Comparison of Different Loss Functions for Image Colorization

Anton Bothin
KTH Royal Institute of Technology
abothin@kth.se

Alex Norlin
KTH Royal Institute of Technology
alexnor@kth.se

Jakub Reha
KTH Royal Institute of Technology
jakubr@kth.se

Abstract

This project tackles the problem of colorizing gray scale images by treating it as a multinomial classification problem. This is done by training a deep convolutional neural network with an architecture and objective function first proposed by Zhang *et al.* [1]. While not achieving completely lifelike colored images we can clearly see that their method results in images with more vibrant colors than other objective functions could achieve. We also evaluated our model by looking at the percentage of correctly classified pixels for different thresholds and by testing if it improves image classification of gray scale images. The result for these experiments show that our models with the tweaked objective functions prove to be good choices for this task.

1 Introduction

When photography was invented in the 19th century it was an expensive and time consuming process that resulted in a black and white picture. It was not until over a hundred years later cameras capable of taking full-color pictures were starting to become the norm. Before that happened people experimented with coloring their black and white photographs by hand. Following in their steps it is not unreasonable to think that this task can be automated with the advent of advanced machine learning algorithms. However, where the colorizers of the 19th century had the luxury of often knowing the colors of objects in the photo by, for example, being there and seeing it, automatic coloring methods do not get any extra information over a black and white image and thus has to guess which colors objects are. In recent years trying to color images with the help of deep convolutional nets has become popular. These perform well for things that usually only have one color like grass or the sky. For things that lack this trait and come in a variety of different colors like cars and clothes the deep CNNs might not produce the right color. This is however, not that big of a problem, as the models usually produces a believable color for the object. If you were shown a photograph of a red sports car it would look like a normal photo even if the car in reality were yellow.

Another problem with using deep convolutional neural networks is that the objective function, or loss function, often is ill suited for the task. Loss functions such as the L2 norm favors a conservative color guess which often produce washed out, desaturated colored images. On the other hand, the advantage of this colorization task is that it does not require labeled data (self-supervised) and therefore might be easier to train and used as a pretext task for other types of problems. In this project we aim to produce high quality image colorizations using a deep convolutional architecture and a unique objective function and comparing it to some alternatives.

2 Related Work

There has been a lot of work done on the subject of image colorization with a variety of approaches. These approaches vary from treating it as an optimization problem [2; 3], utilizing Deep Convolutional Generative Adversarial Networks (DCGAN) [4], or training conditional Generative Adversarial Networks (cGAN) to generate colorized illustrations from black-and-white line art and color hints [5]. cGANs have also seen success in general image-to-image translation, which include colorization of gray scale images, but also allow other translations such as generating terrain maps from satellite imagery [6]. Most similar to our approach are [1; 7–9] which all more or less take the route of training convolutional neural networks. However, in [7] they use an alredy pretrained VGG16 network, therefore the system is not end-to-end trainable and it defeats the purpose to use colorization as a self-supervised pretext task.

3 Method

3.1 Data

In this project we chose to use a portion of ImageNet to train the model on. ImageNet is a data set composed of millions of pictures of different things like cats, cars, cannons, canoes and castles that has been used as training data for a variety of different models in the field of computer vision. Each of these classes contain, on average, 1000 pictures of their respective class. Since the whole of ImageNet is around 14 million pictures totalling at around 1.6 TB of data we could only train on a small part of it. We therefore constructed a data set only consisting of two classes, sport cars and passenger cars, in total 2512 images (after trying to train on the whole dataset and subsequently on 40k images, without signs of convergence). We used 400 for validation and the rest for training. Before feeding the images to the network they are converted to the Lab color space described below.

3.2 Lab color space

Of the many ways to represent an image, for this task, a natural choice of representation to use the Lab color space (or L*a*b*). It is natural because of its distinct channel for lightness **L** (which corresponds to black-and-white representation) while the two other channels are for encoding color. The **a** component determines the green-red color while the **b** component determines the blue-yellow color with red and yellow corresponding to positive values and green and blue to negative values. Another benefit of this representation is that distance between two colors in Lab space roughly corresponds to the relative perceptual difference between them.

