Image Colorization with Deep Learning Model

Hao Qin hqin32@wisc.edu
Yansong Mao ymao49@wisc.edu
Yazhi Meng meng52@wisc.edu

Abstract

In this project, we implement two measure method to construct colorization neural network. The basic components are convolutional layers and upsampling layers. We transfer the RGB image file into LAB color space, thus the problem is predicting ab channels based on L. Then we introduce the cross-entropy loss to solve desaturated color problem by quantizing ab color space. Finally, our multiclassification model seems outputs some reasonable colorized images but still lack of time to be more precisely.

1. Introduction

Coloring a grey-scale picture seems an interesting perspective to look back at the history and restore the old pictures. It is difficult because a single gray-scale image may correspond to many plausible colored images. But it's also very important and widely applied in areas like restoration of ancient paintings and so on. With the help of deep learning network, we can automatically assign the plausible color into different parts of photograph.

We choose ResNet-34 as our model, a common architecture of convolutional Neural Network, which is an image classification network with 34 layers and residual connections. Firstly, we applied some convolutional layers to extract features from the images, and then we applied deconvolutional layers to increase the special resolution of the features. As for the loss function, since we are doing regression, we'll use a mean squared error loss function.

Moreover, in our work, we loaded the data by LAB color space. This color space is similar to the RGB color space but from different angle, since it still uses three channels like in the RGB space, but channels represent information differently: lightness intensity is encoded through the L channel, green to red is encoded through the A channel encodes, and blue to yellow is encoded through the b channel. Nonetheless, it contains exactly the same information as RGB, but it will make it easier for us to separate out the lightness channel from the other two, because we can extract the light intensity direct from the L channel[3]. Then the task we have is to learn to forecast channels a and b based on L.

2. Related Work

Current shortcomings Current work on image colorization data set have two main drawback. The first one is that the colorization relied a lot on human interaction and annotation for color assigning. For example, Vivek et al. proposed a colorization method using a semi-automatic approach, which requires the user to put the needed color marker in every region to imply how it should be colored.[5]Another drawback is producing desaturated colorization[7], making the image seems unnatural in the human eye. Under this circumstance, we want to implement a learning algorithm with little handcraft procedure and performs well.

Main types of colorization methods As for the statistical method related to colorization, there are two types of methods being used in common, parametric and non-parametric. The parametric method uses basic statistical analysis, turning the colorization into regression optimization problem such as making prediction in the continuous output space or the classification of quantized color values[8]. Otherwise, the non-parametric methods require some human assigned color reference images or picked up automatically. Then the color will be transformed into the input space followed the Image Analogies framework[1, 4, 6]. GAN is also an efficient way to achieve colorization images[2][11]. We plan to use parametric method, more particularly, with convolutional neural network(CNN) to predict color.

3. Proposed Method

We finally build up two models, the regression model and the classification model. Both are based on the encoder-decoder structure combined with ResNet34 structure.

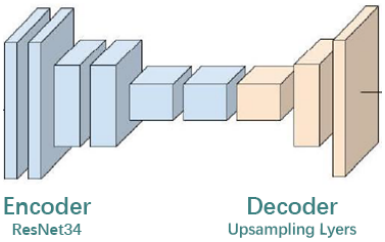

Figure 1. Basic Model Structure

3.1. Regression Model

For the first model, we choose MSE as our loss function. The encoder is the ResNet34 excluded the last several layers considering the limitation of our computation platform. The decoder part is the upsampling layers with Batchnorm layers. There are four upsampling layers since we have squeezing the input image data during encoding and make sure the output sizesize(224*224*3) matching the input size(224*224*3).

The Resnet34 part is a pre-trained model and fine-tuning it to accelerate training procedure. Then we measure the dispersion by MSE and update the weight by ADAM optimizer. By lowering the loss function, this network should learn to match an gray-scale image with a coloring scheme.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2. ResNet Structure

3.2. Multi-classification Model

This model is aimed to solve the desaturated colorization problem. Such phenomena exist among many regression loss colorization model, since the loss function might encourage the model learn a conservative prediction instead of more vivid image.

