Grayscale Image Colorization

Ioannis Ioannidis - Dimitris Patiniotis Spyropoulos

University of Piraeus
NCSR Demokritos

Keywords: Deep Learning, Colorization, CNNs, Self Supervised Learning

1. Introduction

Consider the grayscale photographs in Figure 1. At first glance, hallucinating their colors seems daunting, since so much of the information (two out of the three dimensions) has been lost. Looking more closely, however, one notices that in many cases, the semantics of the scene and its surface texture provide ample cues for many regions in each image: the grass is typically green, the sky is typically blue, and the ladybug is most definitely red. Of course, these kinds of semantic priors do not work for everything, e.g., the croquet balls on the grass might not, in reality, be red, yellow, and purple (though it's a pretty good guess). However, for this project, our goal is not necessarily to recover the actual ground truth color, but rather to produce a plausible colorization that could potentially fool a human observer. Therefore, our task becomes much more achievable: to model enough of the statistical dependencies between the semantics and the textures of grayscale images and their color versions in order to produce visually compelling results.

In the present report, we describe how we tried to create such a model using Deep Learning and Convolutional Neural Networks. The structure is as following: Firstly, in the *Data Description* we provide some information about the data we used for the training, validating and testing procedures while in the *Background* section we give an intuitive taste for the CNNs and pretrained MobileNet, after reffering some of the most important domain's related works. *Experiments* part describes the model's architecture and the final performance outcome. Closing, all the remarks highlighted during the task and the observations we ended up with are quoted in the *Conclusion*.

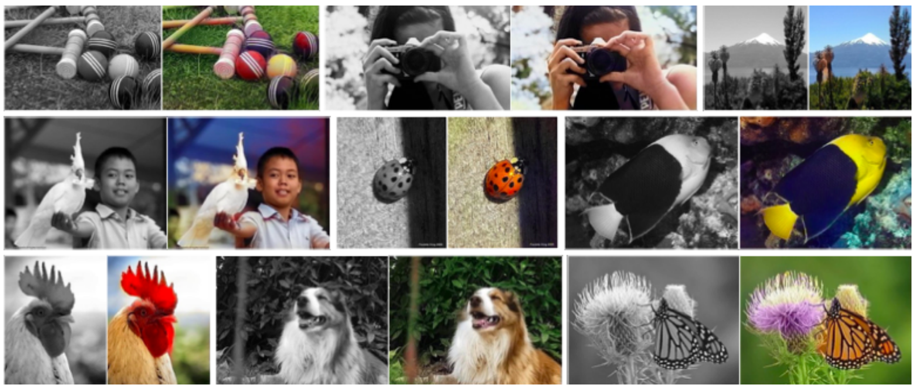


Fig. 1: Grayscaled vs Coloured Photographs

2. Data Description - Preparation

Data for this project is available and extracted by Kaggle [1]. It is about a dataset consisting of 25000 images, including the grayscale and the respective colorized image, at 224×224.

Dataset is based on the LAB color space image format (Fig. 2). Specifically, we have two files:

- *ab.zip* : This contains 3 .npy files consisting of A and B dimensions of LAB color space images.
- *l.zip* : This consists of a gray_scale.npy file which is the grayscale version of the images.

In that way, we will be using the grayscale images (L channel) as inputs and the A, B components as targets. Therefore, training a model in predicting the whole LAB color space, we could get a fully colorized output.

Before passing images through network for training, we need to apply scaling on each channel's values, considering we are using ReLU activations in many layers, as will described in *Model's Architecture* section.

