Image Colorization with Neural Networks

Matías Richart, Jorge Visca, Javier Baliosian
University of the Republic, Montevideo, Uruguay.
Email: {mrichart, jvisca, javierba}@fing.edu.uy

***Abstract*—We propose a method for colorizing photos, this is, providing a color version of a given gray scale image. The method does not depend on human input, and is completely automatic. It does not depend on segmentation, scribbling or sophisticated image processing techniques. It is based on training a simple classifier using back propagation over a training set of color and corresponding gray scale pictures. The classifier predicts the color of a pixel based on the gray level of the pixels surrounding it. This small patch captures a local texture. To keep the domain for the predictor small, the colors are reduced using Self Organizing Maps. This reduction produces a small set of chroma values with enough variation as to generate good approximations for all colors in the training set.**

I. Introduction

In this work we propose a methodology for the colorization of gray level images using a learning approach based on *Neural Networks* (NN). Image colorization has been well studied in the field of image processing and computer vision. However, in this work we propose to develop a methodology with minimal usage of image processing algorithms and mainly based on learning techniques.

The proposal consists of a *Multi Layer Perceptron Neural Network* used for classification to predict the color of gray level pixels. As the number of possible colors to learn is considerable, we propose a vector quantization method based on Self Organizing Maps to reduce the search space and arrange colors in a small number of clusters. Then, our NN classifies gray level pixels into color clusters.

Image colorization has been a field of study for several years. The term colorization was introduced by Wilson Markle in 1970s to define the process of assisting a person on giving color to a black and white movie. The work of Markle [1], as well as many that follow are conceived to assist a human on the coloring process. For example the works of [2], [3], [4] and [5] are considered semi-automatic coloring process as they need some reference input (and knowledge) from a human to colorize an image. All these works, obtain very good coloring results with reasonable computing performance, however they all need a coloring reference provided by a human.

More recently, proposals for fully automatic image colorization have appeared: [6], [7], [8]. These works propose to use recent techniques such as *Convolutional Neural Networks* and *Deep Learning* jointly with a deep understanding on image processing.

Our proposal differentiates from previous works in that we follow a simple and straightforward approach for the learning technique and for the image processing. We propose a novel technique for color clustering with the help of a well known approach as the *Self Organizing Maps*. We also introduce a colorization process which leverages the simplicity and efficiency of traditional *Neural Networks*. Although we could not get the same color results as the more elaborated works, we consider our proposal very effective for its simplicity.

The rest of the article is organized as follows. In Section II we present our approach for image colorization and describe the different steps of the work-flow for training and prediction. In Section III we describe the application of our method to a particular set of images, explaining the particular configurations of the learning algorithms. Finally, we give some concluding remarks in Section IV.

II. Methodology

The methodology we propose is inspired in [9] and [10]. As previously mentioned, the objective is to give color to a gray level (black and white) image. For achieving this, our main proposal consists of using a *Neural Network* (NN) which learns the relationship between the level of luminosity in the gray level image and the color.

Our approach comprises a sequence of steps for preparing the image, reducing the dimensionality of the problem, training the NN and post-processing of the image. The general training work-flow can be seen in Figure 1. In the following subsections we will explain in detail each step.

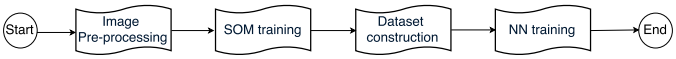


Fig. 1. General Training Work-flow.

A. Image pre-processing

The first step is the image preprocessing, which mainly consists on taking a color RGB image and converting it to the CIELUV [11] color space. In an RGB image, each pixel has 3 channels which correspond to the colors red, green and blue and can take values between 0 and 255. In the CIELUV color-space, there are also 3 channels: L, U and V. The L channel is the luminance or lightness, while the U and V components are the chromatic values. Working with CIELUV has two major advantages in our case: (i) the color space is designed for perceptual uniformity which makes it ideal for computer processing and (ii) our model only has to predict the U and V channels, as the L channel would be the gray scale image.

Then, the second step of the image pre-processing is to separate the image in two new images, one with only the L

component and one with the U and V components. For now we can think that the L component image is the input to our model and the U and V component image is the target.

B. Color reduction with Self Organizing Maps

With the aim of reducing the space of possible values to learn, we propose to reduce the color range. For this objective, we use the technique of vector quantization. Vector quantization consists on mapping k-dimensional vectors into a reduced set of vectors $Y = y_i : i = 1, 2, ..., N$. Each vector y_i is called a *code vector* or a *codeword* and the set of all the codewords is called a *codebook*.

