Baseline Image Colorization: CS5891 Final Project Report

Sameer Puri

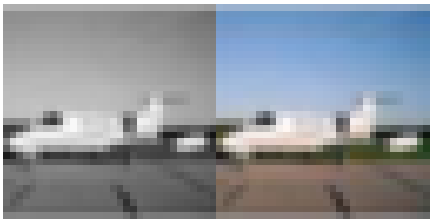
April 29, 2019

Abstract

Image colorization is the task of taking monochromatic imagery and adding appropriate color. Unlike image classification and other problems typically targeted in deep learning, it is an underconstrained problem with a unpredictable truth value, as some objects like vehicles have random color. Past work has made efforts towards producing plausible colorings in face of these uncertainties. Herein, I present a convolutional neural network for image colorization of grayscale images. Instead of producing plausible colorings directly, the network designed in this project aims to produce a baseline colorization of consistently observed features (i.e. grass) which can be built upon. The CIFAR10 dataset is used for learning and evaluation. The test error does not decrease by much, indicative that colorization involves some element of randomness.

Introduction

Image colorization involves adding realistic color to grayscale images. It is often applied in the restoration of monochromatic pictures or video taken prior to the 1970s. The task can be quite time consuming for humans to do by hand and is often assisted with modern graphics editing suites. Several colorization algorithms have recently been developed in an attempt to reduce the effort needed. However, the task is difficult to automate. Consider restoring the color of the image in Figure 1(a). It is clear that the sky should be blue, but the color of the plane is dubious; colors overlap in grayscale, so the plane could be a light shade of red or simply white. This is indicative of how the problem is underconstrained; though the color of some objects in an image may be impossible to predict, the objective is to produce realistic color, not the ground truth color. In this project, I tackle this problem using a convolutional neural network. Unlike classification problems focused on in class, where image features are encoded with successive convolution down to several classes, image colorization requires producing an output that is the same size as the input image.



(a) In grayscale (b) Original

Figure 1: Airplane, photograph from the CIFAR10 dataset sample images.

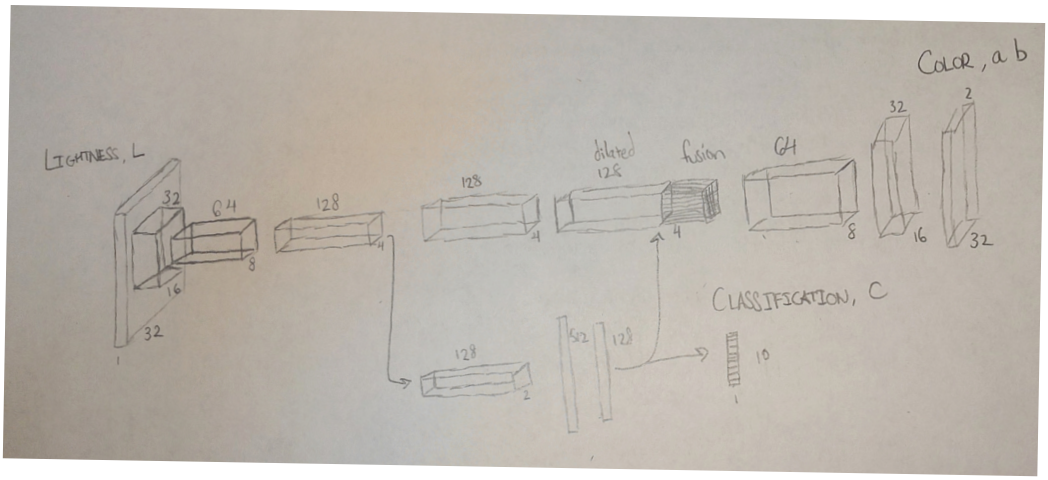


Figure 2: The CNN network architecture developed during the project.