[1]https://www.kaggle.com/datasets/shravankumar9892/image-colorization?select=l

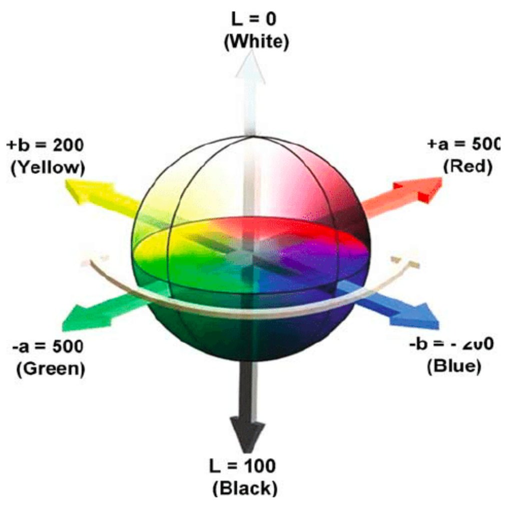


Fig. 2: LAB Color Space

3. Background

3.1. Related Work

First approaches on the photo colorization appeared in the early 00’s. Specifically, in 2002 Welsh T. and his team presented a work, which was able to colorize an input image by transferring the color from a related reference image [1]. At that point, subsequent improvements of this method were proposed, with focus on exploiting low-level features [2] and introducing multimodality on the pixel color values [3]. At the same time, in 2004 another research line was initialized by Levin’s team, who proposed a scribble based method [4] which required some color regions to be specified. This colorization methodology woke the interest of animators and cartoon-aimed techniques were proposed [5,6]. Despite the fact that the results seemed certainly impressive at that time, it turned out that the outcome was highly dependent on the artistic skills of the user. More recently, automatized approaches have been proposed. For instance, Desphande and his coworkers [7], conceived the coloring problem as a linear system problem.

In the last years, CNNs have been proven experimentally to almost halve the error rate for object recognition [8], which led to a massive shift towards deep learning of the computer vision community. In this regard Cheng Z. and his team [9] proposed a deep neural network using image descriptors (luminance, DAISY features [10] and semantic features) as inputs. In 2016, Iizuka, Serra et. al. [11] proposed a method based on using global-level and mid-level features to encode the images and colorize them, while similar approaches have been presented lately as well. For instance, Zhang’s team research [12] proposed a multi-modal scheme, where each pixel was given a probability value for each possible color. Another interesting approach was developed by Larsson [13], in which a fully convolutional version of VGG-16 [14] with the classification layer discarded was used to build a color probability distribution for each pixel. Recently, Zhang’s team [15] presented an end-to-end CNN approach incorporating user “hints” in the spirit of scribble based methods, providing a color recommender system to help novice users and claiming to have enabled real-time use of their colorization system. This recent research proves that this is an ongoing research line.

3.2. Theoretical Background

Artificial neural networks is a very popular approach to solve pattern recognition problems. A neural network is a mathematical model based on connected via each other neural units – artificial neurons – similarly to biological neural networks. Typically, neurons are organized in layers, and the connections are established between neurons from only adjacent layers. The input low-level feature vector is put into first layer and, moving from layer to layer, is transformed to the high-level features vector. The output layer neurons amount is equal to the number of classifying classes. Thus, the output vector is the vector of probabilities showing the possibility that the input vector belongs to a corresponding class.

An artificial neuron implements the weighted adder, which output is described as follows:

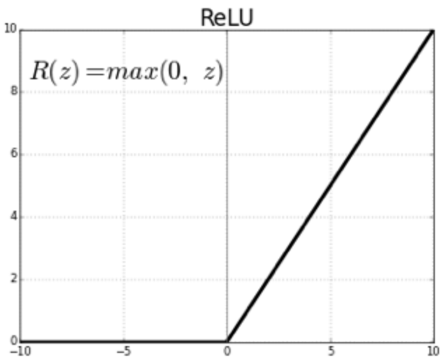
$$a_i^j = \sigma(\sum_k a_k^{i-1} w^j j_k)$$

where a_j^i is the j^{th} neuron in the i^{th} layer, w_k^{ij} stands for weight of a synapse, which connects the j^{th} neuron in the i^{th} layer with the k^{th} neuron in the layer $i-1$. Widely used in regression, the logistic function is applied as an activation function. It is worth noting that the single artificial neuron performs the logistic regression function.

The training process is to minimize the cost function with minimization methods based on the gradient decent also known as backpropagation. In classification problems, the most commonly used cost function is the cross entropy:

$$H(p,q) = -\sum_i Y(i) log y(i)$$

Training networks with large number of layers, also called deep networks, with sigmoid activation is difficult due to vanishing gradient problem. To overcome this problem, the Rectified Linear Unit (ReLU) function is used as an activation function:



Today, using Convolutional Neural Networks (CNNs) is the state of the art pattern recognition method in computer vision. Unlike traditional neural networks, which works with one-dimensional feature vectors, a convolutional neural network takes a two-dimensional image and consequentially processes it with convolutional layers.