There are several techniques for vector quantization, being the LBG algorithm [12] the most famous one. However, in this work we propose to use *Self Organizing Maps (SOM)* [13] as it is a fast and self-learning alternative [10].

Self Organizing Map is a type of neural network introduced by Teuvo Kohonen in 1982 [13]. It consists of an input layer, which size corresponds to the number of elements of the vectors to reduce, and an output layer (or computational layer, or map) where each neuron represents the code vectors (see Figure 2). SOM learning is unsupervised, as there is no target or objective to compare with. It can be seen as a mechanism to split and classify the input data into classes based on common features. In summary, as mentioned previously, the aim is to transform an input of arbitrary dimension into a discrete map of a much lower dimension.

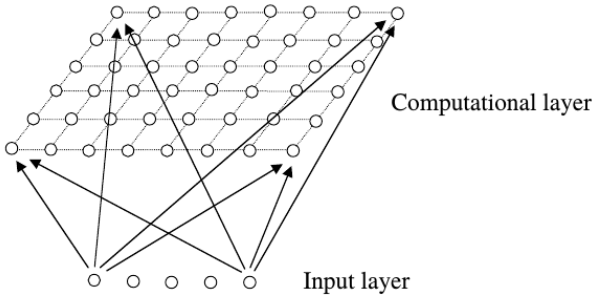


Fig. 2. Self Organizing Map.

In our case we want to reduce the target space, the U and V components of each pixel image. We can see each pixel as a 2-dimension vector, where each component is a real number in the range $[-100, 100]$. Then, with vector quantization the aim is to reduce this space to a finite set of vectors.

As can be seen in Figure 3, the output of this process consist of a trained SOM and a codebook. The trained SOM can be used to clusterize (or classify) U,V vectors and its output is an index which represents the cluster or class that vector belongs to. The color codebook is used to obtain the U,V vector which represents each cluster.

C. Dataset construction

Starting from the images preprocessed and the codebook, the next step is to construct the data set for training. The

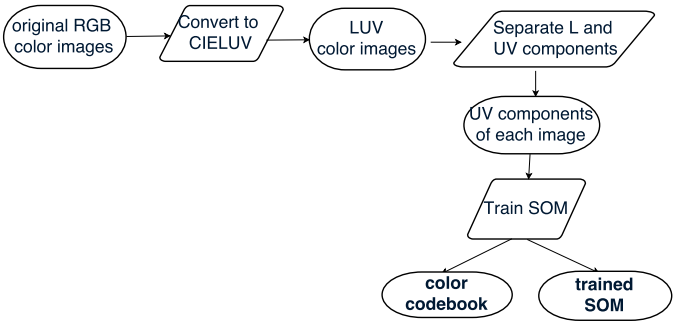


Fig. 3. Image Pre-processing and Color Reduction Work-flow.

simplest dataset would consists of one feature, the L value of a pixel and one target for each feature, the U and V values for that pixel. In this work we propose to add more features to the dataset by also considering the pixels near to the pixel we want to predict. The rationale of this is that, in general, in the images there is some homogeneity in the colors, and that information of the nearby pixels can help to predict the color of the target pixel. Also, the adjacent pixels contains a texture that also can help in classifying the pixel.

So, for each pixel we consider a neighborhood of pixels to create the dataset which we call a *patch*. This patches are squares of odd size where the pixel to predict is in the middle. Then, our dataset has a number of features which depends on the number of pixels of the patch.

After obtaining the patch, we need to obtain the target for that row of features. As previously mentioned, in our case the target is the U,V component of the central pixel of the patch but with a reduction in the possible values. For this reduction we utilize the SOM technique described earlier. The U,V component of the central patch is given as input to the trained SOM and an index is obtained which represents the cluster of the U,V component. This index is used as target in our dataset.

This process is depicted in Figure 4.

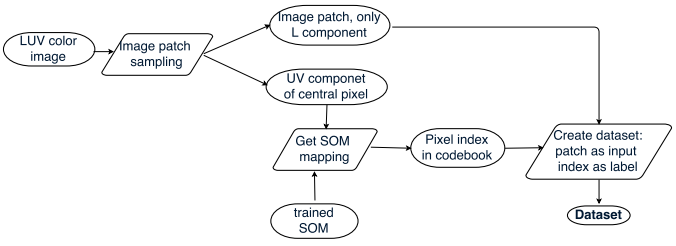


Fig. 4. Dataset Construction Work-flow.

