Image Colorization With Convolutional Neural Networks

Marcus Lindström
marcul@kth.se

Joakim Dahl
joadahl@kth.se

Julia Ericson
juleri@kth.se

Abstract

Colorization of grayscale images is by no means a novelty, but has during the last years become increasingly popular due to fully automatic colorization schemes. In this report we investigate the plausibility of the images colorized with a Convolutional Neural Network (CNN) paired with an objective function pioneered by Zhang et al similar to Categorical Cross-Entropy (CCE), but also utilizing a method referred to as class rebalancing. In order to measure its efficiency, results from a network using regular CCE has been provided with an otherwise equal hyperparameter setup. Our results show (though rather subjectively) that slight improvements can be seen using Zhang's loss function as the images became more saturated. This produced images that often seem more plausible even though this is not always the case. A segment of the report has been devoted to potential changes of the network, where we propose using cyclical learning rate as well as the suggested improvement of the stochastic optimization method Adam called AMSgrad.

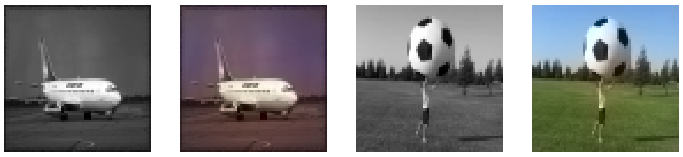

Figure 1: Grayscale inputs compared to our model's output using the loss function of Zhang et al. More results are shown in figure 5 and 6.

1 Introduction

Image colorization is the process of taking a grayscaled image and converting it into a colored image. The success of image colorization is difficult to measure since most objects, apart from surfaces such as sky, sea or forest, do not have a given color. Therefore, the evaluation of the network will be measured in terms of the plausibility of the colorized images, rather than how well they reflect the ground truth. By plausibility, we refer to having a realistic amount of saturation and color fields that are spatially consistent. To this date, Convolutional Neural Networks (CNN) is a conventional model choice for image handling. However, an apparent problem with the CNN approach for colorizing images is defining a loss function that satisfies the goal of producing plausible images. Zhang et al.(9) propose the usage of a loss function with a weightening term which will from here on be referred to as Class Rebalancing (CR). In CR, the loss of each pixel is re-weighed according to the rarity of the color. During training, pixels with rare, saturated colors are therefore empathized over the more common, desaturated colors that typically appears in the background. In this report, we will train

a CNN both with and without CR and compare the result based on the plausibility of the colored outputs.

2 Related work

Image colorization has been a hot topic for the last two decades. Typically, these can be categorized into the following:

1. Scribble-based methods
2. Transfer-based methods
3. Fully automatic colorization

The scribble-based methods requires manual help from the user in the form of scribbles. An example of this is the proposed algorithm by Levin et al. where the user helps the algorithm by making scribbles in the interior of a certain region, using the color that they presume that the region should have. The algorithm itself anticipates that the intensity of the gray-scale pixels to some extent also indicate the intensity of the coloring (5). An obvious disadvantage of this is the manual help that is needed, which gave birth to the transfer-based methods. These methods use reference images in order to predict images, extracting knowledge about the colors from the reference to the required prediction (7). Even though automation is implemented to some extent, it is still based on the fact that you have an arbitrarily similar reference image in your collection. To account for this issue, Fully automatic colorization was developed using neural networks, where a very large image set is used instead of a few relevant reference images for prediction. Different kinds of neural network approaches have been proposed. Three examples of the most prominent solutions are presented by Cheng et al. (1), Larsson et al. (4) and Zhang et al. (9).

