Color Difference Recognition with Neural Networks in an Industrial Color Copy Process

Fernando De Nitto
fernandodenitto@gmail.com
483505

Intelligent Systems
Department of Information Engineering
University of Pisa

2019-05-03

1 Project Description (Part 1)

The project involves an industrial process of copying colors. In particular, the goal of the project is as follows (as shown in the text): in order to objectively compare a copy to a master, it is required to design and develop a neural network, which must be designed and trained to measure the difference between two similar colors. From an operational point of view, the network takes as input the representations of a master color and of a copy, and returns their color difference. To calculate the difference between two colors, you use a machine-independent color space called CIE Lab as shown in figure 1. Using a simple vector calculation, it is possible to obtain the color difference as reported in the formula 1.

$$\Delta E_{ab} = \sqrt{(L_2 - L_1)^2 + (a_2 - a_1)^2 + (b_2 - b_1)^2} \tag{1}$$

In particular, from the result above it is possible to infer as follow:

- when $0 < \Delta E_{ab} < 1$ an observer does not notice the difference;
- when $1 \leq \Delta E_{ab} < 2$ only experienced observers can notice the difference;
- when $2 \leq \Delta E_{ab} < 3.5$ unexperienced observers also notice the difference;
- when $3.5 \leq \Delta E_{ab} < 5$ clear difference in color is noticed;
- when $\Delta E_{ab} > 5$ an observer notices two different colors.

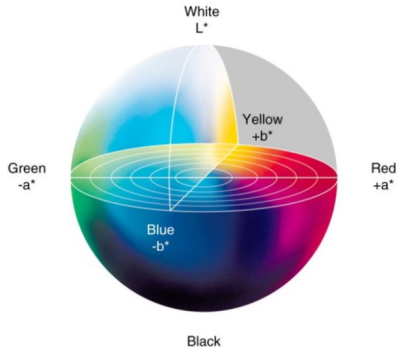

Figure 1: CIE Lab Color Space

2 Resolution of Part 1

The following section explains the choices you make in implementing the first part of the project.

2.1 Dataset Reduction

The dataset is made up of 1269 samples of colors named "master" provided as a spectrum vector at a wavelength of 380nm to 800nm, the wavelenght of visible light. From these masters, distorted copies will need to be generated and then calculate the difference between the original colors and the copies as required by the project. It is possible to see (figure 2), through a visual analysis of the dataset, that many master colors are "similar to each other" so the first operation was to skim the dataset by reducing the number of master colors. This helps reduce the computational cost of operations at first. In the next step, you will use the entire dataset. The solution used to reduce the dataset is to use the formula 1 and eliminate the colors that have a difference between them, in terms of Lab coordinates, less than 3. After the reduction the number of samples in the dataset are 259 and the final dataset is shown in figure 3.

2.2 Copy Creation

For the aim to create an homogeneus dataset The hypothesis used for the creation of the copies is the follow one: since the reference context is an industrial process, it is difficult to have a color difference greater than 5 i.e. a $\Delta E > 5$. A script (noiseInterval.m) then evaluated the noise interval that could keep the low percentage of copies with $\Delta E > 5$ but not negligible (for the purpose of the creation of an homogeneous dataset). Starting from an absent noise i.e. with a factor of 1, noise intervals were analyzed by increasing noise factor by 0.01. Table 2.2 shows the results. The noise factor used is 1.18 and the range therefore varies randomly in [1,1.18]. To obtain a distortion of the spectrum, and then a