Layer name	Accuracy
Resnet34 0	conv1
Resnet34 1	BatchNorm2D
Resnet34 2	Relu %
Resnet34 3	conv2_x Maxpool2D
Resnet34 4	conv2_x BasicBlock
Resnet34 5	conv3_x
Resnet34 6	conv4_x
Decoder 0	conv2d(256, 128, k=3, s1, p=1)
Decoder 1	Batchnorm2D+Relu
Decoder 2	Upsampling
Decoder 3	conv2d(128, 64, k=3, s1, p=1)
Decoder 4	Batchnorm2D+Relu
Decoder 5	Upsampling
Decoder 6	conv2d(64, 32, k=3, s1, p=1)
Decoder 7	Batchnorm2D+Relu
Decoder 8	Upsampling
Decoder 6	conv2d(32, 2, k=3, s1, p=1)
Decoder 9	sigmoid
Decoder 10	Upsampling

Table 1. Regression Model Structure.

Layer name	Accuracy
Resnet34 0	conv1
Resnet34 1	BatchNorm2D
Resnet34 2	Relu
Resnet34 3	conv2_x Maxpool2D
Resnet34 4	conv2_x BasicBlock
Resnet34 5	conv3_x
Decoder 0	Upsampling
Decoder 1	conv2d(128, 64, k=3, s1, p=1)
Decoder 2	Batchnorm2D+Relu
Decoder 3	Upsampling
Decoder 3	conv2d(64, 500, k=3, s1, p=1)
Decoder 4	Batchnorm2D+Relu
Decoder 5	Upsampling
fc 0	conv2d(500, 1000, k=3, s1, p=1)
fc 1	Batchnorm2D+Relu
fc 2	conv2d(1000, 625, k=3, s1, p=1)
fc 3	relu
fc 4	softmax

Table 2. Multi-classification Model Structure.

Then we want switch to the multi-classification model based on the assumption that the penalty of misclassification is equal among all the color label forcing the network to learn accurate color more aggressive.

We quantize the ab output space into bins with grid size 10. For a given input X, we learn a mapping $\widehat{\mathbf{Z}} = \mathcal{G}(\mathbf{X})$ to a probability distribution over possible colors $\mathbf{Z} \in$

$[0,1]^{HxWxQ}$ where Q is the number of quantized ab values, H for height and W for weight.

We choose multi-classification cross entropy as the model loss function to measure $\hat{Z}$ to the ground truth Z.

$$\mathrm{L}_{cl}(\widehat{\mathbf{Z}}, \mathbf{Z}) = -\sum \mathbf{Z}_{h,w,q} \log\left(\widehat{\mathbf{Z}}_{h,w,q}\right)$$

To enhance the capacity of the neural network, we set several fully connected layer behind the decoder. Actually, they are 1x1 kernel CNN layers.

4. Experiments

Due to the limitation of our hardware, we have to force our model size by reducing the layer size and removing several fully connected layers, as well as controlling the batch size. Thus might be the main factor to slow down our model training speed. The learning rate we set is 0.0001. For multiclassification model, the batchsize is 3 and the regression model is 128.

4.1. Dataset

Our image data set is from Kaggle website (`https://www.kaggle.com/shravankumar9892/image-colorization`), consisting of 25k 224*224 gray-scale and normal images. The data set consists of two compressed zip files and 4 GB in total:

(1) ab.zip : This contains 25 .npy files consisting of a and b dimensions of LAB color space images, of the MIRFLICKR25k randomly sized colored image data set. The LAB color space generally takes up large disk spaces, hence is a lot slower to load.

(2) l.zip : This consists of a gray-scale.npy file which is the gray-scale version of the MIRFLICKR25k data set.

