A Novel Approach for Automatic Image Colorization

[1]Akunuri Satyam, [2]A.Chandu Naik, [3]B.Rajendra Prasad

[1]Assistant Professor, [2]Assistant Professor, [3]Assistant Professor
[1] Department of CSE,
[1]SreeNidhi Institute of Science and Technology, Hyderabad, India

Abstract : ***The central problem of Automatic Image Colorization revolves around our ability to come up with a clear way of computing the Colored Image from the given Black and white Image. So, there won't be any accurate answer for the Black and White Images if we give wide range of Black and White images. So, many algorithms came up to solve this problem. There are solutions to this problem through manually colorizing each and every pixel of the image. By doing manually, we can maximum try to make the Black and White Image to a good looking colored Image. But it takes very long time. Before deep learning arrived at the scene, researchers had been handcrafting methods to colorize each pixel of the Image. Even in today's research of style transfer using deep learning there are high impact papers proposing new ways of using a neural network to Colorize the given Black and White Image. This system can be used as an application for colorizing the user given Black and White images.***

IndexTerms - Image Colorization, Tensor Flow, Convolutional Neural Networks, Gray Scale images

---

I. INTRODUCTION

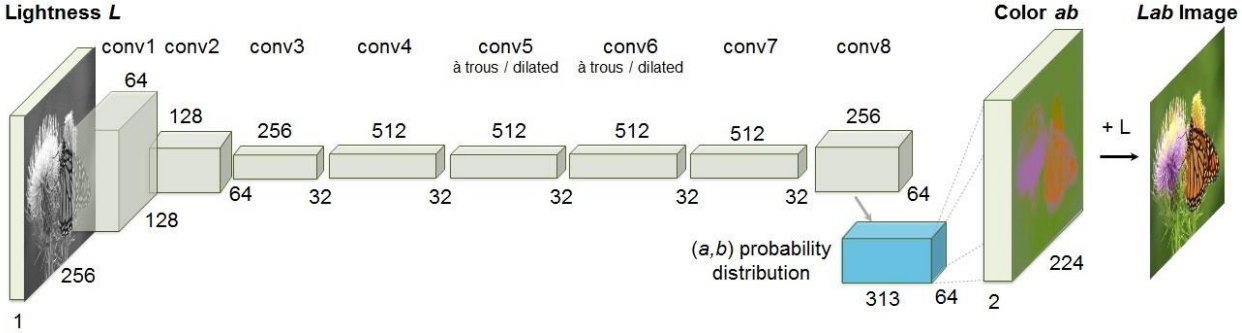
Image Colorization usually refers to Colorizing the Grayscale Image to a color image. Here, the color image may not be the exact ground truth image but it can be called as a predicted color image of a Black and white Image. On this Image Colorization process many computer scientists have worked for years. With the advent of Convolution Neural Networks, the rise of Image Colorization has begun. Usually this Image Colorization is done manually. For example, if we want to convert a grayscale Image to Color Image we have to fill the colors to each part of the image using software. These software include Adobe Photoshop, GIMP for Linux OS etc So, by doing in this way, it takes much time since we have to visualize and paint each part of the image with a specific color. Similarly, if we apply Black and white movie, we should colorize each and every frame of the image. Usually a Black and White film consists of 150 minutes, which means 9000 seconds. For each second there will be 24 frames which mean 24 images which form a single second video. So we have to colorize 9000*24 which is 216000 images. So, we have to manually colorize 2,16,000 images to colorize a black and white movie to color movie. This usually takes 3-4 months. So, there was a thought to automate it. It means, we have to build a system where a Black and White image is given as input and the system should generate a Color Image.

II. RELATED WORK

This paper deals with colorizing the given Black and White image to various colorized image producing the result at faster rate. Compared to other systems, the proposed approach colorizes the input image at a faster rate due to its learned representations. We can even train many Black and white images and even can edit the videos based on Predefined set is a dataset which containing almost 1000s of images with different expressions. The machine will learn to detect color using the given dataset. So, the proposed system to predict the color of the given Black and white Image based on the trained dataset.

Given a grayscale photo as information, the system assaults the issue of fantasizing a conceivable shading rendition of the photo. This issue is unmistakably under constrained, so past methodologies have either depended on huge client collaboration or brought about the saturated colorizations. We propose a completely programmed approach that produces dynamic and practical colorizations. We grasp the basic vulnerability of the problem by acting it like an order undertaking and utilize class-rebalancing at preparing time to build the assorted variety of hues in the outcome. The framework is executed as sustain forward finish in a CNN at test time and it is prepared on finished a million shading pictures. We assess our calculation utilizing a "colorization Turing test," requesting that human members pick between a created and ground truth shading picture. Our technique effectively fools humans on 32% of the trials, fundamentally higher than past strategies. Besides, we demonstrate that colorization can be an intense appearance assignment for self-regulated component getting the hang of, going about as a cross-channel encoder. This approach brings about best in class execution on a few element learning benchmarks.

Figure 2.1 The basic working of proposed system



III. COLOR PREDICTION OF THE IMAGE

Now , we need to be modeled colors and local descriptions of grey scale images, we can start the image colorization problem. From a set of examples of color images, and a new grey scale image I to be colored, we would like to extract knowledge from the learning set to predict colors C(p) for the new image.