3 Data

The network is trained on 100 000 images with 64×64 pixels from the benchmark dataset ImageNet, and validated on 1000 images from the ImageNet validation set. The training set is only a small subset of the total dataset consisting of 1.3M images, but as seen in figure 2, our dataset is distributed fairly even over the 1000 classes of ImageNet. Before training, the images were transformed from the RGB space to the Lab color space. The Lab color space contains three different channels; one light channel L corresponding to the brightness, and two color channels a and b. There are mainly two reasons for this transformation. Firstly, the L channel becomes a given input to the network as it only contains information about the light. Likewise, the a and b channels become given outputs. Secondly, the Lab color space is perceptually uniformed from a human perspective. Therefore, the euclidean distances between the colors of the Lab space have an intuitive interpretation (4). This property will be taken advantage of in the conversion of target-labels, described in the *Method* section.

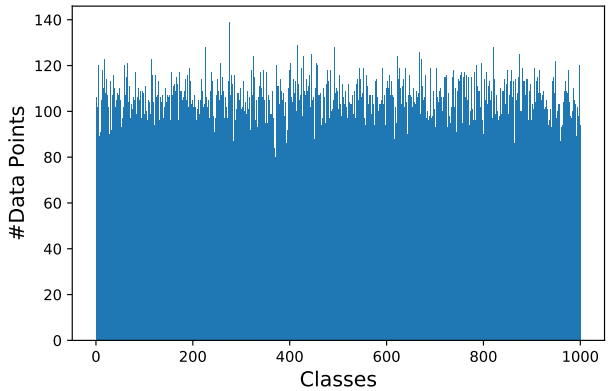

Figure 2: Class distribution of our subset of pictures from ImageNet

4 Method

4.1 Architecture

The architecture of our CNN is inspired by the one proposed by Zhang et al. However, since their model had input images of size 256×256 instead of 64×64, we reduced the size our network in order to avoid overfitting. Our network architecture can be seen in table 1.

Table 1: Network architecture, X is the spatial resolution of the output, C is the number of channels, S is the computation stride, D is the kernel dilation, B means that batch normalization is applied to the output, and US means that the input was up-sampled.

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
X	64	32	32	16	16	16	8	8	8	8	8	8	8	16	16	16
C	64	64	128	128	256	256	256	512	512	512	256	256	256	128	128	128
S	1	2	1	2	1	1	2	1	1	1	1	1	1	1	1	1
D	1	1	1	1	1	1	1	1	2	2	1	1	1	1	1	1
B		*Yes*		*Yes*			*Yes*			*Yes*			*Yes*			
US														*Yes*		

4.2 Specifications

During the optimization of the CNN, we had the most success when using the following setup:

- Cyclical learning rate: $\eta_{min} = 10^{-6}$, $\eta_{max} = 10^{-4}$ and stepsize $= 250$.
- AMSgrad optimizer: $\beta_1 = 0.90$ and $\beta_2 = 0.99$
- weight decay = 10^{-3}
- Xavier initialization of the weights.
- Dropout equivalent to 40% after each layer.

When the loss plateaued, we decided to lower the learning rate parameters according to $\eta_{min} = 10^{-7}$ and $\eta_{max} = 10^{-6}$. The code has been written in Python 3.7 using Keras. The training itself was performed on a Tesla k80 GPU on Google Colaboratory.

4.3 Loss function

An intuitive choice of loss function for a colorization problem is the euclidean norm. However, due to the multinomial nature of color distributions, the optimal solution of this problem will typically be the mean of the color space and the solution would turn out greyish. Instead, we redefine the problem as a classification problem with a categorical cross entropy loss function:

