CSCI 5622 Final Report: Image Colorization

Chandler Baggett
Department of Computer Science
University of Colorado Boulder
chandler.baggett@colorado.edu

Poorwa Hirve
Department of Computer Science
University of Colorado Boulder
poorwa.hirve@colorado.edu

Prathyusha Gayam
Department of Computer Science
University of Colorado Boulder
prathyusha.gayam@colorado.edu

Rakesh Shivanand Margoor
Department of Computer Science
University of Colorado Boulder
rakesh.margoor@colorado.edu

Tarunianand M.
Department of Computer Science
University of Colorado Boulder
tamu0376@colorado.edu

Abstract

Automatic image colorization is the colorization of grayscale images without user intervention. The lack of techniques that can be used to colorize images without prior knowledge is an important challenge. This project aims at developing and training different models that can be used to colorize grayscale images, with no prior knowledge of the colors involved. We have implemented and analyzed two machine learning techniques, Support Vector Machines and Convolutional Neural Networks for this purpose.

***Keywords* —** Support Vector Machines (SVM), Convolutional Neural Networks (CNN), Keras, VGG network

1 Introduction

Images in grayscale are widely produced in several fields, like weather forecasting, electron microscope images, MRI images, satellite imagery and in general photographs from the past. The requirement for colorization of images is important for research and scientific interpretation. We have tried to come up with some models that can colorize these images. We began with SVM to colorize every single pixel, into RGB layers. Further we used CNN, to run a model that converts grayscale images into color by using different training images and obtaining colorized outputs. Finally, we tried to compare these results and analyze the errors obtained. This paper discusses each of these methods in detail, and the comparative results. Grayscale images are widely used in image processing. They scale the RGB colors to roughly one-third the size of the original image. Thus the storage of the grayscale image is less expensive. In this process, however, the original image is lost. Our project aims to reclaim the original image properties using Machine Learning.

An interesting application of our project is its role in black and white movies, as well as reclaiming the colors of black and white photographs. Subreddits like /r/colorization involve people who wish to

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

colorize black and white photos as before colored photography and movies came into picture, only black and white images and films existed. In addition to these, MRI scans and satellite imagery are areas in which this can be applied. Also, it allows better interpretation of modern grayscale images such as those from CCTV cameras or astronomical photography. Manual way of colorizing images would take a lot of time and if it involves colorizing a video it would be a Herculean task. Automating the conversion of grayscale images to color using machine learning techniques would save a lot of time and effort.

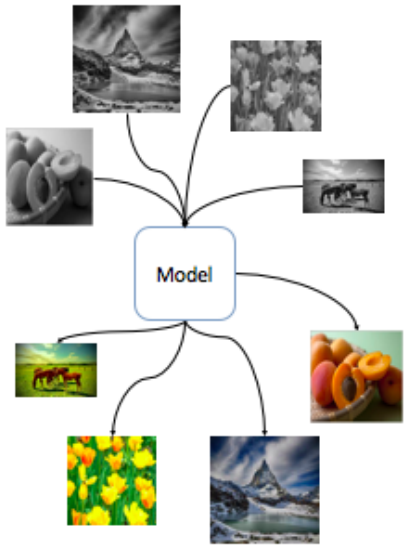

Figure 1: Model

2 Literature Survey

For the task of image colorization, there has been extended research, each experimenting with different machine learning models, starting with logistic regression [1] to the more complex deep learning models. Frenette had experimented with the linear regression model, with a One vs. All model, as opposed to true multinomial models.

There was also significant progress made using support vector models, combined with SIFT feature extraction [1], however both models used single training images to obtain results. There was a reduction in overall brightness and intensity of the image, otherwise producing a valid colorized output. Zanoci et al. used SURF and FFT as features in a support vector machine model [2], to colorize images. The model was tested using a single training image, and tested with the same and different test image. These involved colorizing each pixel's red, greeen and blue layers, converted to Lab space. Kabirzadeh et al. used some edge and contour extraction in addtion to the SVM model, and trained a initial image pixels using K means clustering to quantize the color space. [3][4]

