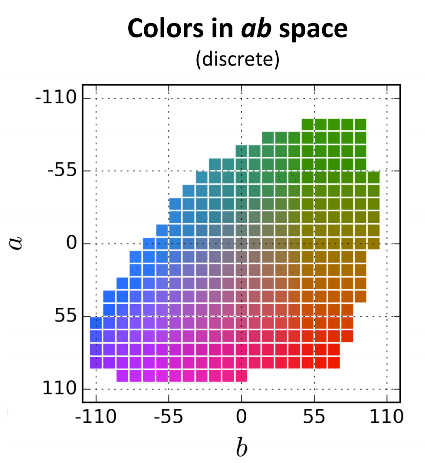
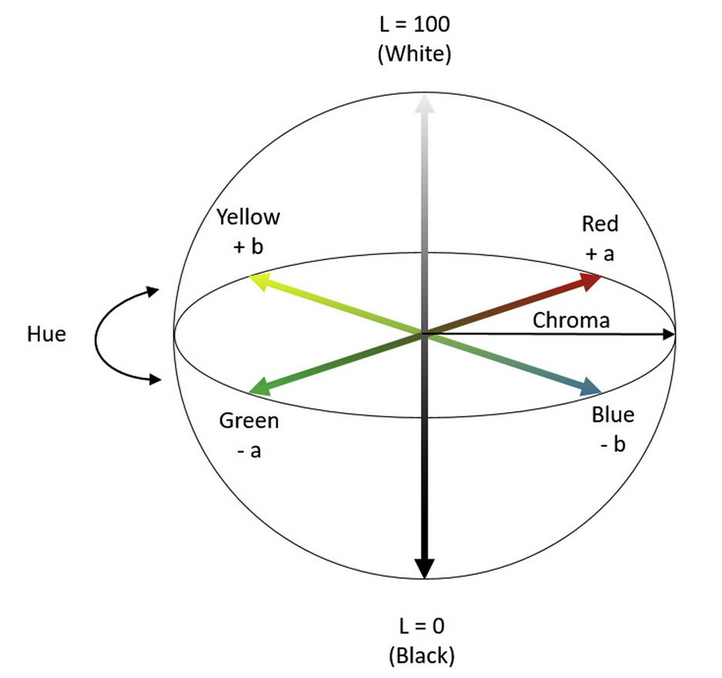
CSI 4106 Project: Image colorisation

Simon Paquette

300044038

The project of coloring black and white images to produce an output with possible real colors from grayscale is a complex subject. With tools like OpenCV and TensorFlow, this project helped me gained experience in the idea of LAB color space, using OpenCV to apply transformation on images and applying an AI model of neural networks to train from a dataset.

I used 2 different datasets, principally of landscapes images. The split train/val/test that I use is 70/15/15. First a dataset consisting of 7129 landscape images (street, buildings, mountains, glaciers, trees, etc.) of size 150x150 from <https://www.kaggle.com/theblackmamba31/landscape-image-colorization>. Secondly, a dataset consisting of 8800 images of travel and adventures of 256x256 pixels from <https://www.kaggle.com/datasets/duttadebadri/image-classification>. The first dataset is easy to use but have a lot of blue/green background colors and represent well my type of images I wanted, a landscape. The second dataset will be used to perform training on bigger images with a dataset with more variety all related to travel: landscape, peoples, city, cultural, object, animals.

NOTE : OpenCV rescale all those AB values in the range [0,255]

How to solve the problem?

During the last few years, many different solutions have been proposed to colorize images by using deep learning.

Colorful Image Colorization paper (<https://arxiv.org/abs/1603.08511>) ( <https://richzhang.github.io/colorization/>) approached the problem as a classification task and they also considered the uncertainty of this problem (e.x. a car in the image can take on many different and valid colors and we cannot be sure about any color for it)

However, another paper approached the problem as a regression task (with different technic like GAN) (<https://towardsdatascience.com/colorizing-black-white-images-with-u-net-and-conditional-gan-a-tutorial-81b2df111cd8>).

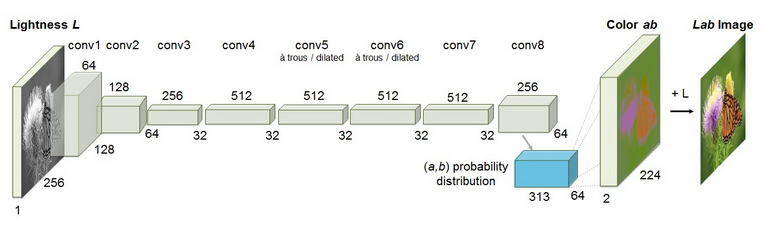
There are pros and cons to each approach. The problem with regression task is a more conservative result, so more grayish images. Personally, I dive into the regression approach. Approaching the problem as a classification of ab color in 313 different bins is a bit more complicated. We need to apply a specialized loss function and download some prebuilt model of detection to classify each pixel to a specific bin using caffemodel or prototxt.

The loss functions used is either MAE (L1) or MSE(L2) or some other homemade specialized loss function.

The Optimizer used or either ADAM or RMSPROP.

The model is generally split in 2 steps, an encoder, and a decoder. The encoder is, for the most application, conv2d layers with LeakyRelu or Relu activation function, sometimes with dilation, followed by batch normalization for some algorithms. The input is filtered either by stride=2 or Maxpooling. The decoder is always upsampling2d with classic conv2d layers.

Here an example of layers to use, the probability distribution step represents the classification task, where I did not try to implement because I favorized the regression task.



Images Preprocessing:

1. With OpenCV, read the image as BGR color.
2. Resize to a default size 128x128 (or 256x256)
3. Make sure the image opened is uint8
4. Convert BGR to LAB
5. Set each value to float32
6. Normalize the pixel by dividing by 255
7. Split each channel to get L (lightness), A (green-red), and B (blue-yellow)
8. Return L channel for features and merge AB channel for labels

Reconstruct the images:

1. Merge the original L channel (grayscale) with the new predicted AB channel
2. Clip each pixel value to make sure they are in the range [0,1]
3. Multiply each pixel by 255
4. Set each value to uint8
5. Convert LAB to BGR to save the image or LAB to RGB to show the image

Problem list:

* Output images: blue and green colors are favorized, probably because landscape image have more of those colors (unbalanced data)
* Out of GPU memory. So, need to set batch=8 and sometimes reduce the number of layers layers

I applied different callbacks during training:

* Early stop
* Model checkpoint
* Tensor board
* Reduce learning rate plateau

Brief description of different parameters:

* **EPOCH:** Generally, between 20 and 25 epochs
* **LOSS:** MSE vs MAE -> hard to say which is best. MAE sometime create false color but produce more vibrant colors. MSE is stable but more grayish image (unsaturated). I have tested more with MSE at the beginning, but now more with MAE.
* **OPTIMIZERS:** ADAM vs RMSPROP -> ADAM, similar, but ADAM detect more specific details. May need to test both sometimes. ADAM loss may have minor worst result but is more stable for each algorithm. Tried SGD optimizer, really bad result