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

where $\hat{Z}$ is the output of the network and Z is the ground truth transformed into encoding vectors. Since the color space is continuous, Z cannot be described as one- hot encoded vectors. Thus, the color space in quantisized into 313 different colors and Z is described as soft encodings based on the distance between the actual pixel values and the quantisized color values. This transformation will be explained further in the subsection *Data transformations*. Unfortunately, the categorical cross entropy still suffers from desaturation. This is due to the fact that the unsaturated colors present in the sky or ground are often overly represented in the image data. Therefore, the network becomes

biased towards these colors. One way to avoid this, suggested by Zhang et al, is to weigh each pixel according to its color rarity. The weight vector is defined as:

$$\begin{aligned} \boldsymbol{w} &\propto \left((1-\lambda)\boldsymbol{p} + \frac{\lambda}{313}\right)^{-1}, \\ \mathbb{E}[\boldsymbol{w}] &= \sum_q \boldsymbol{w}\boldsymbol{p} = 1 \end{aligned} \tag{2}$$

where $\boldsymbol{p}$ is the prior probability of each of the 313 colors being the pixel value based on the colors of our dataset. λ is set to 0.5 as suggested by Zhang et al. The weighing factor for each pixel is thereby defined as:

$$v(Z_{w,h}) = w_{q*}, \quad q* = \text{argmax}_q Z_{w,h,q} \tag{3}$$

The loss function with CR is then defined as:

$$L_{cl}(\hat{Z}, Z) = -\sum_{w,h} v(Z_{w,h}) \sum_{w,h} Z_{w,h,q} \log\left(\hat{Z}_{w,h,q}\right). \tag{4}$$

Since the pixels with rare colors are given larger weights, the learning will be more focused on these pixels. This will reduce the network bias towards the unsaturated colors, and hopefully, the network will make less conservative guesses.

4.4 Data transformations

We define the ground truth outputs as $Y_{h,w,c}$ where H is the height of the image, W is the width of the image and $C = 2$ since there are two color channels. In order to match the output dimension of the network, cubic interpolation was used to reduce the size of the a and b channels by a factor of 4 such that $H = W = 16$. The resulting image is thereafter transformed in such a way that each pixel's *ab*-pair is mapped onto a probability distribution over $Q = 313$ quantisized *ab*-value bins depicted in figure 3, using the 5-nearest neighbours scheme in equation 5. The *ab* bins are stored in a vector $\boldsymbol{x} \in \mathbb{R}^{Q \times 2}$.

$$\begin{aligned} Z_{w,h,q} &= \frac{\exp\left(-\frac{d_q^2}{2\sigma^2}\right)}{\sum_q Z_{h,w,q}}, \quad \text{if } \boldsymbol{x}_q \text{ is a neighbour,} \\ Z_{w,h,q} &= 0, \quad \text{otherwise.} \end{aligned} \tag{5}$$

In equation 5, d is the euclidean distance to a neighbouring coordinate $\boldsymbol{x}_q$, and $\sigma = 5$.

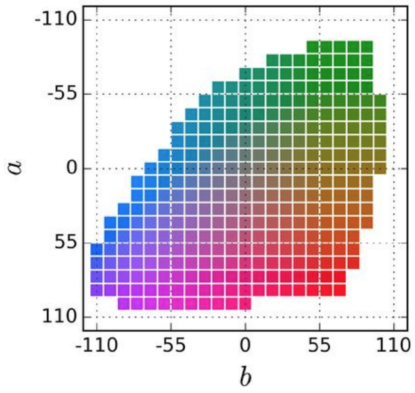

Figure 3: The 313 visible colors coordinates in the quantisized *ab* color space (9)

The softmax output $\hat{Z}$ of the network is transformed into the *ab* color space using a method called *annealed mean* designed by Zhang et al. The method is inspired by simulated annealing and is defined as the following:

$$
\begin{aligned}
f_T(\boldsymbol{z}) &= \frac{\exp(\log(\boldsymbol{z})/T)}{\sum_q \exp(\log(\boldsymbol{z}_q)/T}, \\
\hat{Y}_{h,w} &= \sum_q f_T(\hat{Z}_{h,w,q})\boldsymbol{x}_q.
\end{aligned}
\tag{6}
$$

Setting $T = 1$ is equivalent to taking the mean of the color distribution, while letting $T \to 0$ is the equivalent of taking the mode of the distribution. The mean of the color distribution suffers from similar problems as the euclidean loss does since the most vibrant colors are rarely the mean of the distribution. On the other hand, taking the mode of the distribution can lead to spatially inconsistent results since $\hat{Y}$ can only consist of the 313 quantisized colors. In order to create vibrant but in the same time spatially consistent results, the temperature is set to T = 0.38.

