Image Colorization

Prateek Shroff
Texas A & M University
prateek.shroff@tamu.edu

Stuti Sakhi
Texas A & M University
stuti@tamu.edu

ABSTRACT

Our project aims at generating a plausible color version of an image, given the gray scale image as an input. We have implemented the method in [3] with some changes to suit our requirements and constraints. Most of the previous approaches either required human intervention or generated unsaturated images. The method we have implemented in our project is fully automatic and generates vibrant and saturated images unlike the previous approaches.

1 INTRODUCTION

We have implemented the image colorization method as in [3]. Section 2 discusses the algorithm we have implemented in detail. In section 3, we discuss the implementation details including the dataset and pre-processing. We finally give our results and compare it with the results in [3] in section 4.

2 ALGORITHM

The image colorization problem is treated as a multi-class classification problem here. The main reason being that a given object or image can be colorized in multiple ways and still look realistic. So instead of treating it as a regression task, we treat it as a classification task. We have used the algorithm proposed by [3]. However, we have made two changes in our final implementation. The changes are mentioned in the respective sections. The algorithm can be divided into two main components. In the first component, we estimate the probability distribution over quantized color values from the gray scale input using a Convolutional Neural Network. The loss, architecture and other details regarding the CNN are discussed in section 2.1. From the probability distribution, we infer the point estimates which is discussed in section 2.2.

2.1 Probability Distribution Estimation

This component takes a gray scale image as input and outputs a probability distribution for each pixel over the quantized color space. In section 2.1.1, we discuss regarding the quantized color space. Section 2.1.2 gives the details of estimating the ground truth probability distribution. Finally, we discuss the loss and network architecture in section 2.1.3 and section 2.1.4 respectively.

2.1.1 *Gamut.*

The colorization task is treated as a multi-class classification task. We chose the colors which are in gamut as the classes for our task. We have taken the same set of Q = 313 colors as taken in [3]. So, our classification task had 313 classes with each one corresponding to a color in gamut.

2.1.2 *Ground Truth Probability Distribution.*

In order to train the CNN, we need to estimate the ground truth probability distribution from the ground truth ab channel values. We simply converted each ab value in the image to a one hot vector

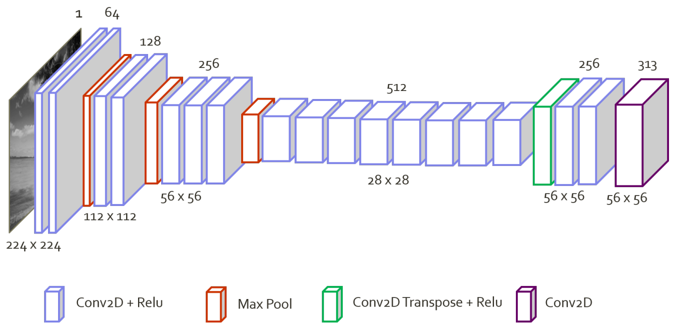

Figure 1: Network Architecture

by searching the nearest color in the gamut. This is a simplification of the soft encoding scheme adapted by [3]. From here we denote the ground truth probability estimate as Z.

2.1.3 *Loss.*

For a given input X, we have Z, the ground truth probability distribution of size [h x w x Q] with values either 0 or 1. $\widehat{Z}$ is the predicted probability distribution of size [h x w x Q] with values ranging from 0 to 1. We define the loss as shown in equation (1)

$$L(\widehat{Z}, Z) = \sum_{h,w} w(Z_{h,w}) \sum_{q} Z_{h,w,q} log(\widehat{Z}_{h,w,q}) \tag{1}$$

The loss is a simple cross entropy loss. However, a class re-balancing term $w(Z_{h,w})$ is added to the loss in order to increase the representation of bright and vibrant colors which are rarely found in the images. These weights were assigned in such a way that the pixels having a rare colour is given higher weight. The weights were pre-evaluated evaluated for each color in the gamut by taking the reciprocal of a mixed distribution as mentioned in [3].

2.1.4 *Network Architecture.*

We worked with two network architectures in our project. Firstly, we replicated the network mentioned in [3]. However, this network took very long to train as we were training from scratch. So we worked on another architecture where we could start our training with pre trained weights. The network mentioned in [3] had a huge similarity with VGG-19 [2]. So we decided to tweak the architecture of VGG 19 according to our need and start our training with pre-trained network. Figure 1 shows the network architecture we used.