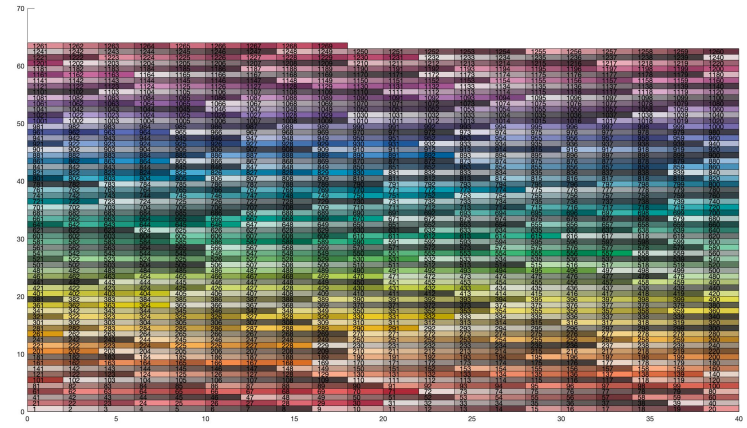

Figure 2: Original Dataset

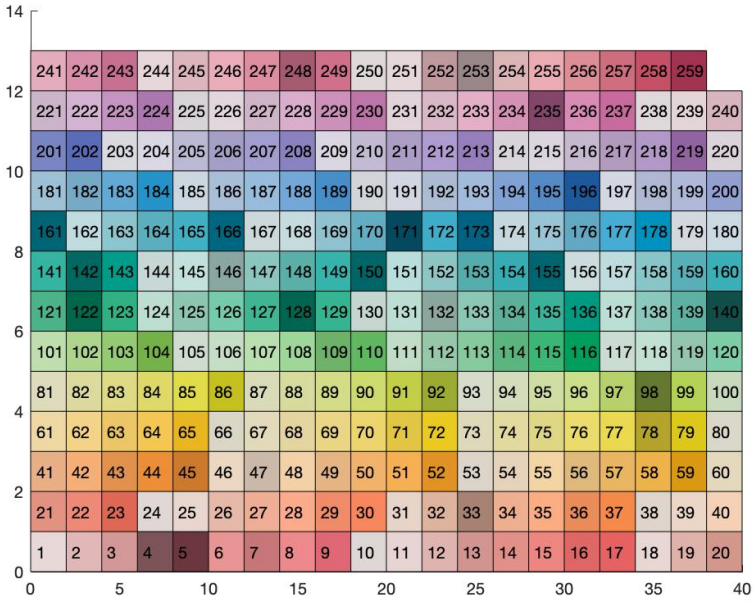

Figure 3: Reduced Dataset

copy, it is chosen to multiply the spectrum by a noise factor. An example can be shown in Figure 4 . The generation of copies was achieved by running the script **makeMasterCopyMatrix.m** that generated a number of 10 distorted copies for each master color in a noise range ranging from [1.00,1,18].

Interval	$0 < \Delta E < 1$	$1 \leq \Delta E < 2$	$2 \leq \Delta E < 3.5$	$3.5 \leq \Delta E < 5$	$\Delta E \geq 5$
$[1; 1.12]$	0.25	0.29	0.38	0.07	0
$[1; 1.13]$	0.23	0.27	0.37	0.13	0
$[1; 1.14]$	0.25	0.24	0.33	0.18	0.001
$[1; 1.15]$	0.21	0.23	0.34	0.21	0.01
$[1; 1.16]$	0.21	0.21	0.31	0.24	0.03
$[1; 1.17]$	0.20	0.20	0.30	0.24	0.06
$[1; 1.18]$	0.19	0.18	0.28	0.25	0.10
$[1; 1.19]$	0.17	0.18	0.26	0.26	0.13

Table 1: Percentage of Value in an interval of noise factors

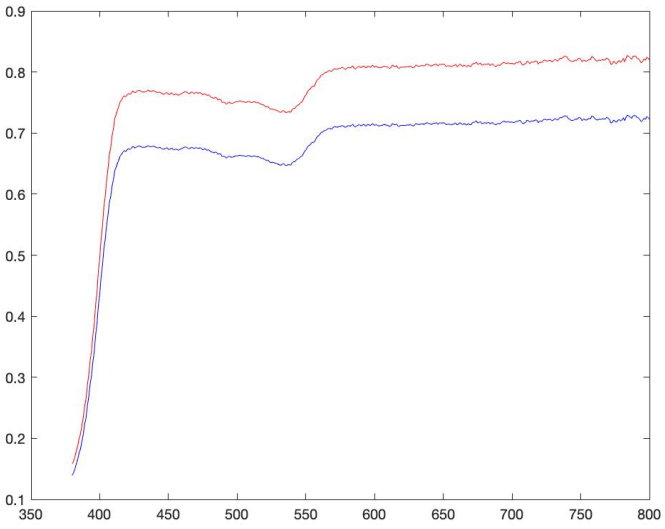

Figure 4: Master Color and Distorted Copy

2.3 Feature Extraction

Since the inputs of the neural network are represented by the 421 components of the master spectrum and 421 of the copy spectrum, a reduction in the number of inputs is necessary since 842 inputs are too much. The basic idea adopted in