A CNN is composed of input and output layers and multiple hidden layers, which can be divided into a convolution layer, a pooling layer, a rectified linear unit layer, and a fully connected layer. The architecture of a CNN is shown in Fig. 3.

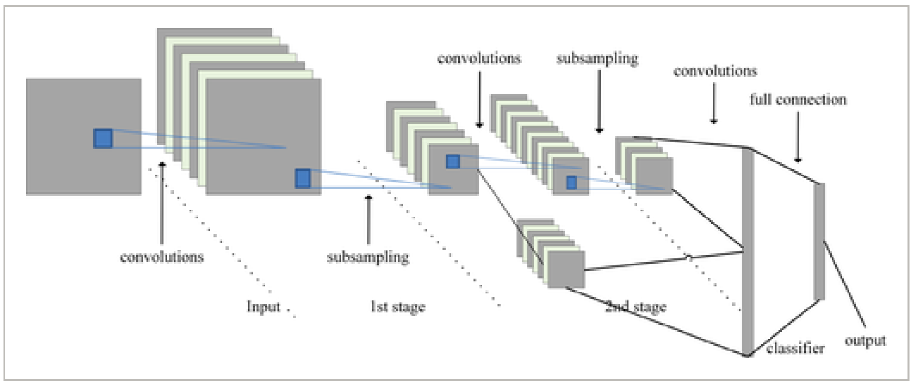


Fig. 3: Convolutional Neural Network Architecture

Each convolutional layer consists of a set of trainable filters and computes dot productions between these filters and layer input to obtain an activation map. These filters are also known as kernels and allow detecting the same features in different locations.

Compared with a traditional neural network, a CNN has three major characteristics, namely local perception, weight sharing, and a multi-convolution kernel. Based on these characteristics, a CNN has several advantages, including the following:

1. the architecture can well capture the spatial topology of input
2. weight sharing reduces the number of trainable network parameters, thus reducing the network complexity and improving robustness
3. the shared convolution kernel allows no pressure to perform high-dimensional data processing

4. the hardware implementation of a CNN is much simpler than a fully connected neural network of the same input size

In order to increase the performance of our final model we also used the pretrained MobileNet. MobileNet is an efficient and portable CNN architecture that is used in real world applications. MobileNets primarily use depthwise seperable convolutions in place of the standard convolutions used in earlier architectures to build lighter models. A depthwise seperable convolution is made from two operations:

1. depthwise convolution
2. pointwise convolution

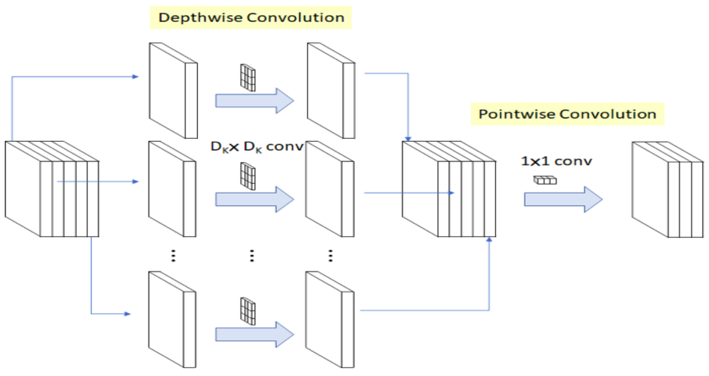


Fig. 4: Depthwise seperable convolution

Counting depthwise and pointwise convolutions as seperate layers, a MobileNet has 28 layers.A standard MobileNet has 4.2 million parameters which can be further reduced by tuning the width multiplier hyperparameter appropriately. The size of the input image is 224 × 224 × 3.

4. Experiments

4.1. Model's Architecture

Our fully model is about a CNN Encoder - Decoder form combined with MobleNet. In this section we describe each branch.

