# A convolutional classification approach to colorization

Vincent Billaut
Department of Statistics
Stanford University
vbillaut@stanford.edu

Matthieu de Rochemonteix
Department of Statistics
Stanford University
mderoche@stanford.edu

Marc Thibault
ICME
Stanford University
marcthib@stanford.edu

## 1. Introduction

The problem of colorization is one that comes quickly to mind when thinking about interesting challenges involving pictural data. Namely, the goal is to build a model that takes the greyscale version of an image (or even an actual "black and white" picture) and outputs its colorized version, as close to the original as possible (or at least realistic, if the original is not in colors). One clear upside to this challenge is that any computer vision dataset, and even any image bank really, is a proper dataset for the colorization problem (the image itself is the model's expected output, and its greyscale version is the input to the model).

Classical approaches to this task, *e.g*. [2] and [3], aim at predicting an image as close as possible to the ground truth, and notably make use of a simple $L_2$ loss, which penalizes predictions that fall overall too far from the ground truth. As a consequence, the models trained following such methods usually tend to be very conservative, and to give desaturated, pale results. On the contrary, authors of [6] take another angle and set their objective to be "*plausible* colorization" (and not necessarily *accurate*), which they validate with a large-scale human trial.

Our goal is to reproduce the approach of [6], as we consider the implementation more challenging in the way the loss function and the prediction mechanism are designed, and the results more visually appealing.

If we reach satisfactory results in a timely manner, we will consider tackling the task of colorizing videos, for which, in order to fully take advantage of the input format, will need to incorporate the notion of consistency between consecutive frames in a sequence.

## 2. Problem statement

### 2.1. Datasets

As previously stated, any dataset is proper for the colorization task, and we are currently working with subsets of both the SUN dataset [5] and ImageNet [4].

More precisely, we are tackling a simplified version of our general task: correctly colorizing scenes that involve beaches and seashores. The idea behind this simplification is that restricting our model to rather consistent scenes, where we typically find the same patterns (the top of the scene is usually the sky, *i.e*. blue, and the bottom usually sand, *i.e*. yellow or brown, as the example of Figure 1 shows), will enable us to have a working proof of concept, on top of which we can extend to bigger sets later. The "`beach_sun`", "`coast`" and "`sandbar`" categories of the SUN database contain 211 pictures. The "`beach`", "`coast`" and "`seashore`" categories of ImageNet respectively contain 1,713, 321 and 2,382 pictures. This amounts to a total of 4,627 pictures, which doesn't appear to be much, but might be enough considering the reduction of our problem that we're focusing on.

Figure 1. Example of a picture from our reduced datasets (in this case, from `ImageNet/beach`)

Upon loading of an image, we switch the encoding from the RGB format to the YUV format, which means that the first dimension of every pixel encodes its grayscale value (the *luminance*) and the two other encode the color itself (the *chrominance*). This trick aims at simplifying the problem, because now the input is just one dimension of the image, and the expected output the two other dimensions.

### 2.2. Expected results and evaluation

The final objective of this project is ideally to be able to recolorize legacy grayscale pictures. For this milestone, our objective is to be able to recolorize seashore pictures in a plausible way. The purpose is not to have the exact reconstitution of the initial image, but rather to output an image that *looks* true. To evaluate this, our final evaluation metric will be the qualitative aspect of the colorized pictures, as it is difficult to numerically determine if the coloring of an image is satisfying.

Our current focus is to find a network that is efficient at solving the classification task that we use to "encode" the colorization problem, the loss being our main metric (Section 3 gives more details on this). As we are writing this milestone, we want, as a sanity check, to confirm that we are able to overfit the dataset with a complex model and are therefore solely working on the training set. We will only look at the classification loss, and not use validation. To check this, we tune the hyper-parameters so that the network has a consistent behavior in terms of loss and actually colorizes images from the training set. This will serve as a baseline to fine-tune the loss definition and prediction procedure, and to sophisticate the structure of the network.