D. Neural Network

We propose to use a Multi Layer Perceptron (MLP) Neural Network for classification. The objective of the NN is to classify each patch into the SOM cluster the central pixel belongs to. So, the number of input neurons of our NN is the number of pixels in the patch (N) and the number of output neurons depends on the size of the SOM (S), as we

have one output for each possible classification (see Figure 5). As we are doing classification, the output of the NN prediction corresponds to the winning output neuron.

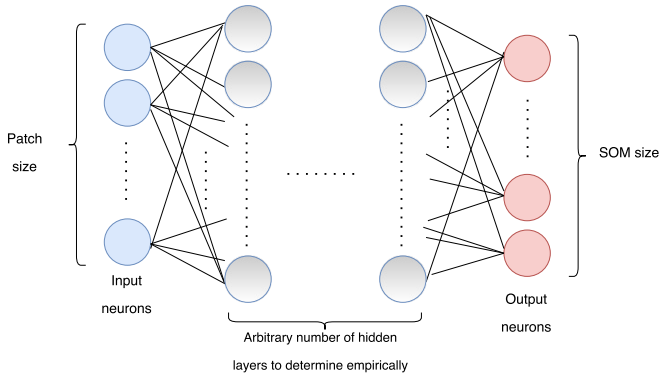


Fig. 5. Neural Network Model.

The training of the NN is done with the Back-propagation algorithm in a batch process using the dataset previously constructed (Figure 6).

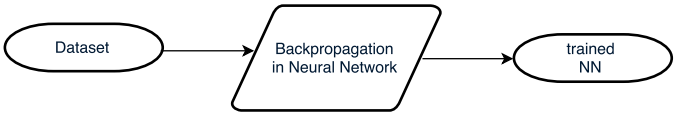


Fig. 6. Neural Network Training Work-flow.

E. Color image recovery

Finally, after having the NN trained, we are able to colorize a gray level image. The process of colorization (Figure 7) consists of three main steps for each pixel of a gray level image:

- Use the NN to predict the index in the codebook for the pixel.
- Use the index and the codebook to obtain the U and V components of the pixel.
- Join the L component from the gray image and the predicted U and V components to obtain the final 3-dimension pixel.

It is important to notice that because of the color reduction performed, the final color image obtained will have a reduced number of colors. The color space of the final image will depend on the size of the SOM used.

III. Use Case

In this section we present the implementation and usage of our proposal. We use the Python language with the scikit-learn library [14] for the NN classifier implementation and for the SOM network we use an external library called sompy [15]. We have made all the code available in [16].

Realistically, the system is not to be used on arbitrary images but to images from a class. This class of images would be used to train the predictor, and can be portraits, outdoor

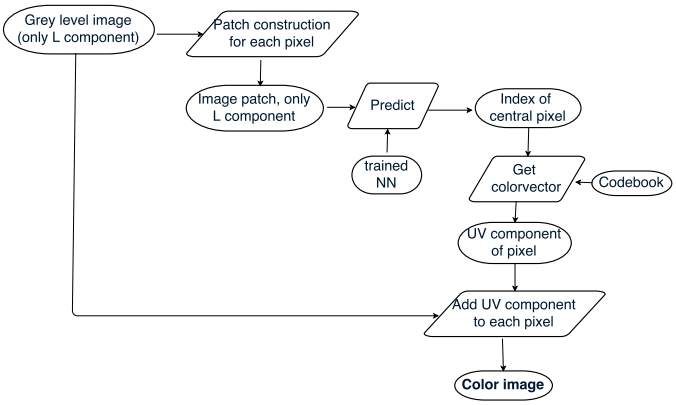


Fig. 7. Color Prediction Work-flow.

nature shots, architecture, etc. For this work, we selected the "Open Country" image collection from the *Label Me* project [17]

A. Dataset

The image collection consists of 410 jpeg-encoded 256×256 color images. The images are exterior nature shots of various landscapes: desert, mountain ranges, open fields, hills, etc. The pictures vary greatly in quality, time of the day, atmospheric conditions, height of viewpoint, and so on.

To get a manageable amount of data for training, we randomly sampled pixels from all images. We processed 2500 pixels from each image, with associated patches, to construct a 1025000 elements dataset. We built 3 datasets using different patch sizes: 3×3, 5×5 and 11×11.