Approach

Model

A generic encoder-decoder convolutional neural network architecture is specialized for the image colorization problem and implemented in PyTorch. As shown in Figure 2, the network consists of four major components. The encoder, which identifies useful image features, consists of three convolutional layers with a stride of 2 followed by a final layer that uses dilation with padding. The classifier uses a sequence of fully connected layers to classify images using the encoder output. To boost the architecture's contextual awareness, the fusion component combines encoder output and intermediate classifier output with fully connected layers. Finally, a decoder colorizes the image with two layers of transposed convolution with a stride of 2. All layers are followed by batch normalization and inplace rectified linear units, except for the classifier output which uses softmax and the decoder output layer which does not have anything. For convolutional layers with a stride of 1, a 3x3 kernel is used. For those with a stride of 2, a 4x4 kernel is used. This is because using a kernel size that is not divisible by the stride can produce checkerboard artifacts in the output colorization[5].

Dataset

The network is trained and tested using the CIFAR10 dataset. It contains 60,000 32x32 images classified into 10 categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. There are 5,000 training images and 1,000 test images per category. Though there are many other available datasets, CIFAR10 is the largest dataset readily available in PyTorch.

Preprocessing

Before learning, images are converted to the CIELAB color space. It represents the color of an image with three channels: L (luminance), a (green to red) and b (blue to yellow)[2]. The luminance channel is input to the model and the a,b channels and image label are predicted by the model. When the a & b channels are 0, the image is just in grayscale. Other color spaces like YUV, YCbCr, and YDbDr were considered and evaluated but performed poorly, frequently produced invalid pixel values during learning. With the HSV color space, the model did not converge at all. Success with CIELAB can likely be attributed to its properties of perceptual uniformity and use in making color balance corrections.

$$z_i = \begin{cases} 0.5(x_i - y_i)^2, & \text{if } |x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, & \text{otherwise} \end{cases}$$

Figure 3: Huber loss function[4]

Data Augmentation

Some data augmentation is done at runtime to improve model generalization. Images are randomly flipped horizontally,

Learning

In examining the CIFAR10 dataset, many images that were already grayscale were found. While I could have filtered out images that had a and b channel values close to 0, I chose to instead use the Huber loss function which is more resistant to outliers than mean-squared error. As shown in Figure 3, it is a smoothing of L1 loss function that applies mean squared error when the absolute value of the difference between the predicted color and the actual color for a pixel falls below 1. The Adam optimizer is used. An L2 penalty was tested but led to underfitting.

Evaluation

Identifying performance criteria

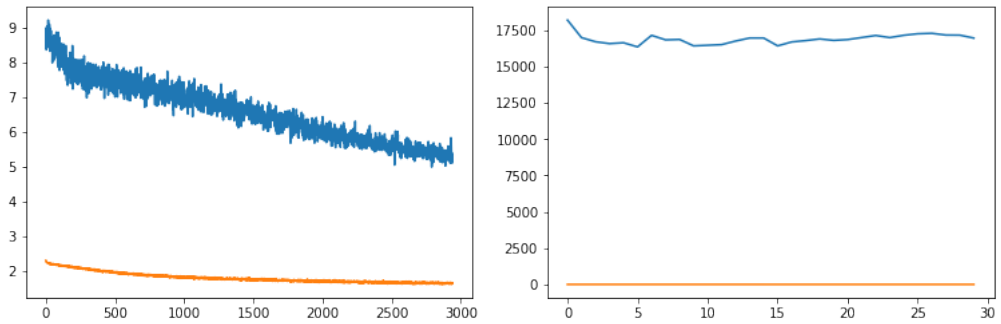
There is no universally best method for evaluating the effectiveness of this model. Using an image's original color as ground truth can be misleading. Consider the automobile class, where determining the color of a vehicle from a grayscale image is effectively a random guess. The model is penalized for guessing incorrectly, so it will tend towards the average color it sees for automobiles during training. It would also be unfair to judge against an image in the test set that is already grayscale or has a color imbalance. For these reasons, I consider the absolute (L1), error on the test set as a reasonable metric for stopping training. When the L1 error reaches a plateau, the model has found the maximum number of consistently colorable features. In other words, this minimizes the generalization gap. Ideally, a group of participants would judge whether images in the test set look real, but or the sake of this project, I judge the images myself.

