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 [7] take another angle and set their objective to be "*plausible* colorization" (and not necessarily *accurate*). One of their methods to validate their results was to set up a blind testing experiment where human subjects had to determine, between their prediction and the ground truth, which one seemed more realistic (and they achieved very good performance).

Our goal is to reproduce the approach of [7], 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 satistfactory results in a timely manner, we will consider tackling the task of colorizing videos, 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.

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

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 [6] and ImageNet [5].

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 shown by the example of Figure 1), 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 ??? pictures. The "`beach`", "`coast`" and "`seashore`" categories of ImageNet respectively contain 1713, 321 and 2382 pictures. This amounts to a total of (only) (?? = 4416+???) pictures, which doesn't appear to be much, but might be enough considering the reduction of our problem that we're focusing on.

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

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], [3] and [4], and the ones who take an angle of classification, and discretize the color space into a finite amount of bins, *e.g.* [1] and [7]. Once again taking the angle of [7], we're approaching the colorization as a classification problem.

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.

[exemple avec le bateau ???]

Following the approach of [7], 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).

- Use an *annealed-mean* to output predictions y from the probability distribution $\mathbf{Z}$ over our n bins to the original, full 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 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.

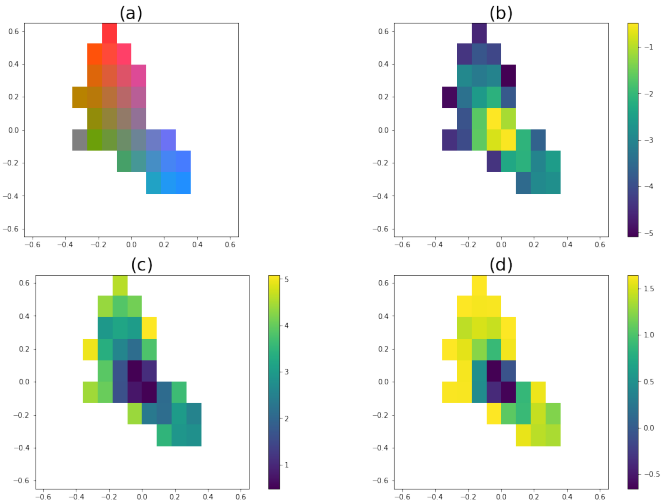

Figure 2. (a) Color map, showing the mean color (*chrominance*) of each of the selected bins (here, we set a threshold of 32 bins to select). (b) Frequency map (log-scale), shows the empirical frequency of the colors within each bin, computed over a dataset beforehand. (c) Inverse-frequency map (log-scale), *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.

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

3.2. Current model

4. Preliminary results

5. Next steps

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] A. Deshpande, J. Rock, and D. Forsyth. Learning large-scale automatic image colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 567–575, 2015.

[5] 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.

[6] 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.

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