3.1 Need for multimodality

One should state that this problem is in simple machine learning terms,learn the function which is associate to any local description of grey scale patches the right color to assign to the center pixel of the patch. This can be done by kernel regression tools such as Support Vector Regression (SVR) or Gaussian Processes . There is an intuitive reason why this would perform poorly. Many objects can have different colors, for instance balloons at a fair could be green, red, blue, etc., so that even if the task of recognizing a balloon was easy and that we knew that we should use colors from balloons to color the new one, a regression would recommended the use of the average value of the observed balloons, i.e. grey. The problem is how ever not specific to objects of the same class. Local descriptions of grey scale patches of skin or sky are very similar, so that learning from images including both would recommend to color skin and sky with purple, without considering the fact that this average value is never probable. We therefore need a way to deal with multi-modality, i.e. to predict different colors if needed, or more exactly, to predict at each pixel the probability of every possible color. This is in fact the conditional probability of colors knowing the local description of the greyscale patch around the pixel considered. We will also be interested in the confidence in these predictions in order to know whether some predictions are more or less reliable than others.

3.2 Probability distributions as density estimations

The conditional probability of the color ci at pixel p knowing the local description v of its greyscale neighborhood can be expressed as the fraction, amongst colored examples ej = (wj , c(j)) whose local description wj is similar to v, of those whose observed color c(j) is in the same color bin Bi . We thus have to estimate densities of points in the feature space of grey patches. This can be accomplished with a Gaussian Parzen window estimator:

$$p(c_i|\mathbf{v}) = \Big(\sum_{\{j\,:\,c(j)\in B_i\}} k(\mathbf{w}_j, \mathbf{v}) \Big) \Big/ \sum_j k(\mathbf{w}_j, \mathbf{v})$$

where $k(w_j , v) = e^{-(wj-v)2/2\sigma 2}$ is the Gaussian kernel. The best value for the standard deviation σ can be estimated by cross-validation on the densities. With this framework we can express how reliable the probability estimation is: its confidence depends directly on the density of examples around v, since an estimation far from the clouds of observed points loses signification. Thus, the confidence in a probability prediction is the density in the feature space itself:

$$\mathbf{p(v) \propto \sum_j k(w_j , v)}$$

In practice, for each pixel p, we compute the local description v(p), but we do not need to compute the similarities $k(v, w_j)$ to all examples in the learning set: in order to save computational time, we only search for the K-nearest neighbors of v in the learning set, with K sufficiently large (as a function of the σ chosen), and estimate the Parzen densities based on these K points. In practice we choose K = 500, and thanks to fast nearest neighbor search techniques such as kD-tree6 , the time needed to compute all predictions for all pixels of a 50 × 50 image is only 10 seconds (for a learning set of hundreds of thousands of patches) and this scales linearly with the number of test pixels

IV. TENSOR FLOW

TensorFlow is Library developed by Google recently. They developed it mainly for Machine Learning Applications. In order to simplify the code development by a programmer, they created TensorFlow which contains many packages. It provides methods on tensors and computes their derivatives. Data graphs are the main and crucial part of TensorFlow where the data is stored in graphs. By using this graphs we can develop back the whole code. It means that, when we usually write code for any application in TensorFlow, it generates dataflow graphs. These are very important. Usually every mathematical operations are made as tensor objects. Now if the code needs to add any two numbers then based on these graphs the addition is done. Apart from the multiplication and division is also done based on the code. So, these tensors are important. Why the name TensorFLow ? Becauseof the main reason of flowing of tensors in a dataflow model. The data is transmitted in a special fashion. So in the below figure we can see the source code and flowchart specific to it. Here in the code, there are constants. Usually constants take specific values and then these values are constant and are can't be changed. Similarly, we have variables and Placeholders. Variables are changed in between based on the code. But these placeholders are very crucial where the values are given by user. Since our Project is about taking the Black and white images as input from user, the values of pixels of these images are usually taken as placeholders and then given as input.

Here we took some variables like x,y,z where x and y are constants and are usually fixed values which can't be changed. These are usually used for Learning Rate and bias values in machine learning Algorithms. Using mul( )function we are multiplying x and y. Everytime we run session because to create dataflow. Session function is very crucial since the code runs only after creating sessions. After that, the output is generated as it is seen in the image. The dataflow diagram can also be visible and can download the diagram for every TensorFlow program.

