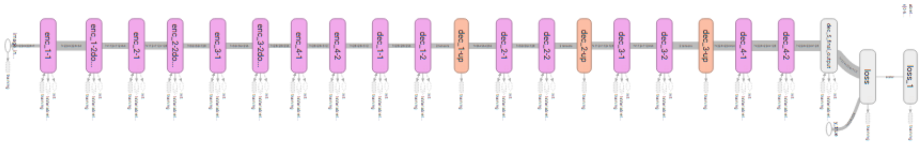

Figure 3: Architecture of CNN without IRN2 fusion

# 5 Results

## 5.1 CNN without IRN2 Fusion

Because our CNN with IRN2 fusion had difficulty training given our limited time and resources, we focused on the simpler CNN without the fusion layer described in the section above. Our model trained for 23 epochs, with the training loss decreasing from about 4.3M to about 2.5M by the end. Figures 5, 6, and 7 display this model's predictions on test set and training set images. Figure 5 demonstrates that our model was able to identify some features such as sky and skin with limited success, although the images remained largely unsaturated with some patches of color. For example, in the first column of this figure, the model successfully identified colors of the sky and horse. Figure 6 displays some selected images from the training set. The predictions displayed here are fairly close in color to the ground truth images, demonstrating that our model was indeed making progress during training. Figure 7 displays selected training set images in which the predicted colors differed significantly from the original. While some inaccurate predictions involved different but perhaps still plausible colors (e.g. the second and third columns of the figure), many predicted images contained desaturated colors and/or splotches of color that were not aligned with edges.

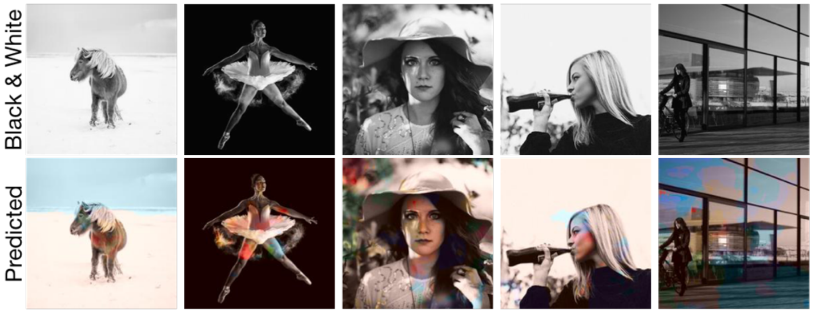

Figure 4: Examples of predictions on test set images

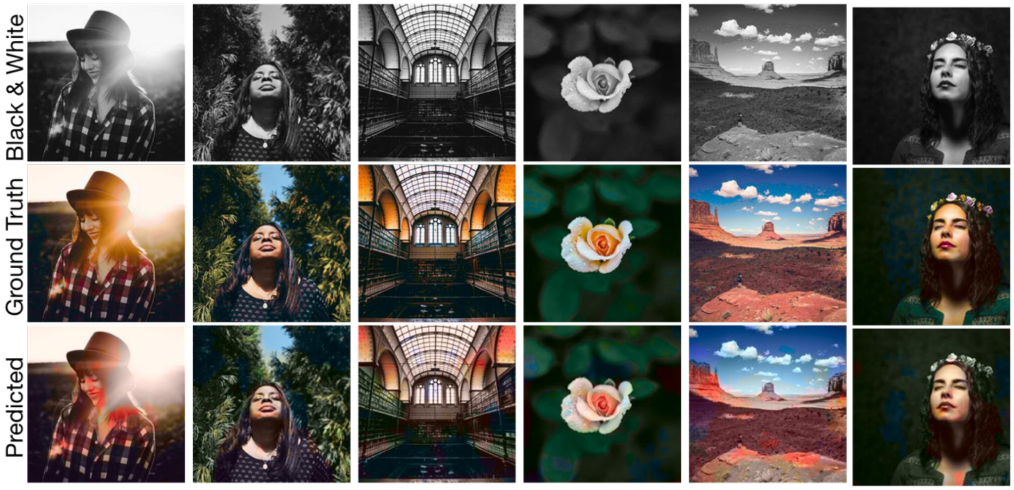

Figure 5: Examples of high-accuracy predictions on training set images

# 6 Discussion

## 6.1 K-clustering

To better visualize our data, we used K-clustering on our training set, representing each image by a histogram of the number of times each color bucket appears, then averaging the counts for each cluster. Some buckets are visualized below, with all buckets with non-zero frequency filled with color.

