Deep Image Colorization with Classification

Arlene Siswanto
siswanto@mit.edu

Pramoda Karnati
pkarnati@mit.edu

***Abstract*— Image colorization is a well studied topic in computer vision. Many approaches have been suggested to complete this task efficiently, from Generative Adversarial Networks to deep Convolutional Neural Networks. In this work, we discuss existing solutions and the intuition behind various designs. We then propose a colorization pipeline that involves passing in a black-and-white image into a classifier, which classifies the image into one of several predefined categories, then introducing a category-specific deep CNN to color the image. From our implementation, we found that creating a separate model for each specific category improved the system's performance for those input images given that they were classified properly.**

I. Introduction

Image colorization of black-and-white images is an interesting recent area of research in computer vision. The applications of this are endless, from video restoration to image enhancement. Throughout recent literature, many different methods have been suggested to approach this problem. Suggested methods range from using deep Convolutional Neural Networks to Generative Adversarial Networks to learn how to colorize black and white images.

CNNs are popular for image classification problems due to their capacity to learn patterns, edges, colors, and shapes within images. Recent literature has shown that these characteristics can be expanded to image colorization methods. In this work, we base our approach on a paper[1] by Baldassarre et al. (2007). describing their approach to solving this problem using a deep Convolutional Neural Network. The model was trained on images from the MIT CVCL Open Country Dataset.

Much of the literature as per our research generally trained a single image colorization model on a large dataset of images. In this work, we propose instead a colorization solution that involves training a model on specific categories of images and, given a new black-and-white image, apply a classification network on the image to determine which colorization model to apply. Training the model specifically for each category seemed to perform well across a variety of different categories.

II. Related Work

Much work has been devoted to the task of image colorization. We base our approach off of a colorization model proposed by Baldassarre et al.[1] (2007). Using an encoder-decoder network in combination with a feature extractor, they were able to generate near photo-realistic images from their network. Performance of the model was evaluated using a Turing Test, in which users were presented with colorized images and were asked to determine if they were fake or real images. Their results were increasingly promising; in some cases, real-perception achieved 80%.

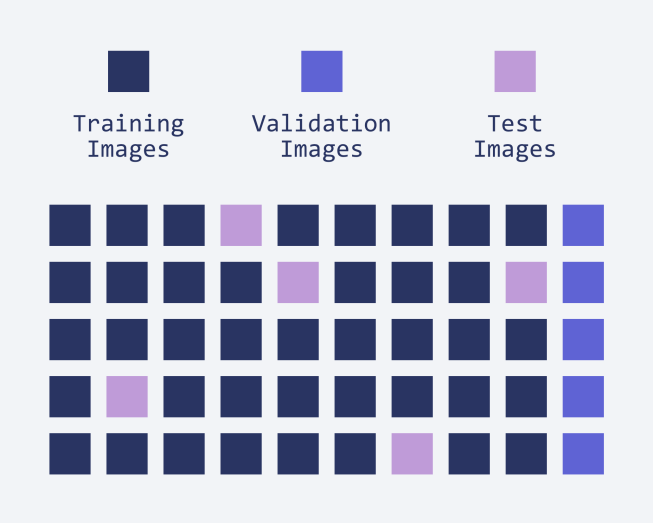

Fig. 1. The dataset contains roughly 1,000 images for each of 20 categories that must be preprocessed. The images are resized, reshaped, then randomly separated into training data, validation data, and test data.

Hwang and Zhou[3] (2016) propose another such CNN based model. They assess the performance of both a classification and regression network scheme and found that the classification network scheme performed better than the regression network scheme overall.

In another related work, Isola et al.[5] (2016) propose another deep convolutional neural network model for image colorization; this was able to fool humans on 32% of trials.

III. Dataset

For our dataset, we scraped roughly 1000 images for each of 20 predefined categories our model is expected to handle. The categories we decided to work with are: animal, autumn, beach, bird, bus, christmas, city, dog, dress, flower, food, forest, fruit, house, landscape, mountain, person, road, sunset, and waterfall.