B. Color reduction

One of the key aspects of this step is the selection of the number of output neurons of the SOM (size of the map). This gives the number of different classes the colors will be classified in. As lower the size, more different colors will be classified in the same class, reducing the color range, but it would reduce the target (or label) space for the following neural network training. Then, we have a compromise between an easier training in the neural network but a reduced number of colors in the final output.

In our experiments we test different sizes for the SOM map: 9, 25, 100 and 400 neurons. For each size, we train the SOM with the same dataset and we measure the quantization error. The results can be seen in Table I.

TABLE I
SOMs Training Results

Number of neurons	Error
9	0.621739
25	0.4585945
100	0.19889
400	0.100954

As expected, as bigger the map size and the possible classes, lower the error, as the classification can be done better.

However, it is also interesting to see how this color reductions affect images of a test. Figures 8 and 9 show an image painted with the color-vectors of different sized SOMs. Figure 8 shows what we consider a ”good case”, in the sense that the reduction of colors does not affect the color perception substantially. On the other hand, Figure 9 depicts a ”bad case”, where a color is so particular that it is lost in the color reduction.

(a) Original (b) 4 neurons (c) 9 neurons

(d) 25 neurons (e) 100 neurons (f) 400 neurons

Fig. 8. Examples of Color Reduction with Different SOM Size. Good Case.

(a) Original (b) 4 neurons (c) 9 neurons

(d) 25 neurons (e) 100 neurons (f) 400 neurons

Fig. 9. Examples of Color Reduction with Different SOM Size. Bad Case.

C. Neural Network training and calibration

An important factor in the system performance is the correct selection of the NN configuration. The main configuration parameters are the number and size of the hidden layers, and the activation function. Also, the input and output sizes must be selected. In our case, the input is given by the patch size, and the output by the size of the SOM.

Preliminary tests showed that increasing the SOM above some moderate size do not increase the quality of the colored image noticeably. At the same time an increment in the SOM's size has a heavy impact on the training as it relates directly to the output space. This leads to a much slower learning. Hence, we selected to use a SOM of 9 neurons.

As the NN is used to perform classification, the *rectified linear unit* activation function was used. The NN model used was Multi Layer Perceptron Classifier, with learning rate of 0.001. Several hidden layer configuration were tested: single layer with 10, 50 and 100 neurons; and two layers with 50, 100, 200 and 300 neurons each. We refer to these configurations as *10*, *50*, *100*, *50+50*, *100+100*, *200+200* and *300+300*, respectively.

To evaluate the different NN configurations we performed a grid search, testing each configuration against each patch size. We trained over the dataset cross-validating with 50000 as the training sample size and 5000 as the test sample size. To compensate for the varying frequency of labels in the training output we used *Stratified Shuffle Split* (with 10 splits), which attempts to maintain an homogeneous representation of labels in each split.

The results can be seen in Table II. For each patch size and Hidden layer configuration there are two number shown: the training score, and the test score. A big difference between both could indicate over-fitting. This is the case with the two biggest networks, the *200+200* and *300+300* configurations. It must be noticed that *50+50* and *100+100* configurations achieve a lower score over the training set, but actually perform better over the test set which is the true indicator of the NN predictive power.

In the one-layer configurations, there are little differences between the *50* and *100* scores, indicating that there is little performance to be further extracted. The two-layer configuration perform better, but the performance decreases with configuration bigger than *100+100*, as indicated.

Bigger patch sizes do not improve predicting performance. In fact, *11x11* patches perform noticeably worse.