Finally, cubic interpolation is again utilized in order for $\hat{Y}_{h,w}$ to fit the dimensions of the ground truth L channel.

5 Experiment

In this section we will investigate how much of a difference this rebalancing factor introduces by analyzing the plausibility of the colored images. The weights used for color rebalancing are based on the color distribution $\boldsymbol{p}$ of the pixels in our data set (see equation 2) and the result is shown in figure 4. This result further justifies the introduction of weights as the probabilities vary substantially.

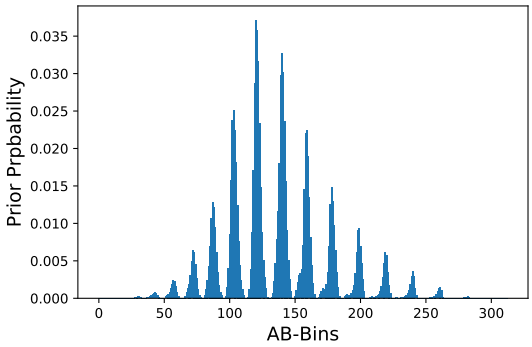

Figure 4: Color probability distribution of the dataset

The network was thereafter trained with and without CR until convergence. The model was tested with the remaining part of the ImageNet validation set that had not been used for validation. Results that we consider successful are shown in figure 5 and results that we consider unsuccessful in figure 6. A common factor between all the images that we consider as plausible predictions is the high amount of grass, sea, sky or forest contained in these pictures. This is also reflected on some images that turned out implausible. The dog in figure 6 is for example given a green body due to the grass in the background. Other images that turned out implausible contained things such as human faces, buildings or detailed environments.

When using plausibility as metric, it is often difficult to determine whether CR turned out successful or not. In general, we believe that CR does improve the result in some cases, while it has no effect or even worsen the result in other cases. The red device in figure 5 is an example of where we judge the more vibrant red color achieved with CR to be more plausible. For the coral reefs however, CR causes the blue color to spill over to the reef and saturate the picture's overall color. In the case of

ocean pictures, it is difficult to determine which loss function generated the more plausible result. On the other hand, in cases where the background is grass, coloring the whole picture in shades of green is not desirable. This effect is seen on the faces of the girls in figure 6. Our guess is that this is a case of a network that has mostly learned to recognize the background features that are overly represented in the dataset. Without CR, other features mostly turn out gray but when CR is implemented, the network can reduce the loss further by coloring these features in the same color as the background.