this project is to divide the spectra into intervals and calculate for each range the features to be proposed as input to the network. The first features chosen for each range were the standard statistical indexes: **Mean, Median, Maximum, Minimum, Variance and Standard Deviation** as shown in Figure 6.Since the mathematical **standard deviation** is the square root of the variance it was decided not to insert it in the characteristics since for the network it could be a redundant information (there is a very strong correlation about the two features). After many experiments with statistical indexes the number of inputs has been reduced considerably with good performance. For example with a number of 10 intervals the inputs go from 842 to 120 with an 86% gain in terms of number of inputs. After this process, through the feature selection phase it is possible to optimize this gain even more. However, this approach has been discarded since it is possible to do even better with the following one. If you take into consideration the only average of each interval, you can reconstruct the original spectrum and also remove the noise (as shown in Figure 5). **In practice we tried to understand if the only averages are sufficient to characterize the entire original spectrum.** The first question to answer is what is the number of intervals. The choice of the number of intervals was made according to the following criterion: the spectrum is divided into intervals and the average is calculated for each interval. The new spectrum will have a number of values equal to the number of intervals. The new spectrum is interpolated (with a cubic interpolation) using the **interp1()** function of the Curve Fitting Toolbox and calculates how much interpolation differs from the original spectrum by the imse() function that calculates the mean squared error. If the value is very close to 0 then the interpolation represents a good approximation of the original function. In addition, the interval division also reduces the noise of the original spectra as shown in figure 5 . After many experiments, the number of intervals chosen is 15 as the value of the average quadratic error settles into an order of magnitude ranging from 10^{-4} to 10^{-5} that is a very good result.

2.4 Feature Selection

After extracting the characteristics for each interval, it is possible to further decrease the number of inputs by selecting the characteristics. This selection was made through the selectionfs function. After numerous experiments, a correct tradeoff was chosen for the number of features extracted equal to 6. The stop criterion of the function was the performance of the neural network used for the selection of the characteristics and that is the mse. After many iterations and experiments the final columns included at the end are almost the same: [1 7 16 22]. **In this way the number of inputs are reducted from 30 to 4**.

2.5 Setting the Neural Network

At this point in the project it is possible to use the inputs and targets as input of a shallow neural network for function fitting. The choice of the implementation of the neural network is performed by creating a MATLAB network using the