There are other color spaces that share a similar properties with the Lab color space, HSL (standing for hue, saturation and lightness) being one of them. This representation does however lack the connection to perceived color that Lab space has while also introducing problems like H and S being unstable when L is either close to 0 or close to 1.

3.3 Objective Function & Class rebalancing

When trying to train neural networks to colorize gray scale images one of the problems is choosing an adequate objective function as can be seen from the many that have been tried [3; 7–10]. In this project we follow in the steps of Zhang *et al.* [1] and treat the problem as a multinomial classification problem. This also means that we have to quantize the *ab* space into different bins. This was done by mapping every RGB value to its Lab counterpart and dividing the result into different bins with a grid size of 10. In total we arrived at $Q = 322$ bins which is slightly different from the 313 bins they arrived at in their original paper. This has to be done as some pairs of (a, b) correspond of colors that are out of gamut. The bins were saved in KDTree structure for quick nearest neighbours lookup.

The objective function we arrive at is thus the one for multinomial cross entropy which takes $\hat{\mathbf{Z}} \in [0, 1]^{H,W,Q}, \mathbf{Z} \in [0, 1]^{H,W,Q}$ as input which are the models prediction and ground truth respectively where H, W are image dimensions and Q the number of quantized *ab* values. We calculate $\mathbb{Z}$ using a soft-encoding scheme where we find the nearest *ab* bin and its 5 nearest neighbors in the output space and weight them proportionally to the distance from the ground truth. This loss function is defined as:

$$L_{cl}(\hat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} v(\mathbf{Z}_{h,w}) \sum_{q} \mathbf{Z}_{h,w,q} \log(\hat{\mathbf{Z}}_{h,w,q}) \tag{1}$$

$$v\left(\mathbf{Z}_{h,w}\right) = \mathbf{w}_{q^*}, \text{ where } q^* = \arg\max_{q} \mathbf{Z}_{h,w,q}$$

where $v(\cdot)$ returns a weight used to rebalance the loss based on color rarity. This is because, as Zhang *et al.* pointed out in their paper most pixels contain desaturated colors and without accounting for this the model will learn to produce desaturated images. To counteract this we pre-compute weights $\mathbf{w}_{\mathbf{q}*}$ from the training set. Each pixel is then weighed by these depending on its closest *ab* bin during loss calculation. The weights $\mathbf{w} \in \mathbb{R}^Q$ were calculated from a smoothed empirical distribution, $\tilde{\mathbf{p}} \in \Delta^Q$, of colors in *ab* space according to equation 2.

$$\mathbf{w} \propto \left((1-\lambda)\, \tilde{\mathbf{p}} + \frac{\lambda}{Q} \right)^{-1}, \quad \mathbb{E}[\mathbf{w}] = \sum_{q} \tilde{\mathbf{p}}\mathbf{w} = 1 \tag{2}$$

3.4 Model

The architecture for the model is the same as the one by Zhang *et al.* [1] and a diagram of the model can be seen in figure 1. The numbers above each block is the number of channels in the convolutional layers of that block with the exception that for the input and output this number is at the bottom. The numbers under, at the end of each blocks is the spatial resolution of the output of the block. All changes in spatial resolution comes from the convolutional layer itself since there are no pooling layers in this network. The only time this isn't the case is in the last step where the output of the model is up-sampled to the original resolution.