The most prevalent colors in our dataset tended towards brownish, desaturated colors, which might explain our model's propensity towards predicting sepia colorizations.

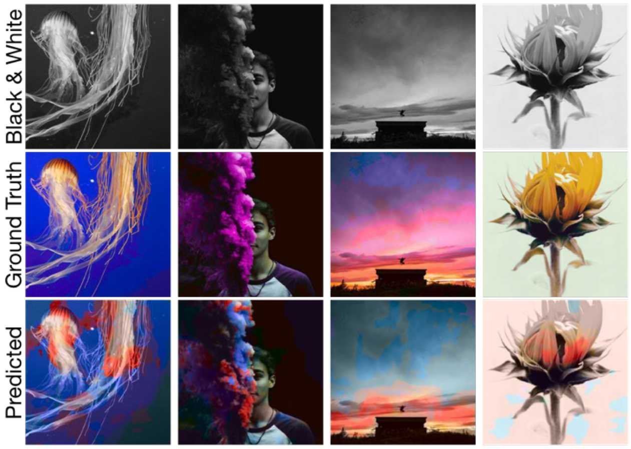

Figure 6: Examples of low-accuracy predictions on training set images

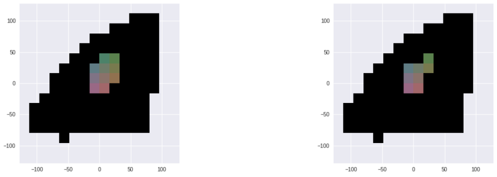

Figure 7: Average histograms for two sample clusters

### 6.2 Training time

Our final model's training time was around 20 hours of GPU time (for 23 epochs), and 35 for the network with the fusion layer. The final model ended with the loss is around 2.5M. Our network would have benefited from much longer training time, which was regretably difficult to achieve given our development speed and time frame for the project.

### 6.3 IRN2 fusion

Given our training time, fusion with the IRN2 embeddings might have been too complex for our model, thus producing poorer results compared to the unfused model. Given results from past papers like Baldassarre et al.[7], we do expect these embeddings to provide useful global context to the image that is not capturable using encoders.

## 7 Future Work

Even with our current bucketing schema, we find that the majority of our predictions still were quite sepia-toned. To further dissuade the model from predicting neutral colors, we can weight different colors in our pixel-wise cross entropy loss such that predicting rare, saturated colors are penalized less than predicting neutral colors. This technique was attempted in Zhang et al.[1] with some success.

To extend the problem space, we have considered applying the colorization technique to sequences of images, as in a short video or gif. We hypothesize that feeding in stacks of n consecutive frames into the model would produce a more stable colorization when compared frame-by-frame and would be more efficient from a training perspective than a RNN-CNN combination alternative. To further speed up training by reducing our output space from the current softmax loss, we could also consider a generative adversarial network (GAN).

## 8 Contributions

*Emily Ling*: color bucketing, data processing, generating predicted images and figures, general debugging, poster

*Cindy Lin*: baseline, milestone writeup, customized colorization loss, K-clustering, final paper

*Owen Wang*: Rainbownet architecture, debugging of loss and training of model, data flow from (relatively) large train set, final paper

## References

[1] Zhang, R, Isola, P, Efros, A (2016). Colorful Image Colorization. University of California, Berkeley. Retrieved from

`https://arxiv.org/pdf/1603.08511.pdf`

[2] Improving Inception and Image Classification in TensorFlow, Research Blog, 31-Aug-2016. [Online]. Available:

`https://research.googleblog.com/2016/08/improving-inception-and-image.html`

[3] History in Color - ColorizedHistory, reddit. [Online]. Available:

`https://www.reddit.com/r/ColorizedHistory/`

[4] Modernized President: Portraits of Abraham Lincoln, In Color, Time. [Online]. Available:

`http://time.com/3792700/a-vibrant-past-colorizing-the-archives-of-history/`

[5] Cheng, Z, Yang, Q, Sheng, B (2016). Deep Colorization. Retrieved from

`https://arxiv.org/pdf/1605.00075.pdf`

[6] S. Iizuka and E. Simo-Serra, Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. [Online]. Available:

`http://hi.cs.waseda.ac.jp/~iizuka/projects/colorization/data/colorization_sig2016.pdf`

[7] Baldassarre, f, Morin, D, Guirao, L (2017). Deep Koalarization. Retrieved from

`https://github.com/baldassarreFe/deep-koalarization`

[8] FloydHub - Deep Learning Platform - Cloud GPU. [Online]. Available:

`https://www.floydhub.com/emilwallner/datasets/colornet`