Training process

The model is trained and periodically evaluated on the CIFAR10 dataset with the Adam optimizer. A batch size of 512 is and a learning rate of 0.001 were identified as appropriate. Smaller batch sizes caused significant perturbation in the test error. Higher learning rates trap the model in a region where images are consistently colored light brown. Figure 4(a) shows the training loss for classification and colorization. Figure 4(b) shows the test error for classification and colorization when evaluated at every epoch. Training was stopped early at 30 epochs, as test error begins to increase. A copy of the Jupyter notebook and model trained in Google Colaboratory is available online at `https://github.com/sameer/colorize`.

Results

As shown in Figure 5, ten random images from each class are drawn from the test set and evaluated with the model. The automobile and truck classes perform the worst with images colored from gray to light reddish brown. This is indicative of how vehicles are found in many different scenes and their color cannot be predicted. The airplane class performs best since the sky can be consistently colored blue and the majority of airplanes are shades of gray. The ship class sees similar performance but there are some dubious cases where boat color is unknown. Animal pictures have quite varied performance. Birds are an odd case that performs poorly; some images are of ground-dwelling birds while others are of birds in trees, which may



(a) Training loss

(b) Test error

Figure 4: Plots of model training progress. Colorization in blue, classification in orange.

be problematic. Cats also colored poorly, as their environments and fur color are varied. Frogs and deer perform well with the ground being consistently colored green and brown.

Related Work

Several convolutional neural network architectures for image colorization have been recently proposed. Zhang et al. design a network targeting the same task but with a different objective: colorizing images to convince viewers that they are realistic. They acknowledge how L2/L1 loss functions in previous work cause colorization networks to output grayish images and propose a loss function that quantizes a & b channel values as classes and rebalances the classes based upon empirical color distribution seen in ImageNet[4]. Adapting this novel loss function to the network in this project would likely lead to much more vibrant colorization. Iizuka et al. develop a similar fusion model composed of several networks, where a low-level feature network is fed into a global feature network used for classification and fused with a mid-level feature network for classification. Their colorization network is akin to the decoder component built in this project, but uses nearest neighbor scaling rather than transposed convolution to increase the size of the input to that of the original image[5]. Though test error decreased by a magnitude of only 7.92%, this is not an indicator of poor model performance, but rather that there is a limited number of features that have consistently coloring, and randomness plays a greater role in colorization than anticipated. Another factor to consider is that the CIFAR10 dataset is quite small compared to datasets other models are trained on. Th network would have many more images to extrapolate from if it were run on a downsampled version of ImageNet, or even the 80 million tiny images dataset.

Conclusion

In this paper, I proposed a network for baseline image colorization. A data fusion component is used to add contextual information to the colorization process. Checkerboard artifacts are avoided by using a kernel size divisible by the size. The task of colorization requires user intervention for now, as selecting an appropriate color is effectively random in some scenarios. Future colorization networks may use image segmentation, multi-level classification, and other techniques to add additional context to the model.

References

[1] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and Checkerboard Artifacts. `https://distill.pub/2016/deconv-checkerboard/`

[2] Wikipedia contributors. CIELAB Color Space. `https://en.wikipedia.org/wiki/CIELAB_color_space`