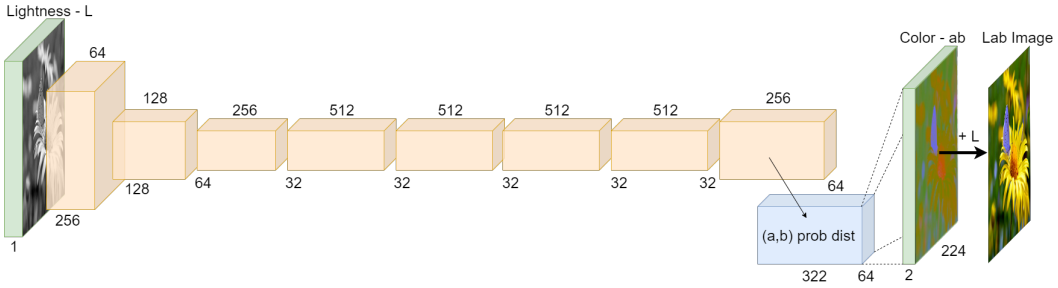

Figure 1: Model architecture. Each of the orange blocks corresponds to 2 or 3 convolutional layers with 1 batch normalization layer at the end of each block.

4 Experiments

In the experiments we test different versions of our trained model and compare the result to some different baselines. They are:

- **Ground truth:** The original image.
- **Gray:** The image has its (*a,b*) values set to 0.
- **Random:** The image has its (*a,b*) values switched with a random one in the dataset.
- **Ours (L2):** Our model trained with L2 loss.
- **Ours (class):** Our model trained without class rebalancing.
- **Ours (full, soft):** Our model trained with a soft encoding scheme.
- **Ours (full, one hot):** Our model trained with a one hot encoding scheme.

"**full**" refers to the full method with Cross Entropy Loss and Class rebalancing as described in the paper [1]. All models are trained from scratch and initialized with Pytorch default Kaiming He initialization in contrast to the k-means initialization in the original paper. Due to memory

constraints we train with batch size 8. We use Adam optimizer with default parameters $\beta_1 = .9, \beta_2 = .99,$ and weight decay $= 10^{-3}$. We train for around 15k update steps and do not use learning rate decay.

4.1 Raw Accuracy (AuC)

Although the model aims for plausibility, measuring the accuracy of predictions can still be useful. The raw accuracy was measured in the same manner as that used by the original paper [1]. This was done by, for each predicted pixel, measuring the L2 distance (in *ab* color space) between it and the ground truth. These distances were then compared to thresholds ranging from 0 to 150 to determine the percentage of correctly classified pixels per threshold, generating a cumulative mass function. To get the final accuracy, the area under the curve (AuC) was integrated and normalized.

An altered AuC measurement was also computed, using class rebalancing to put more weight on visually interesting regions (pixels with higher values of saturation), since these usually correspond to objects of focus. This was done by re-weighting each pixel according to Equation 2, using $\lambda = 0$.

4.2 Semantic interpretability (VGG classification)

There exist multiple Image classification models that try to predict the object in the image. It should be the case that our it is easier for said model to make its prediction on our models colored image rather than its gray scale counterpart. If the classifier performs well you could interpret it as the colored images looking more like its real life counterpart. We chose to test this with the VGG-16 model which has been pretrained on full color images from ImageNet.

5 Results

In figs. 2 to 6 we can see the qualitative results of our model with different loss functions on validation set images.

Figure 2: Ground Truth Images from the validation set

Figure 3: Output from our model trained with cross entropy loss, class rebalancing and soft encoding.

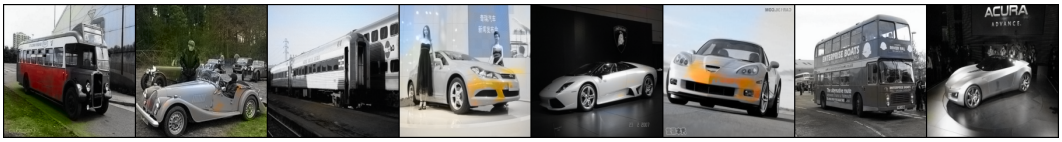

Figure 4: Output from our model trained with cross entropy loss and without class rebalancing (soft encoding).

Figure 5: Output from our model trained with L2 loss.