Baldassarre et al. used a fairly new approach of using convolutional neural networks to colorize grayscale images. [5] The model used high level feature extraction using a pre-trained Inception-ResNet-v2 model for enhancing the colorization process. They had analysed a colorization architecture based on convolutional neural networks. The model's performance was evaluated using a public survey of acceptance and the output of the same was dependent on the training images that were used. Hwang et al, in [6] used CNNs with multilayer aggregation and residual encoders, while experimenting with different color spaces to obtain similar results. A common field of concern was the colorization performance on independent objects, that did not follow a specific coloring pattern. [5][6]

2.1 Dataset

Our data is categorized into the following:

- Faces (color FERET database from NIST.gov)
- Fruits (ImageNet)
- Animals (ImageNet)
- Scenes (ImageNet)
- Flowers (ImageNet)

2.2 Machine Learning Methods

Predicting the actual color of an image automatically, involves necessary learning from different train images. The learning can be done by using different features of an image such as object detection, frequency domain features etc which are explained more in the next section. Hence building a model which predicts the color would definitely involve supervised learning. Based on literature survey and research, we think some of the popular supervised machine learning algorithms that can be applied to this model are discussed below.

- Support Vector Machines
- Convolutional Neural Networks

3 Project Timeline

3.1 Data Preprocessing

The first step was to obtain data and perform certain operations on it. The SVM training images were of size 100x150 px and for CNN, we decided on a certain size 256x256 px. For faces, we obtained permission for the color FERET database of faces and resized the images. We wrote a script in order to obtain data from ImageNet urls and resized them.

Next, we divided each categorical data into an 80-20 split for train and test. The test images had their respective gold images. Training images for each category were around 1600, with 400 test images according to the 80-20 split.

3.2 Machine Learning Approaches

The next step was to short-list machine learning approaches to this problem. We decided to use Support Vector Machines as a baseline, followed by Convolutional Neural Networks. We split up our team into two - Chandler and Rakesh for SVM, and Poorwa, Prathyusha and Taruni worked on CNN.

3.2.1 Support Vector Machines

Based on the literature survey, we started out with SVM, where SVM can be used to predict the color of a pixel in each channel. It provided us a good baseline. Using multiple features, we were able to see which features worked best and which features did not contribute as much to the colorization.

3.2.2 Convolutional Neural Networks (CNN)

To get started with our model, we referred to other implementations. The most commonly used architectures were the VGG network and ResNet.

We observed that many pre-defined models were available for testing. Using these models, we found out that on our data, the VGG network performed the best.

Implementations like [7] required the use of Caffe. However, as it is difficult to install and set up, we decided to go ahead with Python's Keras package with Tensor-flow in the backend

Then, we experimented with various architectures, noting down which combinations worked best.

3.3 GPU

As each CNN epoch was taking 15 minutes to run on a CPU, in order to train on sufficient data, we got access to a Graphical Processing Unit. On this, every epoch ran for an average of 6 seconds.

Configurations:

Model : NVIDIA GeForce GTX 1080
Architecture : Pascal
Frame Buffer : 8 GB GDDR5X
Memory Speed : 10 Gbps
Boost Clock : 1733 MHz

3.4 Evaluation

After gathering the results for each, we performed error analysis and evaluated the model, described further in section 6.

Figure 2: Gray | Result | Gold

4 SVM

Our SVM used feature extraction methodologies such as SURF (Speeded-Up Robust Features), FFT (Fast Fourier Transform), DST (Discrete Sine Transform), and DCT (Discrete Cosine Transform) so that we may gain relevant data in order to draw a comparison between them. The methodologies were implemented in our code by using built-ins found in OpenCV (Open Source Computer Vision Library).

The training and the testing flow is described in Figure 3.

