CSE591 Deep Learning and Visual Computing

Image Colorization with Deep Learning

Chia-Yu Hsu, Jau-Yuan Shiao

Abstract—Image colorization for black and white photos is an interesting more and more popular topic. In this project, we will apply deep learning techniques to do automatic image colorization. Also, we will try to train the neural network with different dataset and slightly modify the model to achieve better accuracy.

Index Terms—Image Colorization, Convolutional Neural Network, Deep Learning

1 Introduction

Video restoration and image enhancement are very popular topic in computer vision for many years. Recently, due to the huge success of CNN, researchers start thinking about auto colorizing for black and white image.

Since CNN has the ability to learn the colors, shapes and patterns of images, researchers believe we can do automatic image colorization through this property. The idea is very simple, use CNN to recognize the object, and then give it a color label. Therefore, we can input a black and white image to CNN and then get a colorized image.

Since training a deep CNN will take lot of time, in our project, we will apply pre-trained network likes VGG-16 to save training time. VGG-16 has been used in some researches[1][2] and achieve good accuracy of image colorization. We might also use other pre-trained networks likes VGG-19 or AlexNet to make some comparison or achieve better accuracy.

2 Proposed Approach

2.1 Preprocessing

Since the input of our network VGG16 is of dimension $224 \times 224 \times 3$, we preprocess each image to the dimension and generate a grayscale version. We then subtract the mean value across all the images in our training set in order to make it zero-centered, which expects to be the input of VGG16.

2.2 Toolbox and Network

The toolbox we use is Keras. Keras is a popular and easy to use Python library for deep learning that can run on top of Theano or TensorFlow. We will set up our network using this libary. And the network we use is VGG16 [6]. The VGG16 has been pretrained on the ImageNet dataset.

2.3 Activation Function

For the activation function, we will apply the rectified linear function since it has been shown to greatly accelerate training convergence [4].

$$f(x) = \max(0, x)$$

2.4 Batch Normalization

For our networks, we place a batch normalization layer before every non-linearity layer except the last few layers before the output. From [5], Ioffe et al introduced batch normalization as a means of dramatically reducing training convergence time and improving accuracy.

2.5 Regression Model

We will apply two different models. The first one is regression-based model [1]. In this task, regression does seem to be well-suited to due to the continuous nature of color spaces. We use the VGG16 as an encoding process and then we try to upscale the preceding layer output, merges the result with an intermediate output from the VGG16 layers via an elementwise sum, and performs a two-dimensional convolution on the result. The network figure can be seen in [2].

2.6 Classification Model

The second one is classification-based model [2]. In order to perform classification on continuous data, we must discretize the domain. We discretize the color space into equi-width bins and apply to each input image prior to feeding it to the input of the network. Then, instead of directly predicting numeric values for color, the network outputs the most probable bin numbers for the pixels. The network figure can be seen in [2].

2.7 Evaluation Metrics

We will apply the sum of the 2 norms of the difference of the generated image pixels and actual image pixels in the U and V channels as the regression loss function.

$$L_{reg.} = ||U_p - U_a||_2^2 + ||V_p - V_a||_2^2$$

To evaluate the accuracy of classification, we will generate the percent of binned pixel values that match between the generated image and actual image for each channel U and V. These values can be viewed as the

closeness between actual image and generated image.

$$\text{Acc.}_U = \frac{1}{N^2} \sum_{(i,j)}^{(N,N)} \mathbf{1}\{\text{bin}(U_p) = \text{bin}(U_a)\}$$

$$\text{Acc.}_V = \frac{1}{N^2} \sum_{(i,j)}^{(N,N)} \mathbf{1}\{\text{bin}(V_p) = \text{bin}(V_a)\}$$

We will also track the percent deviation in average color saturation between pixels in the generated image and in the actual image:

$$\text{Sat. diff.} = \frac{\left|\sum_{(i,j)}^{(N,N)} S_{p_{ij}} - \sum_{(i,j)}^{(N,N)} S_{a_{ij}}\right|}{\sum_{(i,j)}^{(N,N)} S_{a_{ij}}}$$

3 Dataset

For this image colorization problem, the perfect data set is easy to get: any color image can be transferred to black and white image and used as an example. However, to compare with existing results, we will first use MIT CVCL Urban and Natural Scene Categories dataset. This dataset contains eight categories with several thousand images. We will first use the "Open Country" category to train and validate the entire network with less number of images. After that, we will try the dataset used for the 2015 ImageNet challenge, which spans a vast range of categories, artistic styles and subject matter.

4 Proposed Timeline

Date	Tasks
2/17 - 2/24	Read papers and play Keras
2/24 - 3/3	Set-up the CNN with VGG-16
3/3 - 3/17	Train the network with different dataset
3/17 - 3/31	Use different kinds of pre-trained network **OR** modify the architecture of model.
3/31 - 4/14	Make comparison between different architecture and dataset.
4/14 - 4/28	Finetune and write final report

5 Expected Result

Basically, we expect that we can input an image to our CNN and then output a colorized result. Furthermore, we will try different datasets and pretrained network or slightly modify the model structure to get a better accuracy. We will make a comparison between all of the results and show the best architecture and training dataset.

Fig1 Sample result from[1]

References

[1] R. Dahl., "Automatic Colorization", http://tinyclouds.org/colorize/, 2016.

[2] J. Hwang, and Y. Zhou, "Image Colorization with Deep Convolutional Neural Networks", 2016.

[3] R. Zhang, P. Isola and A. Efros, "Colorful Image Colorization", *arXiv preprint arXiv: 1603.08511, 2016.*

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *In Advances in neural information processing systems, pages 1097–1105, 2012.*

[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *arXiv preprint arXiv:1502.03167, 2015.*

[6] [13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556, 2014.*