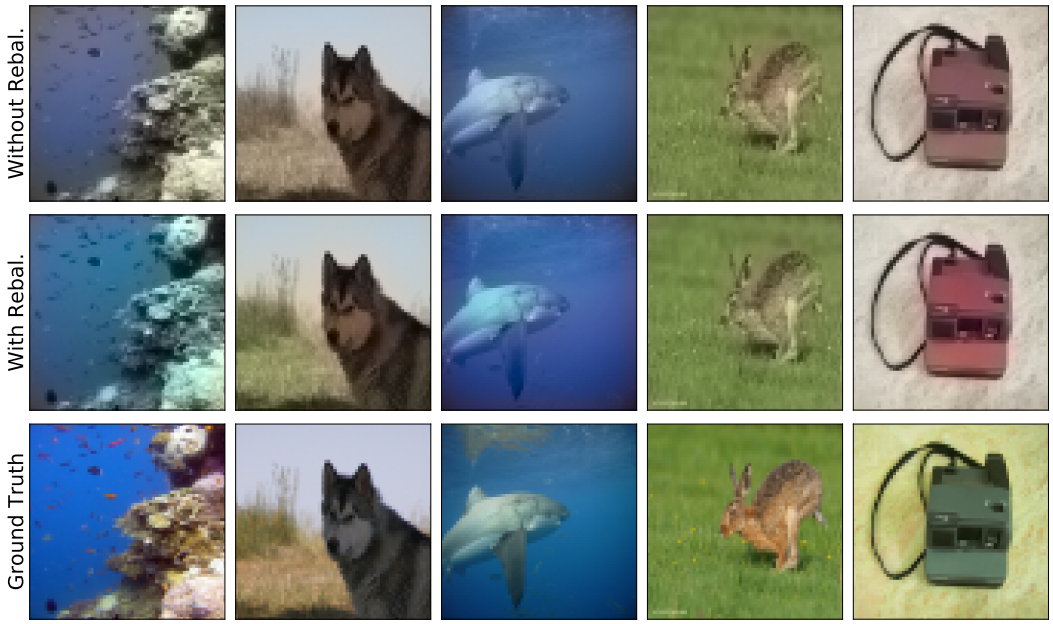

Figure 5: Considered plausible predictions by the network.

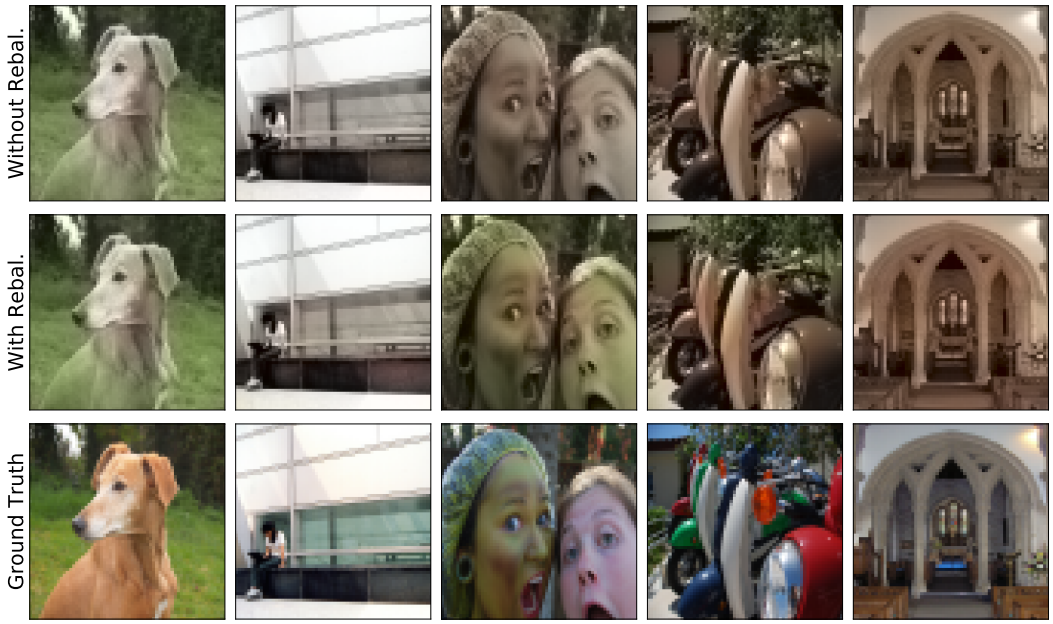

Figure 6: Considered implausible predictions by the network.

6 Discussion and conclusion

As shown in our results, the network could learn to recognize common background features such as grass and sky; yet, the network failed to pick up most other features. There are many potential changes that could improve learning. For example, even though we were not able to increase the number of pixels and dataset size due to storage limitations, these are probably the two factors that would boost learning the most. By increasing the training set, the network would see more examples of similar features and by increasing the image size, we would have the option of expanding the network. In a larger network, fewer features would have to be represented by the same kernel and therefore, the network could learn more. Another improvement to our network could potentially come from experimenting with weight initialization. For instance, Zhang et al. used k-means initialization (3).

