Transferring Colorization with Smaller Samples

Yangze Zhao and Fucheng You

Beijing Institute of Graphic Communication. No. 1, Xinghua Street (two section), Daxing District, Beijing, China
Email: 1263108136@qq.com

Abstract. This paper applies transferring learning to colorization and the main body is based on DNN. However, considering the less complexity of the data set and the equipment constraints, we extract global features from a vgg pre-trained model instead of Inception-resnet-v2. And we initialize the other parts weights by fix vgg weights. The result shows that we use less training time and sample and get comparable performance.

1. Introduction

At the beginning, colorization refers to the process that transfer a monochrome into a full-color image. Nowadays, it can also mean old black-and-white video coloring, which has greatly expanded its usage. Coloring hand painting manually is a time-consuming and laborious process because we have to carefully choose colors of different positions in an image. And it has been the preserve of human artists to make image real. Advances in technology have made it possible for machines to do the same. There are about three coloring techniques: scribble-based colorization[10,13], example-based colorization[1,2] and neural network colorization[6,7,8]. The first one requires to scribble on the target grayscale images. Although it has a good performance on small samples, coloring a big number of images may be time-consuming, particularly for amateur. In terms of this issue,[3] put up with an example-based technology and then was adjusted by [1, 11].

The second technology transfers the colorful features from same category to the target monochrome image. Different with scratch means, the picture-based means transfer the colorful features from samples to the source black-white pictures. Another faction, the example-based method, is divided into two categories according to the source of sample pictures: But, searching for a similar and suitable picture could be time consuming. simplies the process by means of making use of the picture data on the Web and adding filtering methods and applies filtering methods to select final sample images. Obviously, they own the same discourages. This method demands same Internet object for concrete per-pixel registration. In a word, images that match each other have to be limited with a rigid shape.

An automatic means is come up to deal with the issue. Initially, as there are not enough details in one image,[4] requests similar reference images. But, the matching noise heavily affects the final performance once a dataset that contains a lot of pictures is adopted in experiment. Then DNN(deep neural network) was applied to solve this problem. The model has powerful modeling capabilities that sometimes be better than human in several tasks and learning methods have been proved by experiments[7]. It has been applied in image classification[5,12], pedestrian detection, image super-resolution, photo adjustment etc. This encourages us to explore its more applications in this field. Our work treats transferring color as a regression problem and DNN architectures are applied in this paper. We guess colorization may improve the performance on face recognition. Restricted to hardware, a small amount of face pictures was applied in our experiment. And because of the low-level amount of
pictures and just one category, we replaced the Resnet with VGG16. To optimize the network, experiment make use of advantages of batch-normalization and Dropout. Finally, our neural network can complete colorization task. Although the training wastes a lot of time and computation resources because of the adoption of a large dataset, the trained architecture can be instantly applied to colorize a source black-white picture.

2. Model Architecture

Figure 1 present the main architectures of our work. Just like other deep learning approaches [1,2,5], after pre-processing, the images are fed in neural network. There are two major procedures in the suggested means: first, training a model using the large training image dataset and then making use of the learned model to output a final colorful picture. The main approach changes the color channel from RGB to Lab. Because Lab space only has two color layers, which means that we can use the original grayscale image in the prediction, and only need to predict two channels at the same time. In addition, about 94% percent of the cells in the human eye detect brightness, with only 6 percent sensing color. That is another reason that we need to keep grayscale layer. In order to convert one to two, we need to use the convolution filter. Just thing of them as blue/red polarizations in 3D glasses. Next several paragraphs in this part, A will introduce the preprocessing, B and C will talk about the Encoder and Feature Extractor. At last, D and E will describe the Fusion and Decoder respectively.

![Figure 1. Architecture](image)

2.1. Preprocessing

We first delete some broken images and mismatching images that which owns some similar features, like size and full black or white pixels with code writing by ourselves. To optimize the data so that the learning convergence can be faster, the pixel values training data are scaled to \([0,1]\) by subtracting the minimum value and then divide the interval length. The formula is as follows:

\[
X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}
\]

(1)

\(X_{\text{norm}}\) denotes the final data. \(X\) is the raw data. \(X_{\text{max}}\) and \(X_{\text{min}}\) refer to the maximum value and the minimum value.

2.2. Encoder

The Encoder process shape \((H, W)\) black-white pictures and give a resized vector representation. And we uses convolutional operations with shape \((3, 3)\). Padding layer is used to limit the image shape. In addition, more concrete information is in Table1. To avoid overfitting in layers which have major nodes, we add Dropout layers to randomly delete some hidden neurons in the network, keep input and output neurons unchanged. As for the probabilities of remaining the nodes, we choose numerical value consistent with the number of layer nodes.

\[
y = \sigma(b + Wx)
\]

(2)
Where \( x \in \mathbb{R}^n \) and \( y \in \mathbb{R}^m \) are the input and output of the layer, respectively, \( W \) is an \( m \times n \) matrix of weights, \( b \in \mathbb{R}^m \) is a bias vector and \( \sigma: \mathbb{R}^m \rightarrow \mathbb{R}^m \) is a non-linear transfer function. Both the weights and the bias are learnt through back-propagation [Rumelhart et al. 1986], which consists of using the chain rule to propagate a loss to update the parameters. The loss consists in the error between the prediction of the network and the training data ground truth.