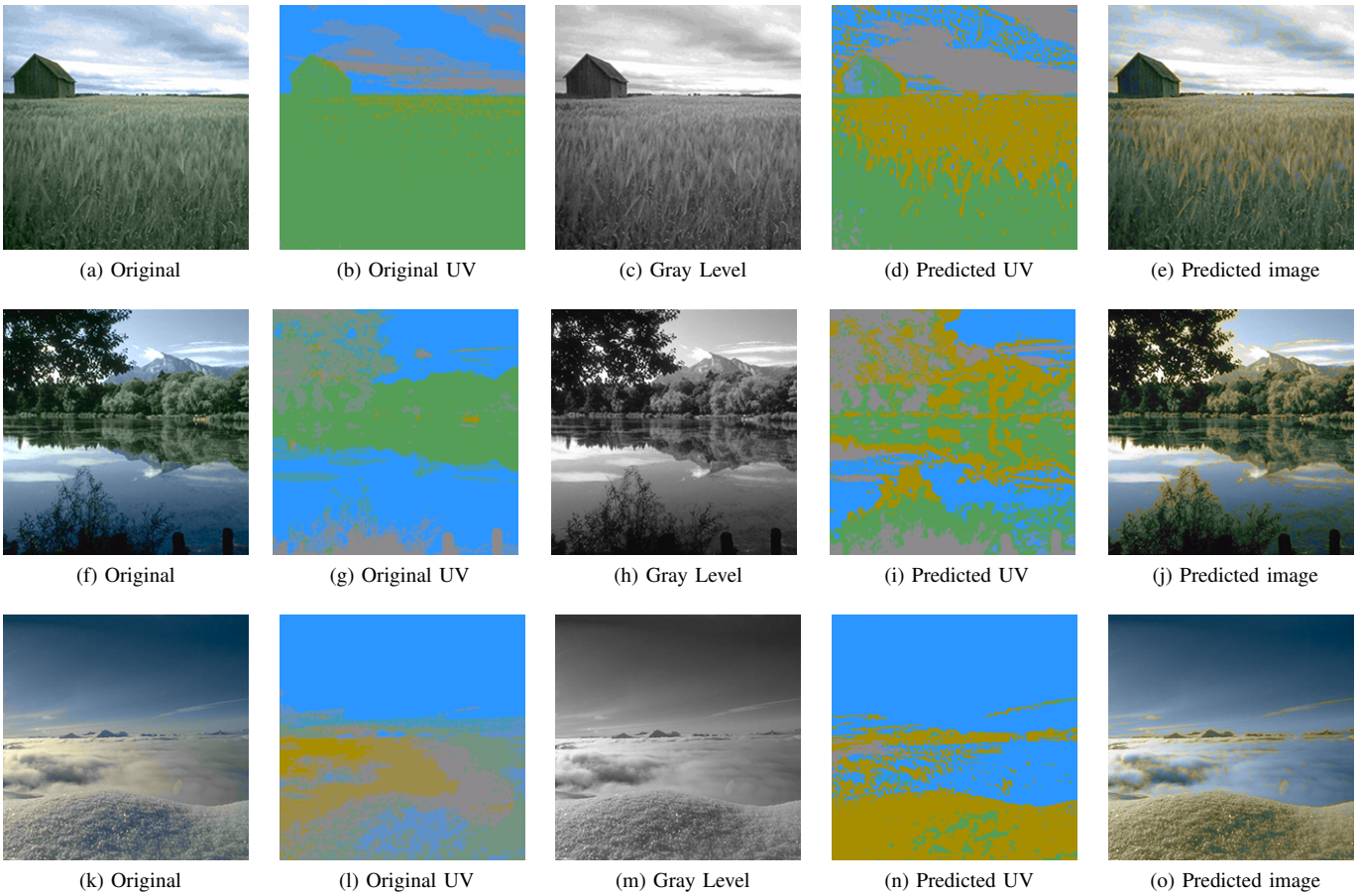
As conclusion, the best balance of performance and NN size is achieved with *3x3* patches and a NN with two layer of 50 neurons each. Nevertheless, there is visual differences between the images generated using *3x3* and *5x5* patches. The former has higher chroma noise in some regions, where adjacent pixels are assigned different colors. The later achieves a smoother image. Thus, *5x5* is a valid option also, using more computational power but generating slightly more appealing images.

D. Results

Some predictions can be seen in Figure 10. The first column is the original image. The second column shows the colors in the image displaying the U and V channels and a fixed 50% value for the L channel. The third column is the image in black and white, this is, only the L channel. The fourth column is the colors as predicted by the NN from the gray-level image (again the U and V channels with a fixed 50% L channel). The last column is a full predicted image using the predicted

TABLE II
TRAINING AND TEST SCORES FOR TRAINING THE NN WITH DIFFERENT PATCH AND HIDDEN LAYER SIZES

Patch size	Hidden layers						
	10	50	100	50+50	100+100	200+200	300+300
3x3	0.376 / 0.371	0.383 / 0.379	0.387 / 0.378	0.419 / 0.409	0.429 / 0.405	0.474 / 0.402	0.497 / 0.399
5x5	0.349 / 0.346	0.370 / 0.367	0.373 / 0.366	0.423 / 0.407	0.444 / 0.403	0.513 / 0.382	0.549 / 0.384
11x11	0.312 / 0.309	0.353 / 0.347	0.345 / 0.328	0.401 / 0.383	0.416 / 0.383	0.433 / 0.376	0.417 / 0.365



(a) Original (b) Original UV (c) Gray Level (d) Predicted UV (e) Predicted image

(f) Original (g) Original UV (h) Gray Level (i) Predicted UV (j) Predicted image

(k) Original (l) Original UV (m) Gray Level (n) Predicted UV (o) Predicted image

Fig. 10. Examples of predictions.