Figure 6: Output from our model trained with cross entropy loss, class rebalancing and one hot encoding.

In Table 1 we have the quantitative comparison of baselines and different loss functions. It seems reasonable that the unbalanced Cross Entropy performs the best within the non-rebalanced AuC metric whereas the rebalanced Cross Entropy performs the best within the rebalanced AuC metric. Also it is interesting that two of the loss functions outperform the ground truth in the VGG accuracy. This is likely because we trained our colorizing model on a very specific subset (cars classes) and therefore the colorizing model is boosting the attributes of these classes in the output images.

Table 1: Colorization results on the validation set of the two ImageNet classes (sports cars and passenger cars, 400 images).

Method	**AuC non-rebal (%)**	**AuC rebal (%)**	**VGG Top-1 Class Accuracy (%)**
Ground Truth	100	100	72.3
Gray	90.9	74.2	69.3
Random	88.5	77.6	58.0
Ours (L2)	90.4	74.0	70.8
Ours (class)	**91.9**	77.8	**73.0**
Ours (full, soft)	85.2	79.5	**73.0**
Ours (full, one hot)	84.0	**81.3**	72.0

In figure 7, we can see the results on other classes from the ImageNet. It seems the model learned to colour the sky and the grass quite well but it is obvious that it overfitted to the sports cars class where yellow and red cars are in almost every image.

Figure 7: Results on classes not seen during training

6 Discussion and Conclusions

Due to our limited time, computational power and reduced dataset we cannot compare our results with those from the original paper [1]. However, we can confirm the conclusion reached in the original paper, that is that their custom objective loss outperforms the L2 loss and simple Cross Entropy loss without rebalancing. There are also some peculiarities that are introduced due to us only having two classes in the dataset such as adding the color from one image to another only has other pictures of cars to choose from. This causes the random method to have quite high accuracy. There's also the fact that the accuracy of the VGG-16 network varies a lot between classes as some can have a very low correct classification rate while others have a disproportionately high one. This ties in to the reason our method produces slightly better results than the ground truth for the VGG classification evaluation. The classification accuracy for the gray scale images is high which suggests the VGG classifier has an easy time with the gray scale versions. Combine that with the our low number of validation images (400) and the obtained result could be explained by variance.

What we found out in addition to the original paper, is that soft encoding doesn't seem to have a big effect as it performs similarly to one hot encoding. Loading one hot encoding also completes in 70% of the time that it takes to load soft encoding with 5 nearest neighbours. In our setting, however, this is insignificant as the computation time bottleneck is the forward and backward pass of the training.

Due to the expensive nature of the training we couldn't test how certain aspects of the original paper's training method affect the performance of the model. In the future work, we would like to explore how significant is role of the k-means initialization and learning rate decay during training, as well as test out other architectures, namely the U-net or SegNet.

References

[1] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*, pp. 649–666, Springer, 2016.

[2] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph.*, vol. 23, p. 689–694, aug 2004.

[3] G. Charpiat, M. Hofmann, and B. Schölkopf, "Automatic image colorization via multimodal predictions," in *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, (Berlin, Heidelberg), p. 126–139, Springer-Verlag, 2008.

[4] K. Nazeri, E. Ng, and M. Ebrahimi, "Image colorization using generative adversarial networks," in *International conference on articulated motion and deformable objects*, pp. 85–94, Springer, 2018.

[5] Y. Ci, X. Ma, Z. Wang, H. Li, and Z. Luo, "User-guided deep anime line art colorization with conditional adversarial networks," in *Proceedings of the 26th ACM international conference on Multimedia*, ACM, oct 2018.

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[7] R. Dahl, "Automatic colorization." In: https://tinyclouds.org/colorize, 2016.

[8] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," European Conference on Computer Vision, 2016.

[9] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 415–423, 2015.

[10] A. Deshpande, J. Rock, and D. Forsyth, "Learning large-scale automatic image colorization," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 567–575, 2015.