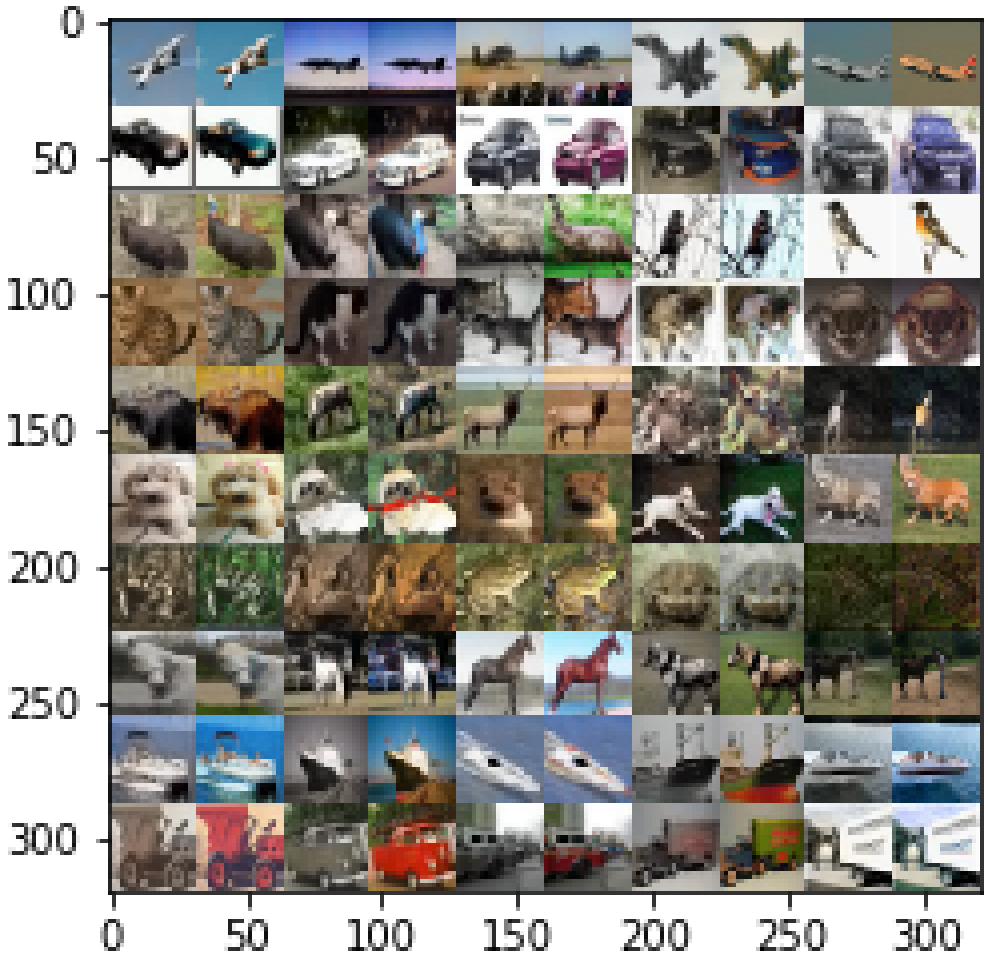
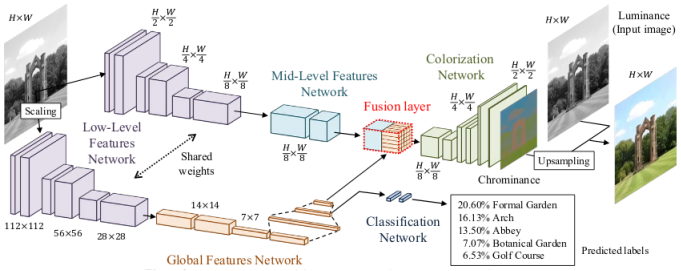
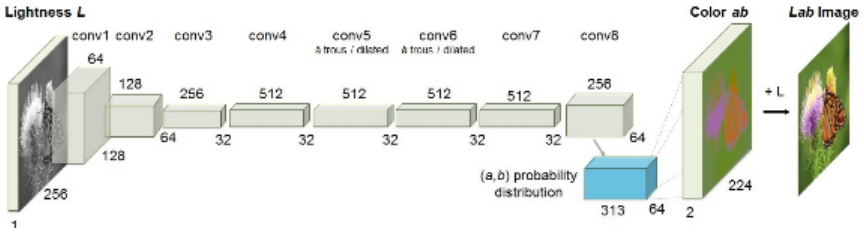


Figure 5: Test images from CIFAR10 evaluated by the model. Left: predicted color. Right: actual color.



(a) Fusion network from Ishikawa et al.[5]



(b) Classifier network from Zhang et al.[4]

Figure 6: Networks proposed in related work

[3] PyTorch project authors. Smooth L1 Loss. `https://pytorch.org/docs/stable/nn.html#smoothl1loss`

[4] R. Zhang, P. Isola, and A.A. Efros. Colorful Image Colorization. arXiv:1603.08511 [cs.CV]

[5] S. Iizuka, E. Simo-Serra, and H. 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 (Proc. of SIGGRAPH 2016), 35(4):110, 2016.*