The MIRFLICKR-25000 is an open project containing 25000 public images imported by open API with manual annotations. And its classification based on the similarity of bag-of-words, which contains 25 thousand images.

We leave ten percents of total image files as test set and ten percents for the valid set.

4.2. Software

Windows 10, CUDA 10.1, Anaconda3

Python 3.8 with several supplement libraries: torch, numpy, skimage, matplotlib, cv2 and PIL

4.3. Hardware

Personal Desktop Platform

NVIDIA GeForce RTX 2070 super (8GB RAM)

5. Results and Discussion

After nearly 5 hours training, we got the colorized results of our deep learning model. Some nice results are shown below and we try to compare features of these images with a good prediction with the others with bad result.

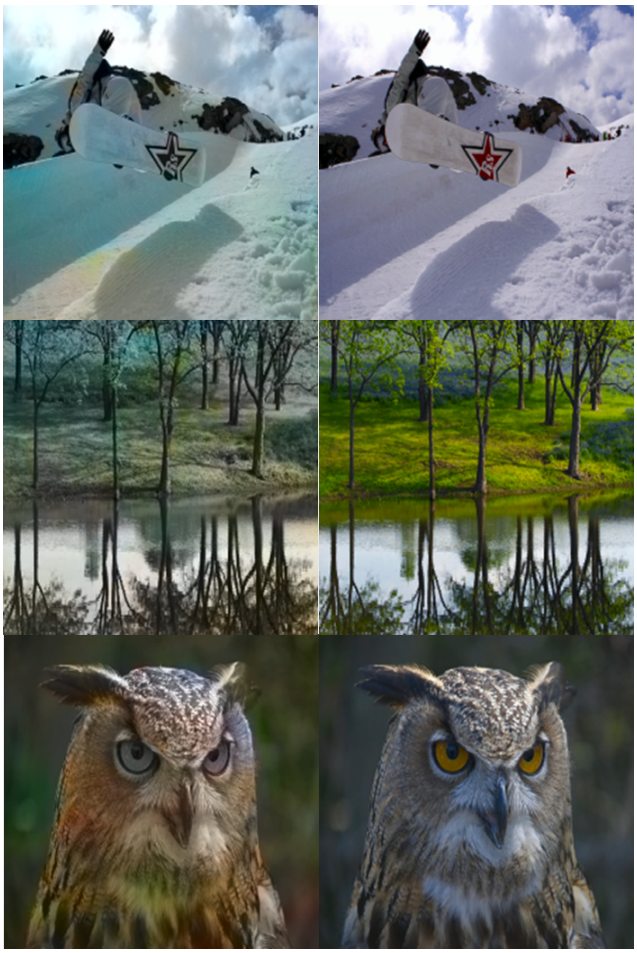

Figure 3. Examples of Regression model output

In order to exclude the overfitting problem, we select some images from valid set to observe the model performance. Without the surprising, the regressing model after training can generate colorized images showing in the Figure 2, with some desaturated problem. But for the second model, since it includes too many parameters, the training speed is pretty slow and the convergence does not come too early like the Figure 3 showing.

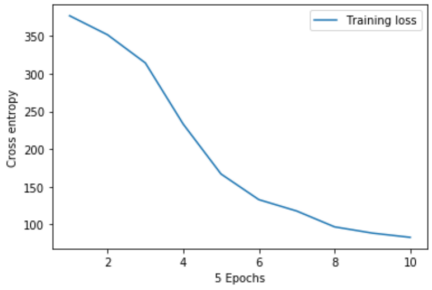

Figure 4. Multiclassification model result

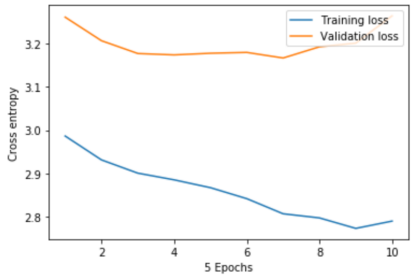

Figure 5. Multiclassification model result

The '5 Epochs' under the X-axis means that the unit along the X-axis is 5.