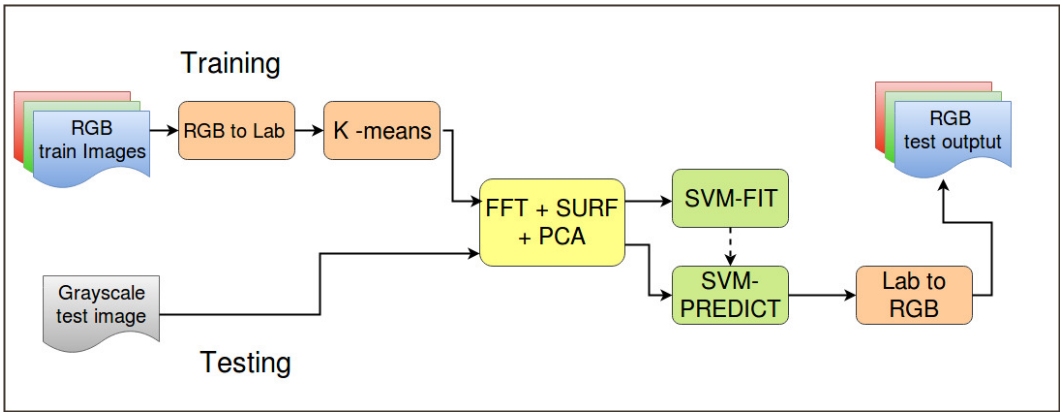

Figure 3: SVM Training

The brief explanation on the flow is described below

- RGB to Lab: RGB has 3 channels, which results in 256x256x256 different types of colors(8 bit image). Converting it into Lab reduces to 256x256x1 as L represents intensity which can be directly used from the test image.
- K-means: Through K-means we reduce the number of labels from 256x256. Our optimum k = 32.
- Feature extraction: We initially tried using FFT and SURF features. Through FFT, we access the geometric characteristics of the spatial domain training image and the SURF features provide the key points in an image, essential in identifying similar patterns to help SVM predict the color based on similar patterns (pattern detection rather than object detection). The descriptors from SURF represents the variation of image intensity (see Figure 4).
- PCA: To reduce training time, we used PCA(Principal Component Analysis) for reducing the feature dimension. The number of principal components we used are 50. This significantly decreased our SVM training time.
- SVM: We have used one vs one approach since having k (kmeans) classifiers is computationally better than k*(k-1)/2 through one vs rest. Time computation for one vs one was 8 times less compared to one vs rest.
- Lab to RGB: Since the input fed to the SVM is Lab, the output from SVM is converted back to the RGB color space.

We also implemented features including DST as well as DCT. While both are similar in origin as well as in the structure of the images that they produce as methodologies of mediation in the processing of images, there are some slight differences that may be relevant.

In the case of image processing, DCT and DST transform an image from the spatial domain to the frequency domain. For example, given some image as a real function of i, j f(i, j), DCT would transform it to some real frequency domain function of u, v F(u, v). Given some 2-Dimensional (N by M) image, DCT can be defined by the following equation:

$$F(u,v) = (2/N)^{1/2}(2/M)^{1/2} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos[(\Pi.u)/(2.N)(2i+1)]cos[(\Pi.v)/(2.M)(2j+1)].f(i,j)$$

And, for DST:

$$F(u,v) = (2/N)^{1/2}(2/M)^{1/2} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).sin[(\Pi.u)/(2.N)(2i+1)]sin[(\Pi.v)/(2.M)(2j+1)].f(i,j)$$

4.1 Feature Evaluation

To analyze the above model, we trained SVM through several images. The impact of adding more images which in turn is adding more features is shown in Figure 5. With one training image with no features for sky, the output image's sky is not colored, however, training with another image with appropriate features for sky, the model was able to predict blue to the sky.

4.1.1 FFT features

Since, we are using SVM to predict the color of a pixel, we have to build features for each pixel in an image. FFT is applied on a fixed-size window around every pixel of an image. Choosing the right window size was one of the challenge. We trained our SVM model on certain categorical images where a size of 13 pixels gave good results. This feature is important as the variation of intensities is explained better in frequency domain than spatial domain.

4.1.2 SURF features

The keypoints and the descriptors are essential to identify certain pattern especially when the object is translated, rotated or scaled. For example, consider the Figure /refsurf where the image is rotated by 90 degrees. Even though the image is rotated, through the keypoints we can identify or match the parts of the horses from the original image. Hence, SURF features have great impact when there are similar objects in the train and test images.

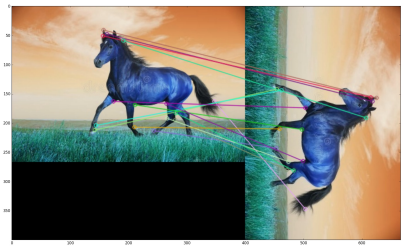

Figure 4: SURF Features

As an illustration, consider Figure /refsurfimpact where the test image is scaled and rotated version of one of the train image. The predicted image with only FFT as features is not colored completely, especially the horse. However, adding SURF features, the horse is colored brown, this is the result of keypoints and descriptors of SURF. It has less impact when the objects do not match significantly which is shown in Figure 6 .

