CS230

Image Colorization - CS230 - Winter 2018

Alex Avery, Dhruv Amin
Department of Computer Science
Stanford University
alavery@stanford.edu, dhruv92@stanford.edu

Abstract

Image colorization is a challenging topic of ongoing research in Computer Vision. We take a grayscale image as input and attempt to produce a coloring scheme. The goal is to make the output image as realistic as the input, although not necessarily the same as the ground truth version. We first explored a convolutional neural network based model to accomplish this task. We then combined our model with a classfier, the Inception ResNet V2 trained on 1.2 million images, to attempt to achieve more realisitic performance based upon the classification of the object to be colorized.

1 Introduction

Photography may appear to be the snap of a picture, but the best photographs often undergo intense after-effects on the computer. Image colorization is one technique to add style to a photograph or apply a combination of styles. Additionally, image colorization can add color to photographs that were originally taken in black and white. This can be used to provide a best-guess as to the context of the picture, and help bridge the gap between the past and the present. The goal of our model is to produce realistic colorized photos. We start with a simple 256 x 256 pixel grayscale image as an input. We then use a neural network to output a predicted colorized image. We have trained our model to output photos with realistic colors by training it on realistic images. This does not mean that the output photo will match the ground truth every time. Instead, the model should produce a colorized image so realistic that a viewer could not spot the fake when looking at a true color image and an image produced by our model.

Figure 1: Grayscale image on the left. Target colorized image on the right.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

2.1 Tasks By Hand

Until recently, colorization was a pain-staking task done by hand. The digital grayscale image would be touched up pixel by pixel. Some improvements were made in this area by adding color bleeding and color continuity. This allowed for batch work to be done, but this was still done manually. In 2005, [9] described how human scribbles could suggest to the computer which colors to fill in. While this did automate the process to a degree, it still required heavy user input.

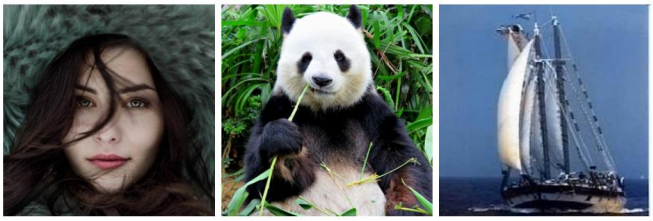
2.2 CNNs

Most of our research involved the use of convolutional neural networks to colorize images [2][3][4][6]. Early models use a simple mean squared error for the loss function. This leads to desaturated photos as it encourages the model to make safe predictions, which can lead to less color or browning. Newer models, such as Colorful Image Colorization [1], tailor their loss function to the colorization problem reweighting to more rare colors given that frequency of colors are not distributed evenly. This helps encourage bolder pixel choices rather than conservative ones.

3 Dataset and Features

3.1 Dataset

We used a publically available series of 10,000 photos from Unsplash [6]. We used 95% in the training set, 2.5% in the dev set, and 2.5% in the test set. These photos are each 256 x 256 pixels, and include various content including faces, animals, and objects. Although the input of our model is a grayscale image, our dataset is all colored photos so that we have a target to optimize towards. To help with generalization, we also performed various transformations including zooms, flips, and shears to prevent overfitting. Here are some examples of the images we used in our training and testing data:



3.2 Colorspace

As part of preprocessing, we converted the RGB layers of each image into Lab colorspace. In Lab, we have an 'L' layer which represents the grayscale image, an 'a' layer that represents the red-green color spectrum, and a 'b' layer that represents the blue-yellow color spectrum. The 'L' layer acts as the input to our model, and the a,b layers become the target. The 'L' layer ranges in value from 0-100, whereas the a,b layers are pixel values that range from 0-255. We normalize the image inputs before training.

4 Methods

We developed two different colorization networks, and will discuss both of those along with our loss function.

4.1 Loss Function

We used a mean squared error loss function. The benefit of using this loss function is its simplicity. We can clearly measure the distance between the target pixel value and our predicted pixel value for

$$L_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2}\sum_{h,w}\|\mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}\|_2^2$$