2.3. Feature Extractor

Like many other papers[16], this paper classifies features into local feature and global features. Local features extract independent image blocks from the image. In general, the first step is to extract some space-time interests and then extract corresponding image blocks. Finally, we combine these image blocks. The advantage of local feature is that it does not depend on the segmentation and localization and tracking of human body at the bottom, and it is not very sensitive to noise and occlusion. However, it needs to extract enough stable interest points related to the action category, so it requires a lot of preprocessing. Global characteristics is interested in the detected the whole of the image, typically by background subtraction figure or tracking methods. But These features are sensitive to noise, partial occlusion, and changes in perspective. We scale the input image into 224*224. To extract mid-level features with a respectively simple method, we use a pre-trained VGG16 model. This results in a 1001*1*1 embedding.

2.4. Fusion

Combination part copy the output vector of VGG16 and after that paste it to the feature vector of the encoder along the depth axis. This method was introduced by [13]. The model take a single vector with the high-level features and the common features with shape (H/8, W/8, 1256). By duplicating the output volume and copy it specific times we promise the high-level features transformed by the feature volume is distributed among every spatial region of the input. In addition, this method can be robust to any image shape, strengthen the architecture representable power. In the end, we make use of 256 convolutional kernels with shape (1, 1), result in a feature vector of dimension.

\[
y_{u,v}^{\text{fusion}} = \sigma(b + W[y_{u,v}^{\text{global}}])
\]  

Where \( y_{u,v}^{\text{fusion}} \in \mathbb{R}^{256} \) is the fused feature at \((u, v)\), \( y_{u,v}^{\text{global}} \in \mathbb{R}^{256} \) is the global feature vector, \( y_{u,v}^{\text{mid}} \in \mathbb{R}^{256} \) is the mid-level feature at \((u, v)\), \( W \) is a 256-by-512 weight matrix, and \( b \in \mathbb{R}^{256} \) is a bias. Here, both \( W \) and \( b \) are learnable part of the network.

Where \( y_{u,v} \in \mathbb{R}^{256} \) is the fused feature at \((u, v)\), \( y_{u,v}^{\text{global}} \in \mathbb{R}^{256} \) is the global feature vector, \( y_{u,v}^{\text{mid}} \in \mathbb{R}^{256} \) is the mid-level feature at \((u, v)\), \( W \) is a 256-by-512 weight matrix, and \( b \in \mathbb{R}^{256} \) is a bias. Here, both \( W \) and \( b \) are learnable part of the network.

3. Decode

As a result, the decoder layer receives the input with shape \((8H, 8W, 256)\) and then the input is extracted features by several convolutional layers. Meanwhile, to resize the shape of the input, model takes few upsampling layers after each convolutional layer. And we applies nearest neighbor approach to add pixel values.

4. Experiment

The experiment can be divided into two parts: training and test. Comparing to original paper[2], we sample the 2/3 dataset trained on the source architecture. We will discuss the details about the experiment below.
From left to right, they correspond to original picture state of the art and ours.

4.1. Loss Function

We adopt MSE (mean-square error) as loss objection. Our goal is to optimal the model parameters until loss function reaches a goal. One reason for which we choose MSE is that we want to quantify the loss between the predicted pixels colors and their true values. The main formula:

$$L(y^{\text{color}}, y^{\text{class}}) = \| y^{\text{color}} - y^{\text{color,*}} \|^2_{\text{FRO}}$$

(4)

the formula, L represent the loss function of the neural network. And the y color and y color * represent the label color and the predicted color respectively. FRO represent the Frobenius norm.

4.2. Batch Normalization

Comparing to common data normalization, batch normalization make the distribution of any neuron in each layer back to standard normal distribution with a mean of zero and variance of one. This method results that the input value will fall into the area where activation function is sensitive to the input, so that small changes in the input result in a large change in the objective function. Further, the gradient becomes larger so that the gradient disappearance problem is avoided. And convergence time of loss function will be shorter.

5. Conclusion

We applies transferring learning to colorization and the main body is based on DNN. We extract global features from vgg pre-trained model instead of Inception-resnet-v2. And we initialize the other parts weights by fix vgg weights. The result shows that we use less training time and sample and get comparable performance.

6. References

[1] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012: 1097-1105.

[2] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[3] O. Bulan, V. Kozitsky, P. Ramesh, and M. Shreve, “Segmentation and annotation-free license plate recognition with deep localization and failure identification,” IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 9, pp. 2351–2363, Sept 2017.

[4] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[5] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[6] C. Gou, K.-F. Wang, Y.-J. Yao et al., "Vehicle license plate recognition based on extremal regions and restricted Boltzmann machines", IEEE Trans. Intell. Transp. Syst., vol. 17, no. 4, pp. 1096-1107, 2016.

[7] G. Charpiat, M. Hofmann, and B. Scholkopf. Auto Automatic image colorization via multimodal predictions. In ECCV, pages 126 – 139. Springer, 2008. 1, 2, 6, 7

[8] 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. In TOG, volume 30, page 156. ACM,2011. 1, 2, 5, 6

[9] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. PAMI,24(5):603 – 619, 2002. 7

[10] Cheng Z, Yang Q, Sheng B. Deep Colorization[J]. 2016.

[11] Zhang R, Isola P, Efros A A. Colorful Image Colorization[C]// European Conference on Computer Vision. Springer, Cham, 2016:649-666.

[12] Yarin Gal, Zoubin Ghahramani. Dropout as a Bayesian approximation: representing model uncertainty in deep learning[J]. 2015:1050-1059.

[13] Everingham M, Van Gool L, Williams C K I, et al. The pascal visual object classes (voc) challenge[J]. International journal of computer vision, 2010, 88(2): 303-338.