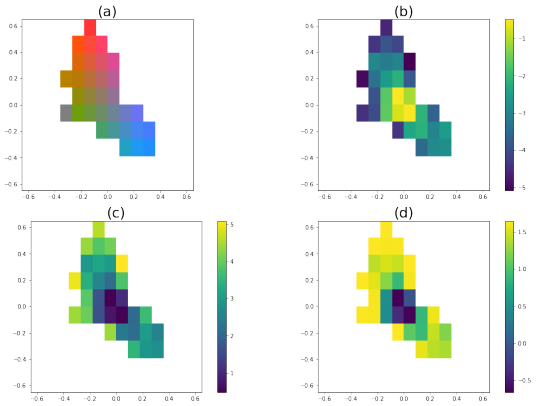


Figure 2. (a) Color map, showing the mean color (*chrominance*) of each of the selected bins (here, we set a threshold of 32 bins to select). (b) Frequency map (log-scale), shows the empirical frequency of the colors within each bin, computed over a dataset beforehand. (c) Inverse-frequency map (log-scale), *i.e.* the inverse of (b). (d) Weight map (log-scale), shows the weights assigned to each bin after rebalancing. Interestingly, we notice that the amplitude in weights is much smaller than the amplitude in frequency (2 orders of magnitude against 4), which means that we partially make up for the underrepresentation bias and therefore encourage the prediction of rare colors.

## 3. Technical approach

### 3.1. Overview

The practical approaches to learning the color of a grayscale image can be split in two groups: the ones who take the angle of regression and try to predict a continuous color value, *e.g.* [2] and [3], and the ones who take an angle of classification, and discretize the color space into a finite amount of bins, *e.g.* [1] and [6]. Once again taking the angle of [6], we're approaching the colorization as a classification problem, and therefore using a (weighted) cross-entropy loss.

Concretely, we want to discretize our colorspace, and for that we simply split our colormap into equal sized bins. As a first step, in order to reduce the computational toll, we don't want too many bins, and are therefore restricting our discretization to the $n$ most frequent color bins, as learned on a big dataset beforehand. In what follows, if a color from our actual image does not fall within one of our $n$ bins, we will simply assign it to the closest available. This simplification leads to a rather faint degradation of the images.

Following the approach of [6], we want to boost the possibility of a rare color being predicted, and therefore reproduce the following tricks:

- Use *rebalancing* to give larger weights to rare colors in our loss function. Precisely, each pixel value $p$, assigned to its closest bin $b \in \mathbb{R}^n$ is given a weight $w_p$ such that

$$w_p \propto \left((1-\lambda)\widehat{P}(b) + \frac{\lambda}{n}\right)^{-1}$$

  where $\widehat{P}(b)$ is the estimated probability of the bin (computed prior to our model's training) and $\lambda \in [0, 1]$ a tuning parameter (the closer to 1, the less we take the bin's frequency into account).

- Use an *annealed-mean* to output predictions $y$ from the probability distribution $\mathbf{Z}$ over our $n$ bins to the full original color space. The idea is to find a compromise between taking the color class with the maximum probability (the mode), which gives a rather colorful result but sometimes lacking spatial consistency, and taking the weighted average over the bins, which gives a rather flat, sepia kind of tone. To achieve this we use a temperature parameter $T \in (0, 1]$ in the following softmax-like formula for one pixel

$$y = f_T(\mathbf{z}) = \frac{\exp(\log(\mathbf{z})/T)}{\sum_i \exp(\log(\mathbf{z}_i)/T)}$$

  where $\mathbf{z}$ is the $n$-dimensional probability vector of a given pixel over the $n$ bins, and the sum in the denominator is over all the bins.