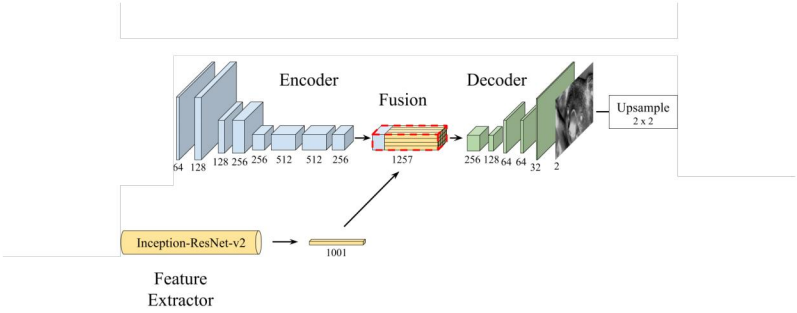
each of the color layers. For most images, getting back to the ground truth would be the best outcome, so this loss function helps optimize for that. The drawback of this loss function is that it encourages conservative predictions. For example, it heavily penalizes picking red instead green, when either choice might be realistic, say for an apple. This is a reason that many of our output images came out with a brown tint. Brown ends up having the lowest loss value, so if the model isn't sure which direction to move, it may conservatively choose the mean. We considered alternative loss functions, but ultimately attempted to correct for this bias by itegrating a classifier into our model

4.2 Our Approach

We ended up building two different colorization models. The first is a Convolutional Neural Network. For this, we chose a learning rate of 0.001, used the rmsprop optimizer, and a mean squared error loss function. We settled on a batch size of 20.

Conv - F:64, K:3, S:1, P:1, A: relu
Conv - F:64, K:3, S:2, P:2, A: relu
Conv - F:128, K:3, S:1, P:1, A: relu
Conv - F:128, K:3, S:2, P:2, A: relu
Conv - F:256, K:3, S:1, P:1, A: relu
Conv - F:256, K:3, S:2, P:2, A: relu
Conv - F:512, K:3, S:1, P:1, A: relu
Conv - F:256, K:3, S:1, P:1, A: relu
Conv - F:128, K:3, S:1, P:1, A: relu
Conv - F:64, K:3, S:1, P:1, A: relu
Conv - F:32, K:3, S:1, P:1, A: relu
Conv - F:2, K:3, S:1, P:1, A: tanh

The second is made up of four components: an encoder, a parallel classifier, a fusion layer, and a decoder. The encoder works similarly to the first half of the first model, while the classifier runs a forward pass on the grayscale image in parallel and gets a 1000 x 1 prediction. This vector is concatenated with the output from the encoder in the fusion layer, which is then run through the decoder. Our base architecture was inspired by the the Deep Koliarization network [7] and we then experimented with deeper convolutional layers to extend the work of the paper. For this model, we had a learning rate of 0.001, used an adam optimizer, and a mean squared error loss function. We ultimately settled on a batch size of 50 images by monitoring the shape of our training loss.



The model transfer learns from the Inception ResNet V2 classifier to create an embedding for each input image. We merge this embedding with the output of the econder before feeding both as inputs to the decoder. The Inception ResNet v2 [10] has been trained on 1.2 million images (ImageNet) to achieve state of the art accuracy in classifying images. By transferring the learning from the classifier to the coloring network, the network can get a sense of what's in the picture. This enables the network to match an object representation with a coloring scheme.

5 Experiments/Results/Discussion

Our results were not as colorful we were hoping. Despite numerous experiments in model architecture we were unable to replicate the results achieved by previous papers. The poor results of the first simple convolutional neural network can be explained by a lack of a sophisticated enough model to represent the task. We found that the model produced test images that were brown in color, a frequently cited outcome for simple models that lack the ability to generalize effectively.

The poor results of the fusion network were more surprising. We found that our training loss decreased as expected and our validation accuracy improved over subsequent epochs. Nevertheless, our final test images also came out brown in color rather than a vibrant set of realistic colorizations.

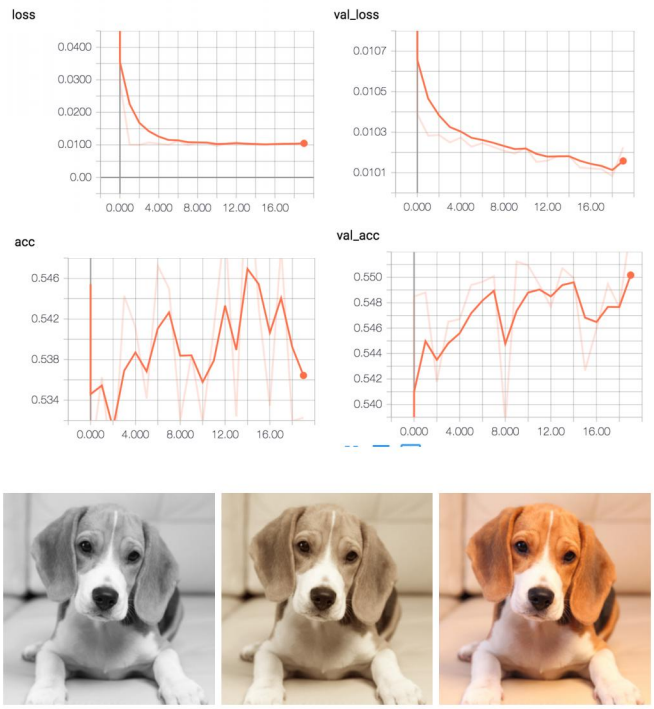


Figure 2: Grayscale image. Colorized test image output from fusion model. Ground truth.

