Automatic colorisation of photos (grayscale) using Deep Learning Techniques

Raghav Jindal
1501CS36
Email: raghav.cs15@iitp.ac.in

"CS399: Seminar Report"
Spring 2018

Department of Computer Sc. and Eng.
Indian Institute of Technology Patna

Abstract

Given a gray scale photograph as input, this paper attacks the problem of hallucinating a plausible color version of the photograph. Here I discuss 2 papers which developed models for colorization of black and white photos: 1) First using a model involving concepts of CNN and hyper-columns and 2) Using conditional adversarial networks. Both the techniques are fully automatic approach that produces vibrant and realistic colorization. For evaluation a “colorization Turing test,” was used in the papers.

Keywords: Colorization, Neural networks, Convolutional Neural Networks(CNN), Conditional Adversarial Networks

0.1 Introduction

Automatic colorization is an area of research that possesses great potentials in applications: from black and white photos reconstruction, augmentation of grey scale drawings, to re-colorization of images. Traditionally, people used photo-shop achieve these goals. However, it involved a lot of manual labour and time. This problem can be automated easily using some novel Deep learning techniques, as the perfect training data is easy to get: any colored image can be desaturated and be used as an example.

In this report, two models are discussed: 1) Using a ConvNet + Hypercolumns architecture and 2) using a Conditional-GAN. Both the models take gray scale images and produce (RGB) images. It was found that the GAN architecture can generate better results. The first model was developed by Ryan Dahl [1]. It was one of the first works that was done in this topic.

The second model was developed by Phillip Isola and Jun-Yan Zhu et. al in their trademark paper "Image-to-Image Translation with Conditional Adversarial Networks"[2]. We define image-to-image translation as the problem of translating one possible representation of a scene into another. Traditionally, every different image translation problem has been tackled with separate, special-purpose machinery[[3], [4], [5]]. The paper by [2] had a salient feature: It showed that the same Conditional GAN architecture could be used for any image translation problem.

0.2 Method

Method 1: Using ConvNet and HyperColumns

In the model proposed by [1], the input image is supposed to be of 224 x 224 gray scale image. 3 copies of the image are concatenated together to get a 224 x 224 x 3 sized "matrix". These matrix is passed through first few layers of a pre trained VGG-16 model. At each step, the activation map is bilinearly upscaled to make its size equal to that of the input image (Fig 0.1). Thereafter, the hypercolumns for every input pixel is obtained [Fig 0.1]. **Hypercolumns** are defined for every pixel as the vector of all activations above that pixel.

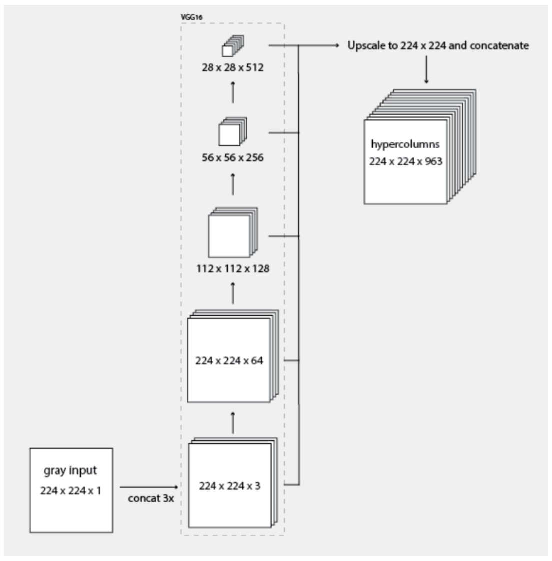

Figure 0.1: Obtaining the hypercolumns for every pixel of the input gray scale image