Even we have made several experiments, the train speed does not perform well. However, we can still have a glimpse of its achievement from current output.

Another important thing is that the valid loss does not descent too much, that means the model need more improvements to generalize its colorization performance.

Figure 6. Examples of multi-classification model

One thing we notice that the multiclassification seems prefer region tuning instead of overall changing which is common in the regression model. It has a sharp boundary showing in some output images, which we have never seen from the regression model.

Maybe we should combine the regression loss and multi-classification penalty. Another problem is that wrong color does not mean that output is totally wrong. Such as in the next figure, especially in the last row, it is hardly to predict the correct color without any custom assignment.

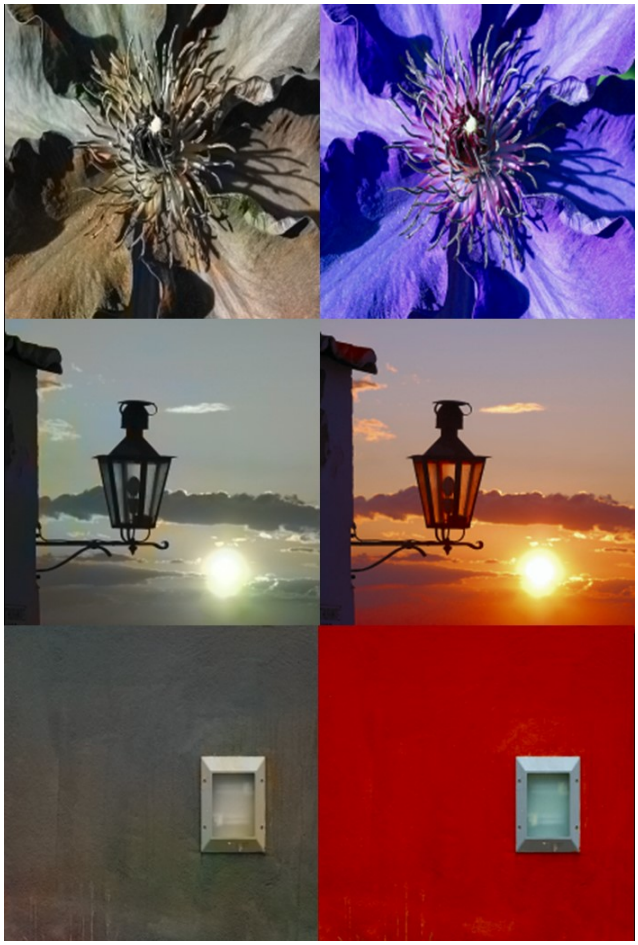

Figure 7. Examples of bad result

As for bad results, they can distinguish easily since it is just look like desaturated and washed-out photos, but when they watched the original, they are surprised and impressed with the original color, which is we discussed before that an outstanding photography are able to give colors that machine failed to predict. For example, the model did not learn flower's petal is purple and it is very rare color. Also, it gave a normal and common scenery of sunset, but the original photography gives us a bright and amazing sunset scenery which really impressed audience. And finally it did not predict red of the wall but the red wall is a really extreme exam-

ple and the wall is usually gray color but photograph gave us a surprised color.

Another method is to use similar images to train the model[4][9]. However, the most interesting thing is that, we cannot directly tell that the colorized is bad without watching the original one. The image itself might perform pretty good. Under this situation, we should say that the self-consistent might be more important than the correctness compared to the original images.

6. Conclusions

As we can see from the above, when the original images have low saturation and no surprised color, the model predicts well while the model does not give ideal result when original image have high saturation like red sunset ,bright yellow light, purple flowers on green grass.

After researching other papers, we took some reasons explaining the deficiency of out result as followings.