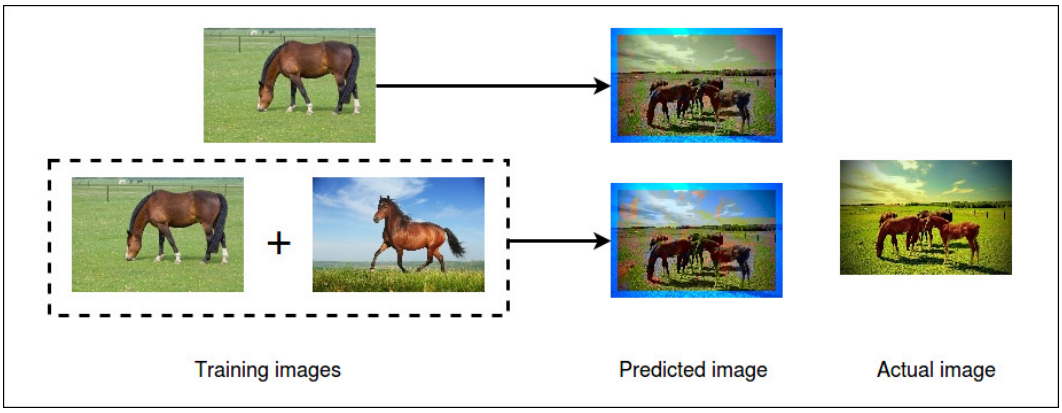

Figure 5: Results of adding features

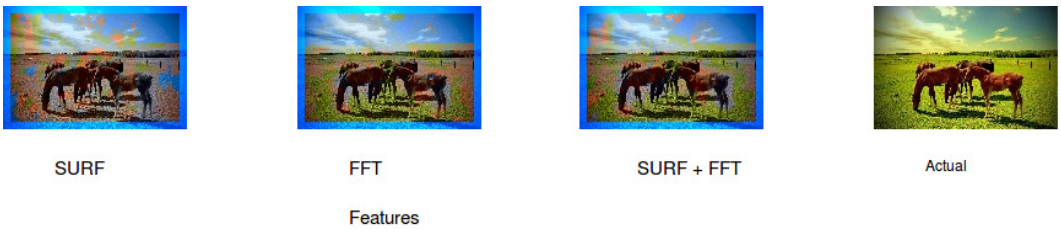

Figure 6: Impact of features

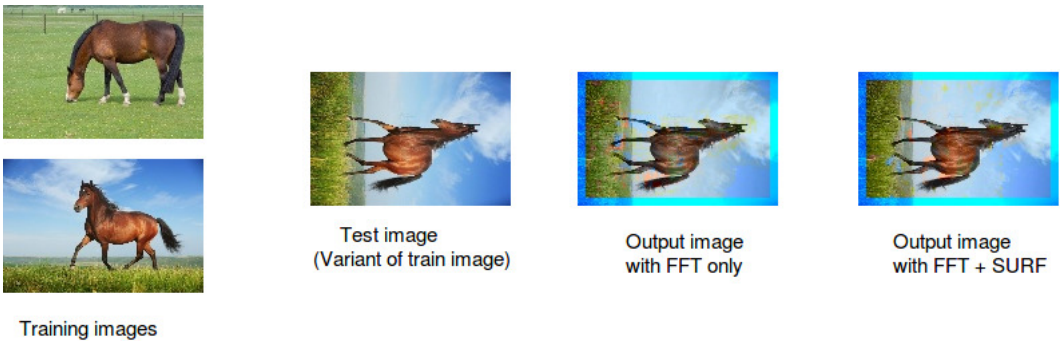

Figure 7: Impact of SURF features

4.1.3 K-means

The value of k is equal to the number of classes for the SVM. The impact of k on the output image is shown in Figure /refkvariation. With k=2, the horses are colored green having only two labels blue and green. The output image looks good when k is equal to 5. However, this cannot be generalized as it depends on the number of colors present in the actual train and test images. In our training and test data, we had less number of color shades (mainly, blue, green and brown) and hence lower value of k gives better results, due to the fact of having fewer classifiers. In general, the value of k should be chosen based on how discretized coloring is required and the number of color shades the images have.