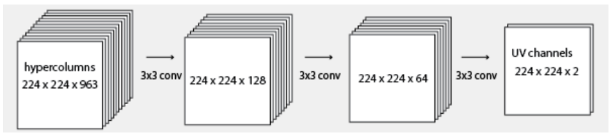

Figure 0.2: Getting UV channels from the hypercolumns

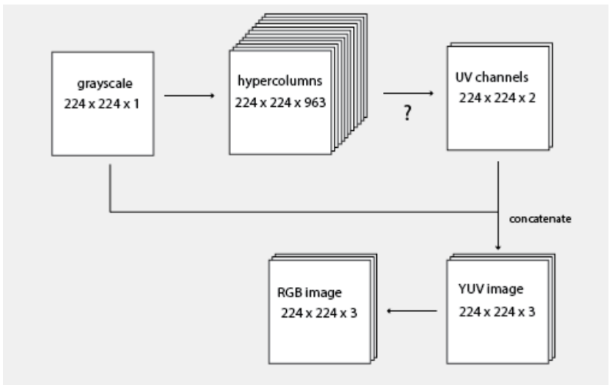

Figure 0.3: Getting final RGB image

Once the Hypercolums are obtained, a triplet of 3 x 3 convolutions are applied with varying number of filters (see Fig 0.2) to obtain the UV channels. YUV format of the image was used instead of RGB, because the "Y" in YUV stands for intensity and the intensity in both the input and output are same. So, we only need to find 2 values i.e U and V instead of 3 values in case of RGB.

Once we get the UV channels, we concatenate the Y channel from the input image to get the YUV image (fig 0.3). Next, a simple matrix operation is applied to convert the YUV to RGB.

Objective function: The loss for i^{th} training example is defined as:

$$L_i = \frac{1}{2}\sum_{p=1}^{n} |F(x_i;\theta)^p - y^p|^2$$

So the overall objective function is represented as:

$$L_{ConvNet} = \sum_{i=1}^{N} L_i$$

Method 2: Using Conditional GAN

GANs are generative models that learn a mapping from random noise vector *z* to an output image *y*,

$$G : z \rightarrow y$$

However, this paper uses a modification called **Conditional GAN**. Conditional GANs learn a mapping from observed image *x* and random noise vector *z,* to *y*

$$G : x, z \rightarrow y$$

The Generator *G* is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator,*D*, which is trained to do as well as possible at detecting the generator's "fakes".
Objective function: The objective funciton consists of 2 terms:

$$L_{L1}(G) = \mathbf{E}_{\mathbf{x},\mathbf{y},\mathbf{z}}[||y - G(x, z)||_1]$$

With this term, the generator is tasked not only to fool the discriminator but also to be near ground truth output in a L1 sense. So the Generator will try to minimise this term. For *D*, this term is constant.
The other term is:

$$L_{cGAN}(G, D) = \mathbf{E}_{\mathbf{y}}[\log D(y)] + \mathbf{E}_{\mathbf{x},\mathbf{z}}[\text{log}(1 - D(G(x, z))]$$

This term is the general objective function of a conditional GAN. The Generator tries to minimize this term while the discriminator tries to maximize the term. More details can be found in [6].

Combining these 2, the objective function of our model becomes:

$$G^* = arg\underset{G}{min}\underset{D}{max}L_{cGAN}(G, D) + \lambda L_{L1}(G)$$

Architecture

The salient feature of the paper was that it showed that the same architecture could be used for any image to image translation problem; which is defined as the problem of translating one possible representation of a scene to another. Fig 0.4 shows the architecture of the paper. Here *G* is the Generator, *D* is the Discriminator, *x* is the image fed into the conditional GAN and *y* is the real image.