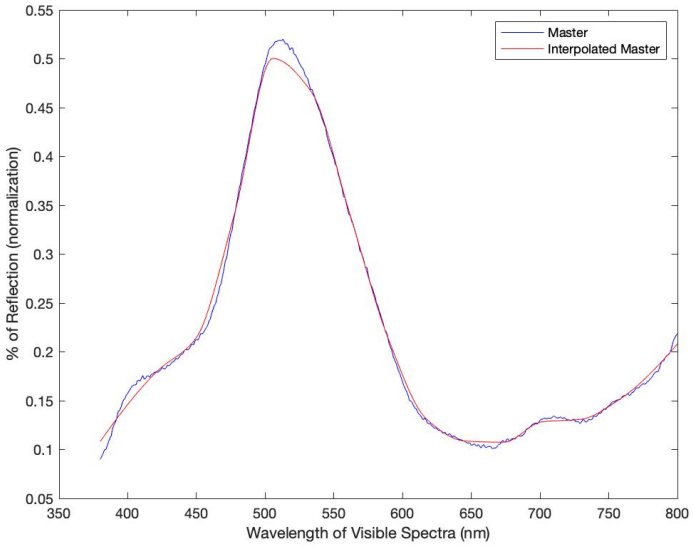

Figure 5: Master Color and Interpolation

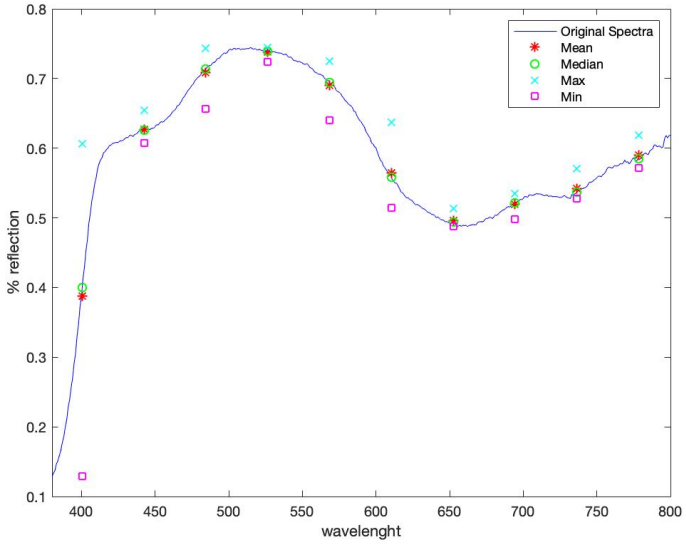

Figure 6: Features extracted from a spectra

fitnet() function with a number of neurons equal to n and calculating the

performance:

```
hiddenLayerSize=n
net=fitnet(hiddenLayerSize);
net.divideParam.trainRatio=70/100;
net.divideParam.valRatio=15/100;
net.divideParam.testRatio=15/100;
net=train(net,inputs,targets);
```

The performances are calculated starting from a number of hidden neurons equal to 1 and increasing it. This is done 10 times and network performance in terms of *mean squared error* and *training time* are calculated.The results of the experiment are shown in the following Table 2.5.From the results of the experiments it is possible to see that a good trade-off for the choice of the number of hidden neurons is represented by n = 4. The randomness of the

No. Hidden Neurons	**mse**	**Training Time**
1	0.15	0.30
2	0.03	0.98
3	0.027	0.66
4	0.010	0.72
5	0.010	1.3

Table 2: Performances evaluated in Neural Network's Training

experiment repetitions is introduced by the 'dividerand' option of the MATLAB neural network toolbox. In this way the training, testing and validation sets are represented by different elements for each iteration. In general the choice of a greater number of hidden neurons involves more time for the traning and more samples are required. Having a greater number of neurons also results in a better fit and a better regression coefficient. Adopting the results of the experiments and the choices made above the final network has a **regression coefficient equal to 0.9982 and an mse equal to 0.0103**. The value of the regression coefficient very close to 1 confirms that the fitting of the function is very good as show in figure 7.

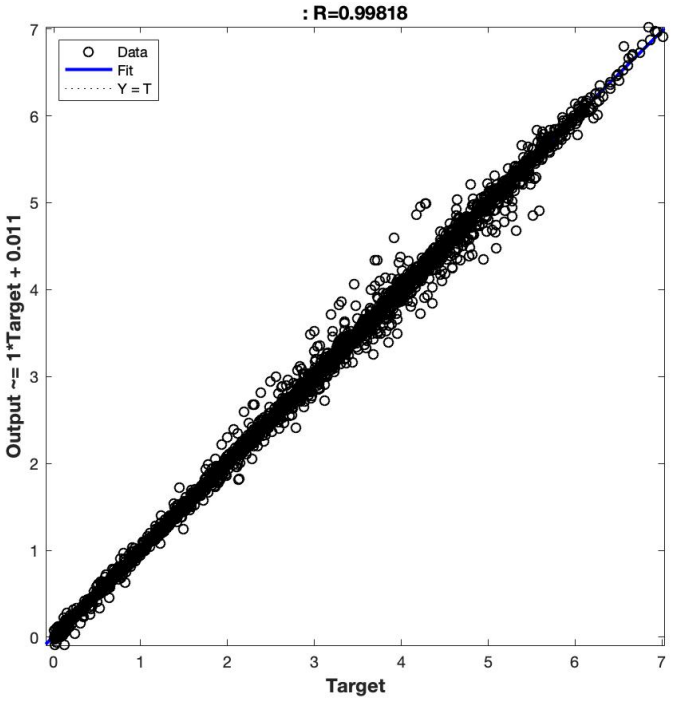

Figure 7: Regression Plot of the Final Neural Network