The encoder network receives the input grayscale image and uses a sequence of convolution and Leaky and simple ReLU layers to decompose the input image into low-level (edges, colors, textures, etc.) and high level features. This procedure is applied for a depth k layers (here we use k=5). Also features are extracted by the parallel pass of the grayscale input through MobileNet.

Those parallel features are combined by the following fusion layer using ReLU activation function.

Fusion's output is passed through the final decoder, which runs it through a sequence of convolution and ReLU layers (except the last one which includes Tanh activation) to recover the LAB's A and B channels.

During training both encoder and decoder are trained end-to-end using the Mean Squared Error loss function.

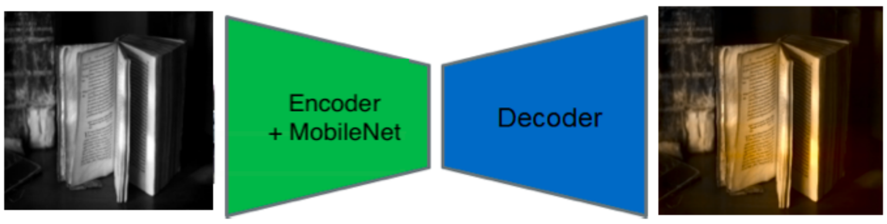


Fig. 5: Model's architecture

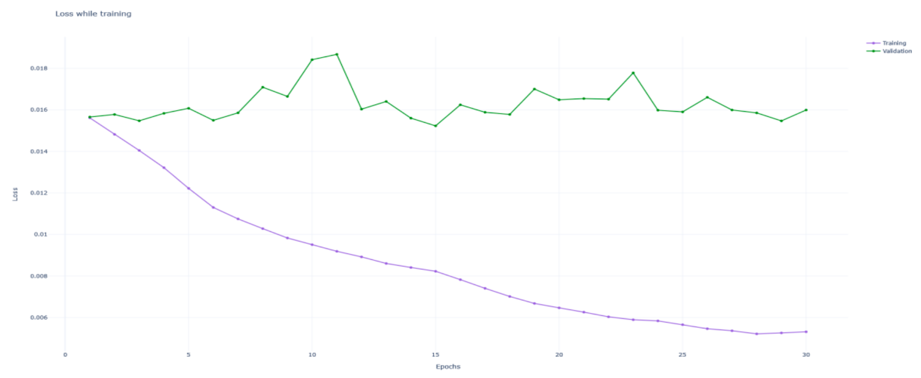
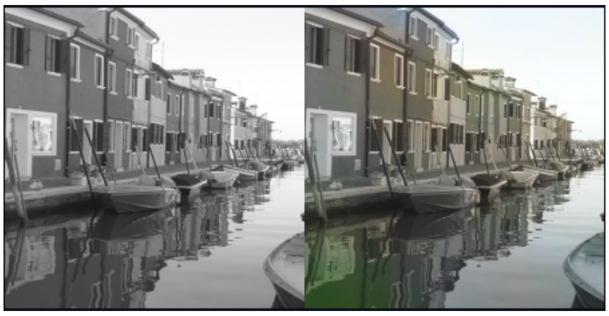
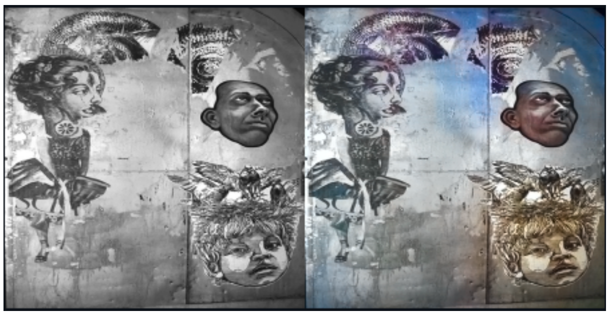
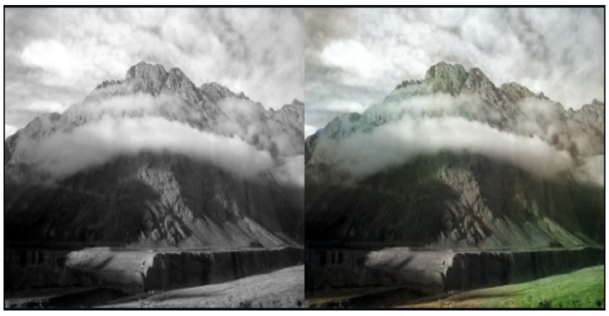
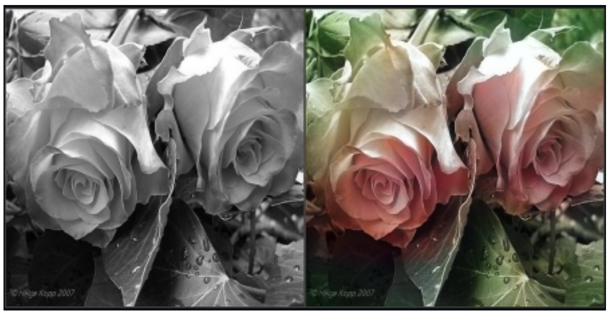