Figure 8: Impact of k(k-means)

5 CNN Architecture

During the train time, our program reads images of size 256x256x3 corresponding to R(Red), G(Green) and B(Blue) channels in the RGB color space. The images are converted to CIE Lab color space and the luminance channel L is fed as input to the model. The U and V channels are extracted at the output layer.

During the test time, the model accepts 256x256x1 black and white image. It generates two arrays of dimensions 256x256x1 corresponding to U and V channels. These predicted image is the concatenation of L, U and V channels.

Experimenting with VGG and ResNet architectures, we got the best results with the VGG architecture.

Convolution and Up-sampling layers:

Convolution and Up-sampling layers are the major components of VGG network. Convolution layers are mainly used in extracting the conditional features. Additionally, stacking multiple convolution layers with strides 2, to downsize the data shape, works as a data encoder. Up-sampling

layers are then used to up-size the data shape and works as a decoder of the data representation. We used the encoder-decoder model with batch normalization.

We designed three VGG networks and based on the performance we labeled them as bad, good, and best models. The bad model contains 5 layers of convolutional layers arranged sequentially. The good and best model has different combinations of convolution and up-sampling layers, the optimizer used is RMSProp. To eliminate the problem of over fitting our best network contains batch normalization layers. Adding dense/fully connected layers didn't improve any performance in color prediction. So, to reduce the number of parameters and improve the speed we eliminated dense layers from our network.

Figure 9: CNN architectures(Bad, Good and Best)

Figure 10: Image Colorization with bad, good, best models and Real image

6 Error Analysis

6.1 Data Imbalance

As mentioned in section 3.1 and figure 1, before categorizing the data, we ran our model on all types of images together. However, it resulted in mis-classifications, for example in the figure 11, the sky is shown to be red instead of blue, and in the figure 12, some parts have been colored red, suggesting that red colors were dominating over the real colors in the training set.

Figure 11: Predicted Matterhorn

Figure 12: Predicted Blackberries

Therefore, we categorized the images as mentioned in section 2.1

6.2 Structural SIMilarity Index (SSIM)

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

The SSIM index is formulated as above, where c1 and c2 are constants for when the mean and variances approach 0. It is a measure for the similarity between the two images, from the range 0 to 1, 1 meaning that they are identical. Given a gold file, we can measure the similarity of the predicted image. We aimed to maximize the SSIM index as much as possible while avoiding overfitting.

6.3 Root Mean Square Error (RMSE)

The RMSE of each image was calculated as follows:

1. R, G, B channels were obtained separately for gold and result images.
2. RMSE was found for each channel
3. Average over all was taken to calculate Average RMSE

6.4 Euclidean Distance

The Euclidean Distance per pixel, that is, the L2 norm of the result image with respect to the gold image was found.

Figure 13: SVM output

Figure 14: CNN output

Table 1: Evaluation of SVM vs CNN

	SVM	CNN
SSIM	0.62	0.74
RMSE	10.17	7.69
L2	727.66	570.6

6.5 Confusion Matrix

We obtained a confusion matrix from a single image. By getting sub-images of small sizes, we found the dominant color in the gold image, compared it with the dominant color in the result. Comparing the true R, G, B channels to the predicted R, G, B channels, the confusion matrix was as follows:

Table 2: SVM

	R	G	B
R	7	32	0
G	9	100	2
B	0	0	0

Table 3: CNN

	R	G	B
R	25	122	2
G	13	760	0
B	5	93	4

6.6 Progress of the Model

We decided on 1000 epochs by trial and error. Based on the intermediate output, we can see how the neural network progresses at 10, 20, 70, 300, 600 epochs, with the total coming up to 1000.