U and V channels combined with the gray-scale L image from the original.

The image 10e is reasonably good. Comparing the colors in figures 10b and 10d we can see it correctly separated ground level from sky, with some differences. The clouds are more desaturated (gray clouds vs. blueish), and applied a yellowish color to the crop. Both changes are reasonable guesses to the original colors.

In Figure 10j can be seen how the predictor correctly tinted the sky reflection blue. It again preferred a yellowish tint. Some color bleeding into adjacent areas can be observed.

Figure 10k is a very difficult picture. It has a very rare view of a cloud bank from above. The predictor incorrectly tinted the cloud blue (as either water or sky). It also tinted the snow green/yellow. This suggest that snow scenes should be trained separately.

IV. CONCLUSIONS AND FUTURE WORK

In this work we design and develop a methodology, strongly based on neural networks, to colorize gray level images. Our proposal consists on a series of steps to finally obtain a trained neural network which predicts the color of a gray level pixel. The methodology joins two main ideas from previous works: color reduction using vector quantization and using a group of neighboring pixels to predict the color of a single pixel.

We implement the methodology and test it with an open set of images. In our opinion the results are good and promising although the prediction scores obtained are below the 50%. To a human eye the images obtained show good results and have colors close to the intuition. As the prediction highly depends on the set of training images, the images more similar to the training set show better results than those more different. This proposal is a work in progress and we are convinced that with

more extensive tunning and precise configuration the results will improve. Nevertheless, we believe that the methodology proposed is an interesting first step towards colorizing images without the need of heavy image processing algorithms.

In addition to better tunning, future work includes more tests with different patch sizes and SOM sizes. In particular, the size of the SOM is an important aspect to improve, as the colors obtained depends on the size of the SOM used. Our results are for small SOMS and the images obtained lack of a full color range, but bigger SOMs makes the neural network much more difficult to train. The presented experiments and results are for a single use case, experimenting with more image sets of different classes is another aspect to improve.

Finally, a possible improvement to the methodology to try as future work is to consider the position of the pixel as a a feature to train the NN.

REFERENCES

[1] W. Markle and B. Hunt, "Coloring a black and white signal using motion detection," Jul. 5 1988, uS Patent 4,755,870.

[2] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 277–280.

[3] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM transactions on graphics (tog)*, vol. 23, no. 3. ACM, 2004, pp. 689–694.

[4] R. Ironi, D. Cohen-Or, and D. Lischinski, "Colorization by example." in *Rendering Techniques*, 2005, pp. 201–210.

[5] L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," *IEEE Transactions on Image Processing*, vol. 15, no. 5, pp. 1120–1129, 2006.

[6] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 415–423.

[7] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," *arXiv preprint arXiv:1603.06668*, 2016.

[8] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *arXiv preprint arXiv:1603.08511*, 2016.

[9] J. Hwang and Y. Zhou, "Image colorization with deep convolutional neural networks," Stanford University, Tech. Rep., 2016. [Online]. Available: http://cs231n.stanford.edu/reports2016/219_Report.pdf

[10] J. Yoo and S.-Y. Oh, "A coloring method of gray-level image using neural network," in *Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems*, vol. 2, 1997, pp. 1203–1206.

[11] *ISO/CIE 11664-5:2016 - Colorimetry – Part 5: CIE 1976 L*u*v* colour space and u', v' uniform chromaticity scale diagram*, International Organization for Standardization, Geneva, Switzerland. Std., 2016.

[12] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on communications*, vol. 28, no. 1, pp. 84–95, 1980.

[13] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] V. Moosavi, S. Packmann, and I. Vallés, "SOMPY: A Python Library for Self Organizing Map," 2016. [Online]. Available: https://github.com/sevamoo/SOMPY

[16] M. Richart, J. Visca, and J. Baliosian, "Source code for image colorization with neural networks." 2017. [Online]. Available: https://github.com/mrichart/NNcoloring

[17] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1011139631724