A major challenge for image colorization is designing an effective loss function that does not necessarily produce images close to the ground truth but instead, images that seem plausible. CR makes an attempt to design a loss function that defines this goal but it does not always produce the best results. To avoid the issue of defining a complicated loss function, Conditional Generative Adversarial Network (cGAN) have been explored as an alternative method for image colorization. A cGAN would allow us to specify the high-level goal "make the colored output as plausible as the original image", and then automatically generate a loss function that satisfies this goal (2).

To summarize, we cannot conclude that the CR method increases the plausibility of the colorization. In many cases, using CR does increase the plausibility by encouraging saturated colors but sometimes, a very saturated image is not desirable. Up to this date, there is no optimal method for image colorization and more research is needed in order to tailor the loss function and design better network architectures.

References

[1] Cheng, Z., Yang, Q., Sheng, B. (2015) Deep Colorization. International Conference on Computer Vision.

[2] Isola, P., Zhu, J. Y., Zhou, T., Efros, A. A. (2017). Image-To-Image Translation With Conditional Adversarial Networks. The IEEE Conference on Computer Vision and Pattern

[3] Krahenbuhl, P., Doersch, C., Donahue, J., Darrell, T.: Data-dependent initializations of convolutional neural networks. International Conference on Learning

[4] Larsson, G., Maire, M., Shakhnarovich, G. (2016) Learning Representations for Automatic Colorization. European Conference on Computer Vision

[5] Levin, A., Lischinski, D., Weiss, Y. (2004) Colorization using Optimization. ACM Transactions on Graphics 23(3).

[6] Reddi, S., Kale, S., Kumar, S. (2018) On the Convergence of ADAM and Beyond. International Conference on Learning Representations.

[7] Charpiat, G., Hofmann, M., Schölkopf, B. (2008) Automatic Image Colorization Via Multimodal Predictions. Computer Vision – ECCV 2008 pp 126-139.

[8] Smith, L. N. (2017). Cyclical Learning Rates for Training Neural Networks. *arXiv preprint arXiv:1506.01186*

[9] Zhang, R., Isola, P., Efros, A. (2016) Colorful Image Colorization. European Conference on Computer Vision. Representations (2016)

Learning Outcomes

Marcus

I learned about the limitations of high end GPUs and in particular of how one has to adapt and make trade offs between time and quality. There was a lot of work not only to optimize our model in terms of hyper parameters but also how to implement a stable learning algorithm where the model gets fed large data batches without GPU / RAM memory crashes.

It was also interesting to learn about how to outsource the computational workload to more powerful GPUs than the one we had access to in my own PC, which was also made possible by how we had to learn about how to adapt the code towards working on a GPU rather than a CPU.

I did learn more about CNNs but possibly not as much as I'd hoped, implementing them in Keras was straight forward but I noticed my own lack of understanding when it came to troubleshooting.

Julia

My goal for this project was to learn more about CNNs and how to implement one in Keras. I believe that I now have a good understanding on how a standard CNN work and how it can be implemented in Keras. I am also able to make some extensions to the Keras network by defining my own learning rate and loss function. Furthermore, I thought it was interesting to investigate the loss function and its impact in a case where the function definition is not obvious. The project has given me some aspects to look for when designing a loss function as for example looking at prior probabilities or deciding whether it should be a classification or regression problem.

Joakim

Before this project the aim was to learn a bit more about tensorflow and keras. Due to the amount of data needed for this project, the computational power we had to work with was insufficient. Therefore, we started to work in Google Colaboratory (not Google Cloud as was suggested) and was able to 'borrow' a Tesla K80 GPU for the training. I think this will be a useful tool onwards since you typically need a high amount of data for the training, potentially more than your own GPU will be able to handle in a reasonable amount of time.