2.2 Point Estimation

To arrive at the point estimate $\widehat{Y}$ in ab space, from the $\widehat{Z}$ we take the annealed mean of the distribution as in [3]. The equation (2) and (3) shows the point estimation.

$$f_t(z) = \frac{exp(log(z)/T)}{\sum_q exp(log(z_q))/T)} \tag{2}$$

$$\widehat{Y}_{h,w} = E[f_t(\widehat{Z}_{h,w})] \tag{3}$$

We can see that setting $T = 1$ does not make any change in the distribution and we end up taking the mean of the distribution. As we decrease T to 0, the distribution turns into a single spike at the mode of the distribution. This way we end up taking the mode of the distribution. However, taking the mean results in grayish images and taking the mode results in spatially inconsistent images. We set the $T = 0.38$ as mentioned in the [3].

3 IMPLEMENTATION DETAILS

In this section, we discuss the details of our implementation.

3.1 Dataset

We use publicly available ImageNet [[1]] dataset. It is a large visual database which contains over 1.3 million hand-annotated images ranging over several categories like balloon, dog, planes etc. We use all the images for the training our model.

3.2 Preprocessing

All the images were resized to 224 x 224. Also, the images were converted to Lab colorspace from rgb.

3.3 Train vs Inference Implementation

In case of training, the L channel and ab channel were separated. The ab channel was used to generate the ground truth probability distribution. We trained the network for 140K iterations with a batch size of 32. In case of inference, just the L channel was given as input to the network. The output from the network underwent point estimation to give us the ab channel for the image. The predicted ab channel was combined with the input L channel to get the final colorized image as the output.

4 RESULTS & COMPARISONS

Our network does learn to colorize the image comparable to the paper. But, we do see some artifacts/incorrect hallucinated colors in the final results. Our results were not as colorful as we were hoping.

4.1 Train and Validation Loss

Figure [2] shows the validation error on 10 images taken every 2000 iterations. And Figure [3] shows the training error over the whole training period of 160K iterations. From both of the graphs, there is a clear trend that if we could have trained for more number of iterations we could definitely achieve better results.

4.2 Comparison With Referenced Work

We have compared outputs from Zhang[[3]] paper and from our network. Fig [4] shows the comparison, we can see that our output is quite comparable if not better. Also, referring to Fig [5] we see that our model actually performs better.It colorize the image which looks better and closer to the ground truth. Below we discuss the exact differences between Zhang [3] paper and our implementation.

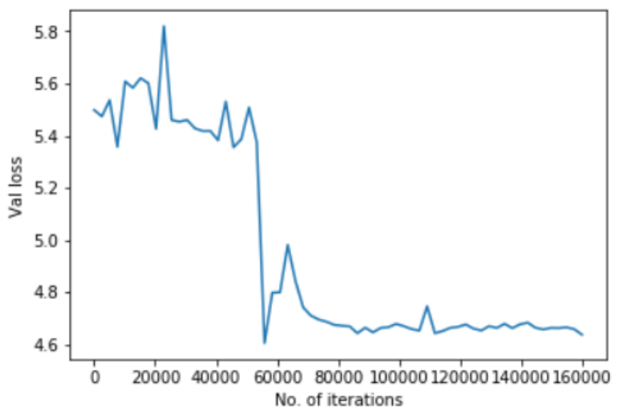

Figure 2: Figure shows the validation error on 10 images taken every 2000 iterations.

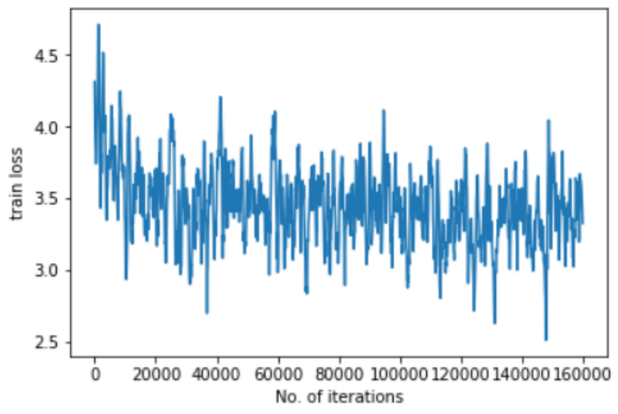

Figure 3: Figure shows the training error taken every 500 iterations.

4.2.1 *Architecture.*

Our final network architecture is different form the one implemented in [3]. We went with this architecture as we could start our training with pre-trained VGG 19 [2] weights which made our training faster. Figure [6] shows the difference in colorization between the outputs of these two architecture at the same iteration. We can see the network which was initialized with pre-trained weights has learned faster.