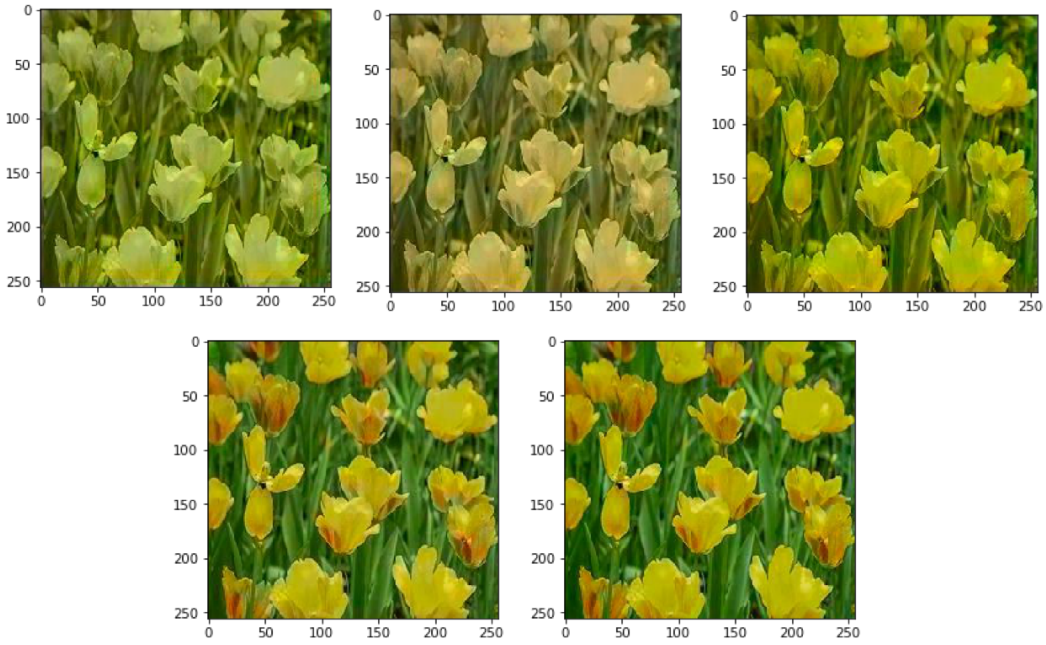

Figure 15: Progress throughout the 1000 epochs

Figure 16: Training and Validation Errors Vs. Epochs

7 Conclusion and Future Scope

We started the project with the goal of creating a model that would colorize images. By exploring various machine learning methods, through our efforts, we concluded that CNN is always better than SVM for the purpose of image colorization. Although a generalized model did not work well with uncategorized images, our best model showed a good performance with categorized images. We were successful in producing images which were nearly similar to the original one, with relatively small images of 256x256 px.

While we have realized a certain extent of colorizing images, we do have some issues to be resolved. The results seem to be convincing in some cases like that of nature and objects that have a specific or predetermined color. However, the model that has been developed has some changes in the overall brightness and intensity. This in turn is scope for improvement. Finally, to have these results be trained on several categories and further tested on a mixed test set, can yield some different results.

We can see the colorization model being used as a method to colorize old films (grayscale and sepia). This application can be developed by isolating every frame of the movie, and running the

same on the model. The results would have to depend on the results obtained from the previous frames, to ensure uniformity in coloring. Further, when this type of model can be further developed to support real time colorization, there would be a larger plethora of applications, like colored MRI scans.

References

[1] https://cs.uwaterloo.ca/~zfrenett/CS886-Project.pdf

[2] http://cs229.stanford.edu/proj2015/163_report.pdf

[3] http://cs229.stanford.edu/proj2013/KabirzadehSousaBlaes-AutomaticColorizationOfGrayscaleImages
pdf

[4] Zarezadeh, A & Alizadeh Ghazijahani, Hamed & Judi, Arash. (2012). Color Image Segmentation with K-Means Clustering and Edge Detection Methods.*ISCEE*

[5] https://arxiv.org/abs/1712.03400

[6] http://cs231n.stanford.edu/reports/2016/pdfs/219_Report.pdf [7] R Zhang, P Isola, AA Efros, Colorful Image Colorization, - *European Conference on Computer Vision*, 2016

[8] https://arxiv.org/abs/1705.02999

[9] A Deshpande, J Rock, D Forsyth - Semantic colorization with internet images, *Proceedings of the IEEE International Conference on Computer Vision*, 2015

[10] A Deshpande, J Lu, MC Yeh, DA Forsyth, Learning diverse image colorization, *CoRR, abs/1612.01958*, 2016

[11] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, Huang Zhiyong, Learning large-scale automatic image colorization, *20th ACM international conference on Multimedia*, 2011

[12] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, Yann LeCun, *arXiv preprint arXiv:1312.6229*, 2013

[13] Welsh, T., Ashikhmin, M., Mueller, K.: Transferring color to greyscale images. In: SIGGRAPH 2002: Proc. of the 29th annual conf. on *Computer graphics and interactive techniques, pp. 277–280. ACM Press, New York* (2002)

[14] Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954. Springer, Heidelberg (2006)

[15] http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html