After further research, we deduced that the result was due to a poor choice in loss function. Brown results in the lowest mean squared error. Our next set of experiments would be to experiment with alternative formulations of the loss in order to increase color saturation. We also believe our model could benefit from increased training time and a larget set of images. We ran our experiments over 20 epochs and with 10,000 training images each run took 12 hours to complete running on an Azure GPU. Given the positive directions of the training and validation losses, we believe more training time over a larger training set would result in more realistic image colorizations. Finally, we found the task especially challenging due to a lack of an intermediate quantitative accuracy metric to determine the direction of our training. We did a simple survey at the end of training runs to determine the frequency of accurate colorings, but we found this qualitative approach to be too slow and blunt to iterate in the right direction.

6 Conclusion/Future Work

6.1 Conclusion

Overall, we gained a newfound appreciation for the challenge of producing realistic colorizations and enjoyed experimenting with the approaches of previous papers. The Lab colorspace seems like the best way to work with grayscale and color images. The series of convolutional neural networks is a good simple model to colorize photos, and adding a parallel classifier to learn more about these

grayscale photos to help in the colorization process makes intuitive sense. Still, we find that we need need to iterate further on these approaches to realize better colorings than the desaturated nature of our test results.

6.2 Future Work

If we had more time, there are many ways in which we could improve our model. Per usual, more time and compute would let us train on more images and work with larger images. This would most likely lead to more interesting results.

Beyond that, the first would be to explore different types of loss functions. These loss functions could better reflect the goal of our model, which was to produce realistic images. New approaches would improve on the currently desaturated images by penalizing the model less for picking the incorrect color.

We would also look into color rebalancing. Color prediction is inherently multimodal in that many objects can take on several plausible colorizations. For example, an croquet ball is typically red, green, blue, or yellow, but unlikely to be purple or orange. To appropriately model the multimodal nature of the problem, we could predict a distribution of possible colors for each pixel, and then re-weight at training time to emphasize rare colors. This encourages our model to exploit the full diversity of the data on which it is trained. The end result is colorizations that are more realistic than our current approach.

Lastly, if the color prediction becomes stronger, we would like to apply our approach to interesting use cases. One could be colorizing historical black and white photographs. Another would be applying one photographer's style to a set of black and white photographs by training purely on a certain photographer's work. The final would be applying these concepts to videos and accounting for consistent colorization across multiple frames.

7 Contributions

Research - Alex and Dhruv

Preprocessing of images - Dhruv

Azure GPU setup - Dhruv

Base model - Dhruv

Model architecture iteration - Alex and Dhruv

Hyperparameter search - Alex

Milestone/Paper/Poster - Alex and Dhruv

Presentation - Alex and Dhruv

References

[1] Zhang, Richard, Phillip Isola, and Alexei A. Efros. "Colorful image colorization." European Conference on Computer Vision. Springer, Cham, 2016.

[2] Cheng, Zezhou, Qingxiong Yang, and Bin Sheng. "Deep colorization." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[3] Larsson, Gustav, Michael Maire, and Gregory Shakhnarovich. "Learning representations for automatic colorization." European Conference on Computer Vision. Springer, Cham, 2016.

[4] Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification." ACM Transactions on Graphics (TOG) 35.4 (2016): 110.

[5] Zhang, Richard, et al. "Real-time user-guided image colorization with learned deep priors." arXiv preprint arXiv:1705.02999 (2017).

[6] Wallner, Emil. https://medium.freecodecamp.org/colorize-b-w-photos-with-a-100-line-neural-network-53d9b4449f8d

[7] Baldassarre, Federico, Diego González Morín, and Lucas Rodés-Guirao. "Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2." arXiv preprint arXiv:1712.03400 (2017).

[8] Chollet, F. (2015) keras, GitHub. https://github.com/fchollet/keras

[9] Huang, Yi-Chin, et al. "An adaptive edge detection based colorization algorithm and its applications." Proceedings of the 13th annual ACM international conference on Multimedia. ACM, 2005.

[10]Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." AAAI. Vol. 4. 2017.