4.2.2 *Ground Truth Probability Estimation.*

We have used simple one hot encoding to estimate the ground truth probability distribution over the quantized color space. The paper [3] however, uses a softencoding scheme to do the same. Our performance could have improved by implementing the softencoding scheme.

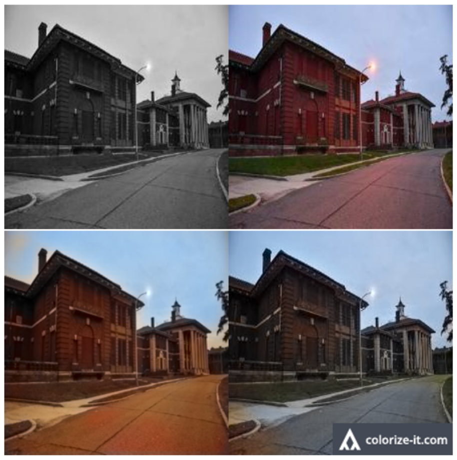

Figure 4: (*top-left*)Input image (*top-right*) Ground truth image *bottom-left*) Our output image (*bottom-right*) Zhang [3]

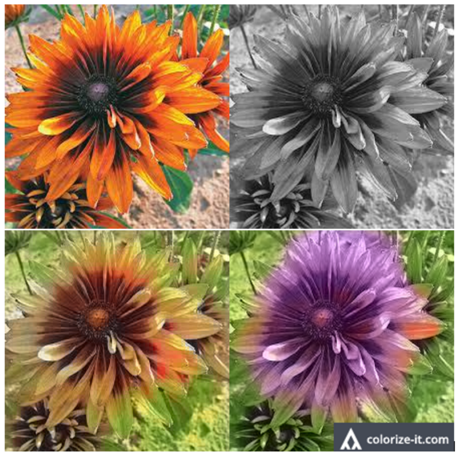

Figure 5: (*top-left*) Ground truth image (*top-right*) Input image *bottom-left*) Our output image (*bottom-right*) Zhang [3]

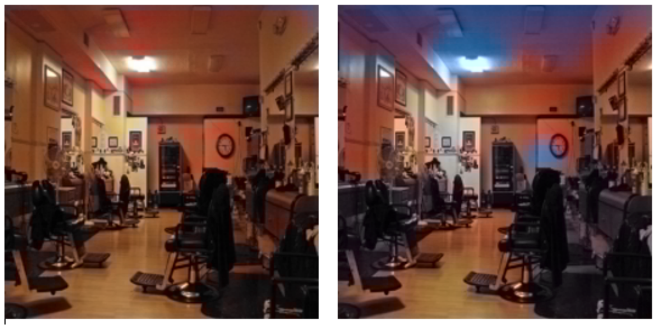

Figure 6: (*left*) Output of pre-trained VGG19. (*right*) Output of Zhang [3]. Both of them are taken after 40K iterations.

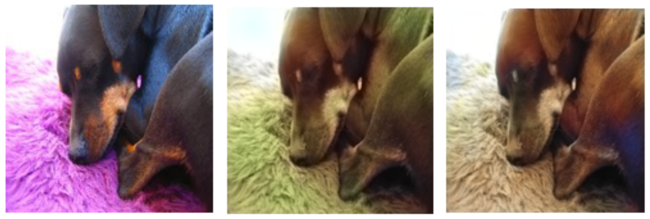

Figure 7: (*left*) Ground truth image. (*middle*) Output of our network after 80K iterations. *right*) Output of our network after 120K iterations.

4.2.3 ***Number of Iterations.*** The official implementation of paper make 500K iterations with batch size of 40 images while due to resource(/time) constraints we could only train our network for 150k iterations in mini batch of 32 images. Training for more number of interations would have resulted in better results for us.

4.3 Observations

Also, we observe that network has learned colors that are ubiquitous in the images like blue, green. But, it is unable to hallucinate complex and very vibrant color like purple. Fig [7], shows that the purple color is not learned yet. but on close inspection we can see that the network learns the gray/black color on running for more iterations. This can be seen from the transition of image at 80K to the same image at the end of 120K iterations.

5 CONCLUSION

Through our experiments, we have demonstrated the efficacy and potential of using deep convolutional neural networks to colorize black and white images. Moving forward, we have identified several avenues for improving our current system. This includes change in architectural design, implementation and training.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009

[2] Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[3] ZHANG, R., ISOLA, P., AND EFROS, A. A. 2016. Colorful image colorization. In Proc. ECCV, 649âĂŞ666.

[4] github repo: https://github.com/richzhang/colorization

[5] github repo:https://github.com/chuchienshu/Colorization