Fig. 6: Loss vs Epochs Plot

4.2. *Results*

We trained the network for 30 epochs, attempting to minimize validation's loss. Despite the gradual and normal fall of the error in training set as was expected, the validation's one remained on the same levels (Fig.6).

However the model proved to efficiently predict a part of the colorized photo. Some of the final results are presented below:

5. Conclusion

We ended up that image colorization consists a challenging task. It is computational expensive and requires a lot of time and memory. Our final model returns plausible representation of the colorized image using a bespoke architecture, which was our initial goal. However, there are many ways to improve the final outcome for getting more realistic images. Increasing sample size, better hardware, more training epochs and tweaking the model's architecture are some of those.

References

1. Welsh T., Ashikhmin M., Mueller K. (2002). *Transferring color to greyscale images.* ACM Transactions on Graphics (TOG), Volume 21.
2. Ironi R., Cohen-Or D., Lischinski D. (2005). *Colorization by example.* Rendering Techniques, Citeseer.

3. Charpiat G., Hofmann M., Scholkopf B. (2008). *Automatic image colorization via multimodal predictions.* Computer Vision–ECCV 2008.
4. Levin A., Lischinski D., Weiss Y. (2004). *Colorization using optimization.* ACM Transactions on Graphics (TOG), Volume 23.
5. Qu Y., Wong T.T., Heng P.A. (2006). *Manga colorization.* ACM Transactions on Graphics (TOG), Volume 25.
6. Sykora D., Dingliana J., Collins S. (2009). *Lazybrush: Flexible painting tool for hand- drawn cartoons.* ACM Transactions on Graphics (TOG), Volume 28.
7. Deshpande A., Rock J., Forsyth D. (2015). *Learning large-scale automatic image colorization.* Proceedings of the IEEE International Conference on Computer Vision.
8. Krizhevsky A., Sutskever I., Hinton G.E. (2012). *Imagenet classification with deep convolutional neural networks.* Advances in neural information processing systems.
9. Cheng Z., Yang Q., Sheng B. (2015). *Deep colorization.* Proceedings of the IEEE International Conference on Computer Vision.
10. Tola E., Lepetit V., Fua P. (2008). *A fast local descriptor for dense matching.* Computer Vision and Pattern Recognition
11. Iizuka S., Simo-Serra E., Ishikawa H. (2016). *Let there be color!: joint end-to-end learn- ing of global and local image priors for automatic image colorization with simul- taneous classification.* ACM Transactions on Graphics (TOG).
12. Zhang R., Isola P., Efros A.A. (2016). *Colorful image colorization.* European Conference on Computer Vision.
13. Larsson G., Maire M., Shakhnarovich G. (2016). *Learning representations for automatic colorization.* European Conference on Computer Vision.
14. Simonyan K., Zisserman A. (2015). *Very deep convolutional networks for large-scale image recognition.* ICLR.
15. Zhang, R., Zhu, J.Y., Isola, P., Geng, X., Lin, A.S., Yu, T., Efros, A.A. (2017). *Real- time user-guided image colorization with learned deep priors.*