Figure 2 shows our discretized colorspace, as well as what the weights of the bins look like after rebalancing.

| | |
|---|---|
| ConvBlock1 | CONV3-32(2), ReLU<br>CONV3-32(1), ReLU<br>MAXPOOL2, BATCHNORM |
| ConvBlock2 | CONV3-64(1), ReLU<br>CONV3-64(1), ReLU<br>MAXPOOL2, BATCHNORM |
| ConvBlock3 | CONV3-64(1), ReLU<br>CONV3-64(1), ReLU<br>MAXPOOL2, BATCHNORM |
| UpConvBlock1 | CONVT3-32(2), ReLU,<br>CONV3-32(1), ReLU<br>CONV3-32(1), ReLU |
| UpConvBlock2 | CONVT3-32(2), ReLU,<br>CONV3-32(1), ReLU<br>CONV3-32(1), ReLU |
| UpConvBlock3 | CONVT3-16(2) |
| OutputBlock | CONV1-$n$ |

Figure 3. Architecture of the network (CONVK-C(S) is a conv layer of size KxK with C channels and stride S.)

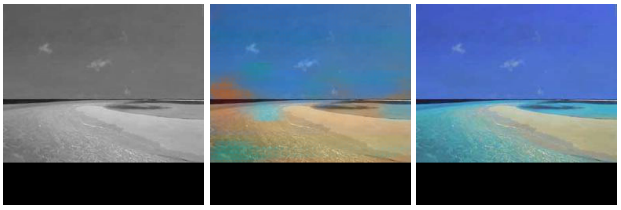

Figure 4. In-sample prediction after 100 epochs, on a complicated case. (left) grayscale input image. (center) prediction after 100 epochs. (right) groundtruth image.

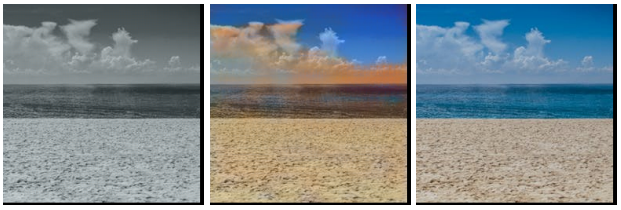

Figure 5. Out-of-sample prediction after 100 epochs. (left) grayscale input image. (center) prediction after 100 epochs. (right) groundtruth image.

### 3.2. Current model

Our current choice is to have a purely convolutional model (so that it can easily adapt to any image size if needed). The general idea is that we stack blocks of convolutional layers to scale down the image, progressively increasing the number of filters as the dimension decreases, then we upscale it to its original dimension using convolutional transpose layers. This is summed up in Figure 3.

## 4. Preliminary results

After training our model (100 epochs, learning rate of $2\ 10^{-3}$, batchsize of 64), we can see some overfitting occur (even though for complicated images, *e.g.* on Figure 4 where the sea is mixed up with the beach, the model still struggles a bit). Using this trained model to colorize an out-of-sample picture, we see that it has captured the notions of beach and sky (coloring the former in yellow tones and the latter in blue). As we notice (Figure 5), the model still has trouble handling the clouds and separating the sea from the rest, but this is definitely a good step.

## 5. Next steps

These results are encouraging as they show the robustness of the preliminary blocks we built, such as the color discretization and the learning and validating framework.

The next step of this project is to experiment with the network's composition. We want to test the influence of meta-parameters (layer size, regularization rate, dropout rate) on our learning performance. We expect to strengthen our results by optimizing our network this way. Next, we'll want to test new and more fancy network structures. Typically, a U-net could strengthen the gradient flow and help the upsampling layers. Our current model could also be seen as the baseline for the generator of a GAN-based approach. Finally, we'll want to expand our analysis to larger datasets. This step is crucial as it provides the perfect testing ground to quantify the ability of our model to generalize. If this performs well, we'll push towards colorizing videos, which should yield the most exciting results.

## References


[1] G. Charpiat, M. Hofmann, and B. Schölkopf. Automatic image colorization via multimodal predictions. In *European conference on computer vision*, pages 126–139. Springer, 2008.

[2] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[3] R. Dahl. Automatic colorization. `http://tinyclouds.org/colorize/`, 2016.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[5] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

[6] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.