"MSE loss is not robust to the inherent ambiguity and multimodal nature of the colorization problem. If an object can take on a set of distinct ab values, the optimal solution to the Euclidean loss will be the mean of the set. In color prediction, this averaging effect favors grayish, desaturated results. Additionally, if the set of plausible colorizations is non-convex, the solution will in fact be out of the set, giving implausible results."[8] From the paper, we know that the predicted color prefer to giving an average color of what the learn from training sets, so the predicted color tends to result washed-out or desaturated picture which looks like old photos.

7. Future work

We conduct Resnet-34 to experiment the model and the reason is that architecture is more simple and time-saving than Resnet-152. We also tried Resnet-152 and VGG-16 but it is beyond memory and break down immediately. But the paper applied modified VGG-16 structure, so we will try to fix the problem or using to cloud computing to avoid breaking down. And compare the VGG-16 and Resnet to see which has a better prediction.

As for desaturated problem, we tend to use multiclassification model to solve this problem since there is a successful example which uses such method combing user guide to recolor grayscale images[10]

Another approach is that we should imporve our loss function. To compare predicted $\mathbf{Z}$ against ground truth, we define function $\mathbf{Z} = \mathcal{H}_{gt}^{-1}(\mathbf{Y})$ which converts ground truth color $\mathbf{Y}$ to vector $\mathbf{Z}$, using a soft-encoding scheme [2] We then use multinomial cross entropy loss $L_{cl}(\cdot, \cdot)$, defined as:

$$\mathrm{L}_{cl}(\widehat{\mathbf{Z}}, \mathbf{Z}) = -\sum v\left(\mathbf{Z}_{h,w}\right) \sum \mathbf{Z}_{h,w,q} \log\left(\widehat{\mathbf{Z}}_{h,w,q}\right)$$

where $v(\cdot)$ is a weighting term that can be used to rebalance the loss based on color-class rarity. [8]

And because of time limitation we implement a simple evaluation of our result. In the future we can enlarge the number of surveys and more colorized image to test if we can fool the participants.

8. Contributions

Our contributions can be divided into two parts. For the computational part, Yansong Mao was responsible for looking for the datasets,loading the data, and data preprocesssing; Hao Qin was responsible for the experimental part, architecture design and loss function design. Yazhi Meng was responsible to find some related papers and developed the evaluation approach. All of us were responsible for building the deep learning model.

As for the writing tasks, Yazhi Meng mainly wrote the introduction, related work and contributions; Yansong Mao mainly wrote the result and conclusions; Hao Qin wrote the experiment and architecture design part. All of our work is distributed evenly.

References

[1] A. Y.-S. Chia, S. Zhuo, R. K. Gupta, Y.-W. Tai, S.-Y. Cho, P. Tan, and S. Lin. Semantic colorization with internet images. *ACM Transactions on Graphics (TOG)*, 30(6):1–8, 2011.

[2] K. Frans. Outline colorization through tandem adversarial networks. *arXiv preprint arXiv:1704.08834*, 2017.

[3] D. Futschik. Colorization of black-and-white images using deep neural networks. 2018.

[4] R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong. Image colorization using similar images. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 369–378, 2012.

[5] V. G. Jacob and S. Gupta. Colorization of grayscale images and videos using a semiautomatic approach. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1653–1656, 2009.

[6] H. B. Kekre and S. D. Thepade. Color traits transfer to grayscale images. In *2008 First International Conference*

on Emerging Trends in Engineering and Technology, pages 82–85. IEEE, 2008.

[7] T. I. Murphy. Coloring black & white images using deep learning. https://mc.ai/coloring-black-white-images-using-deep-learning/. Accessed Jan 2, 2020.

[8] Z. Richard, I. Phillip, and A. Efros Alexei. Colorful image colorization. ECCV, 2016.

[9] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2017.

[10] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros. Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999*, 2017.

[11] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.