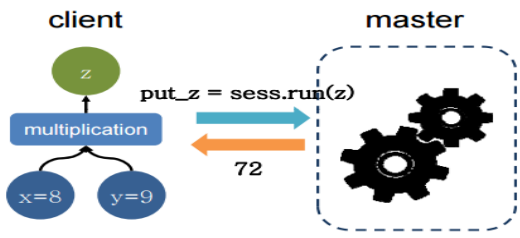
Figure 4.1. A TensorFlow program and it's datagraph

source code

```
import tensorflow as tf

x = tf.constant(8)
y = tf.constant(9)
z = tf.mul(x,y)

sess = tf.Session()

out_z = sess.run(z)

print('out_z: %d' % out_z)
```

output

```
out_z: 72
```



4.1.ConvolutionalNeuralNetworks

Convolutional Neural Networks, famously known as CNN, is the most famous process model used in Machine Learning right now. We have listened about deep learning which is the repeated application of these Convolutional Neural Networks. These are mainly used for Image Problems. Convolutional Neural Networks are developed during 1990s and came to real time application recently. Because of its in depth intuition of understanding and analyzing an image, CNN was very famous. Usually for MNIST dataset which is used for Handwritten

digit recognition, using these CNN models, scientists achieved state of art around the accuracy of 98%. Apart from that there are very developed in other applications such as Object Detection.

V. ARCHITECTURAL DESIGN

Architectural Design is very crucial since the main, final product depends on the architecture which we follow. It covers all the basic information and helps the developers to develop the code in the fashion which user wants or here we can say that the Architectural design is to develop the final project well. As usual, in the systems requirements, the systems analyst defines requirements that specify what the proposed system is to accomplish. The analysis phase looks at different alternatives from a system's point of view. In the architecture where the user first opens IDE and then selects the desired B&W images containing folder and then user selects the output folder where user would like to store colour images.

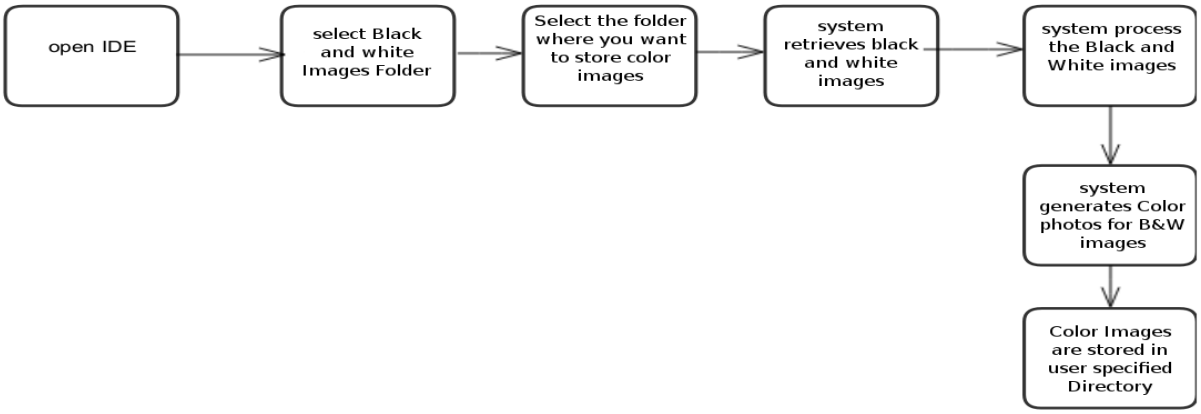


Figure: 5.1 Architecture of Image Style Transfer

Figure: 5.2. Sample image transfer from black and white to color

VI. CONCLUSION

Since the future is building mainly on Machine Learning applications, we would like to develop an application in that field and even the developed application should be helpful for users and it should benefit the society by decreasing their effort by automation. So, this paper we discussed the Automatic Image Colorization through which one can colorize the Black and White Images. Since it takes many hours to colorize a Black and White image due to it's many pixels. Usually, we have to colorize manually each and every pixel of the Black and White image to make it colorize. So, this process has automated it using Neural Networks, especially Convolutional Neural Networks, where we used a pre-trained model for training the dataset.

REFERENCES

[1]. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. In: SIGGRAPH '04, Los Angeles, California, ACM Press (2004) 689–694

[2]. Yatziv, L., Sapiro, G.: Fast image and video colorization using chrominance blending. IEEE Transactions on Image Processing 15(5) (2006) 1120–1129

[3]. Horiuchi, T.: Colorization algorithm using probabilistic relaxation. Image Vision Computing 22(3) (2004) 197–202

[4]. Takahama, T., Horiuchi, T., Kotera, H.: Improvement on colorization accuracy by partitioning algorithm in cielab color space. In Aizawa, K., Nakamura, Y., Satoh, S., eds.: PCM (2). Volume 3332 of Lecture Notes in Computer Science., Springer (2004) 794–801

[5]. Welsh, T., Ashikhmin, M., Mueller, K.: Transferring color to greyscale images. In: SIGGRAPH '02: Proc. of the 29th annual conf. on Computer graphics and interactive techniques, ACM Press (2002) 277–280

[6]. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In Fiume, E., ed.: SIGGRAPH 2001, Computer Graphics Proceedings, ACM Press (2001) 327–340

[7]. Irony, R., Cohen-Or, D., Lischinski, D.: Colorization by Example. In: Proceedings of Eurographics Symposium on Rendering 2005 (EGSR'05, June 29–July 1, 2005, Konstanz, Germany). (2005) 201–210

[8]. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: 9th European Conference on Computer Vision, Graz Austria (May 2006)

[9]. Automatic Image Colorization via Multimodal Predictions Guillaume Charpiat , Matthias Hofmann, and Bernhard Scholkopf