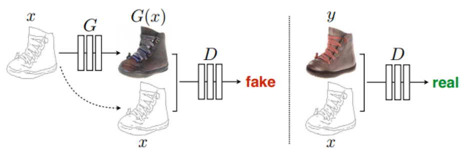

Figure 0.4: CGAN overview used in paper

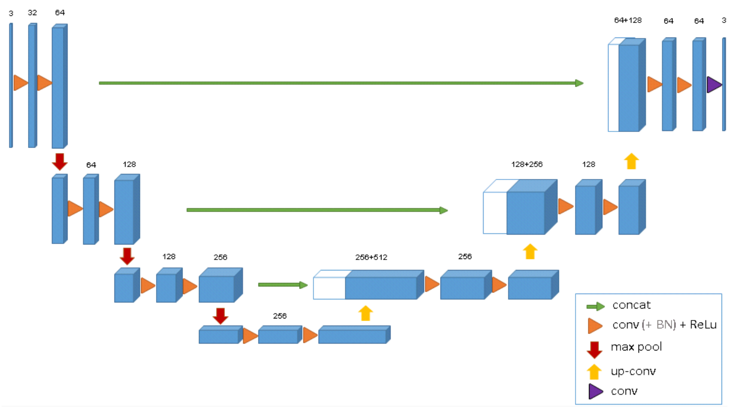

Figure 0.5: Skip connections are used for the GAN

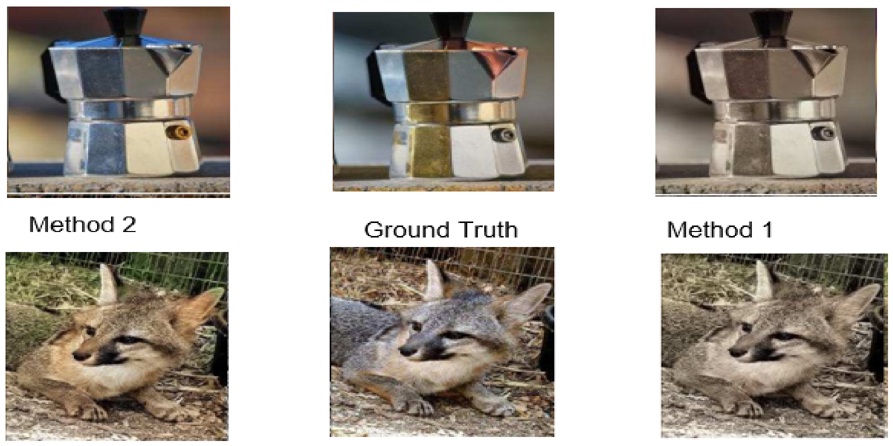

Figure 0.6: Result

Both the Generator and Discriminator use modules of the form convolution-BatchNorm-Relu. For many image translation problems, there is a great deal of low-level information shared between the input and output. eg For the image in Fig 0.4, the edges need to present in both the input and output image. it would be desirable to shuttle this information directly across the network.

So, **Skip Connections** are used in Generator. Fig 0.5 shows skip connections architecture. The output from one layer is directly fed into some forward layer. The green colored lines are skip connections. A U-NET architecture is followed by the GAN i.e the output of layer *i* is fed into layer *n-i*.

0.3 Discussion

Fig 0.6 shows the results of the 2 methods. The leftmost images are the results from method 1, the middle images are the ground truth images, the rightmost images are generated from the second method.

No evaluation metric is available for our results. Plausibility to human observer is the ultimate goal. It is clear that the second method outperforms the first one. The reason for this is that the first method used Mean Squared Error as the loss function, which is a form of L2 loss. It was shown however, by [7] that L2 loss produces blurry results. This can intuitively be understood by considering the case of colorizing a human face. The face of a human being can be black, white or brown. However, when we use L2 loss as objective function, the model will mostly color the face as brown, because brown is the *"average"* color and average value leads to minimization of loss function for L2 Loss.

So to prevent this, we can try to design a different loss function. However, it would involve a lot of time and effort. It would be highly desirable if we could instead specify only a high-level goal, like “make the output indistinguishable from reality”, and then automatically learn a loss function appropriate for satisfying this goal. This is why we used method 2. It uses a Generator and Discriminator based architecture. The main advantage of using a Discriminator is that it designs the loss function for us.

0.4 Conclusion

In this report, we saw 2 methods that try to solve the problem of colorizing black and white photos. The first method used a CNN based architecture. However, it lead to averaging possibilities.

The second method used a GAN based architecture. We found that the second method outperformed the first method as the Discriminator helped to learn the loss function.

One possible extension where the above approach could be used is to colorize old black and white movies.

Bibliography

[1] Ryan Dahl. *http://tinyclouds.org/colorize/*. 2016

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*

[3] A. A. Efros and W. T. Freeman. *Image quilting for texture synthesis and transfer*

[4] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. *Sketch2photo: internet image montage. ACM Transactions on Graphics (TOG)* 28(5):124, 2009

[5] S. Xie and Z. Tu. *Holistically-nested edge detection* In ICCV, 2015

[6] Mehdi Mirza, Simon Osindero *Conditional Generative Adversarial Nets* 2014

[7] G Larsson, M Maire, G Shakhnarovich *Learning representations for automatic colorization* 2016