A. Image Scraping

To obtain our images, we decided to scrape `https://unsplash.com/`—a website that offers free and high-quality images for many different categories. We scraped this website to obtain images for our dataset in utilizing Selenium, an automated web driver, and HTML

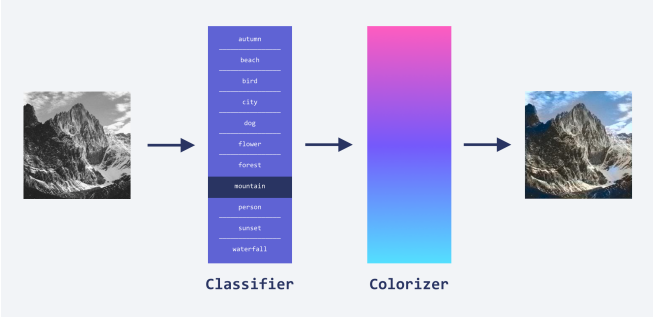

Fig. 2. Our pipeline. Black-and-white images are first fed into a classifier, which classifies the image into one of 20 predefined categories. It then colorizes the image using the model crafted for that specific category.

parsing. We obtained about 1000 images for each predefined category.

B. Preprocessing

The next task was to preprocess the data to ensure that the images are in a format compatible with and readily-usable for our model. We wanted to ensure that size would not be a factor when training the network. We reshaped and resized our scraped images to cropped squares of size `256 x 256px`. Then we went through the dataset and randomly added each image to either the training set or the test set. We made sure there would be no overlap between the training and test sets in order to convince ourselves that the model was not simply memorizing the images and, rather, learning. We set aside some of the training set data to be used as validation data. For each image of the randomly-selected test set, we saved both the black-and-white and colored versions of the image for comparisons of our methods.

IV. Approach

In our work, we propose adding a layer of classification before applying the colorization model to an image. Rather than creating a universal neural network capable of handling images from all categories, we wanted to experiment with image colorization models trained specifically for images from one certain category. The intuition behind this is that, by providing contextual information about an image, the model is able to more accurately colorize an image based on related images. As described above, much of our research showed that many methods tended to train the network on a large dataset of images or integrate information about categories as an intermediary layer in the middle of the neural network.

Our approach can be described in two stages, as follows:

- Classify a grayscale image to a specific category
- Apply the corresponding colorization model based on the category predicted

As detailed in Figure 2, our pipeline contains two main components: a colorizer and a classifier. We use Keras[2] for our experiment purposes. These two parts are described below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_49 (Conv2D)           (None, 256, 256, 64)      640
_________________________________________________________________
conv2d_50 (Conv2D)           (None, 128, 128, 64)      36928
_________________________________________________________________
conv2d_51 (Conv2D)           (None, 128, 128, 128)     73856
_________________________________________________________________
conv2d_52 (Conv2D)           (None, 64, 64, 128)       147584
_________________________________________________________________
conv2d_53 (Conv2D)           (None, 64, 64, 256)       295168
_________________________________________________________________
conv2d_54 (Conv2D)           (None, 32, 32, 256)       590080
_________________________________________________________________
conv2d_55 (Conv2D)           (None, 32, 32, 512)       1180160
_________________________________________________________________
conv2d_56 (Conv2D)           (None, 32, 32, 256)       1179904
_________________________________________________________________
conv2d_57 (Conv2D)           (None, 32, 32, 128)       295040
_________________________________________________________________
up_sampling2d_13 (UpSampling (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_58 (Conv2D)           (None, 64, 64, 64)        73792
_________________________________________________________________
up_sampling2d_14 (UpSampling (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_59 (Conv2D)           (None, 128, 128, 32)      18464
_________________________________________________________________
conv2d_60 (Conv2D)           (None, 128, 128, 2)       578
_________________________________________________________________
up_sampling2d_15 (UpSampling (None, 256, 256, 2)       0
=================================================================
Total params: 3,892,194
Trainable params: 3,892,194
Non-trainable params: 0
_________________________________________________________________
None
```

Fig. 3. Colorization network architecture.

A. Colorizer

For each category of image, we create a colorizer based off of an existing model architecture as described in literature. We propose the use of a Convolutional Neural Network to train the colorization network. The network consists of several convolutional layers to extract features from the input. The network first learns simple patterns, such as lines, edges, and shapes, and slowly builds to learn more complex patterns from the input image. It is then followed by several upsampling layers to recover the original image. While CNNs usually include max-pooling layers to increase informational density, this model does not make use of them as they distort the image; instead, it uses a stride of 2 to decrease the size of the image. The network uses an `Adam` optimizer and uses MSE as the loss function.

Image colorization networks attempt to learn the colors for a grayscale image. This is a natural problem for the *Lab* color space, which contains one channel L for luminance (in which the darkest black is 0 and the lightest white is 1) and two channels a and b for green-red and blue-yellow color,

Fig. 4. Colored image separated into *Lab* color space[4]. From left to right: (1) L luminance channel, (2) a green-red channel, (3) b blue-yellow channel. Source: https://blog.floydhub.com/colorizing-b-w-photos-with-neural-networks/

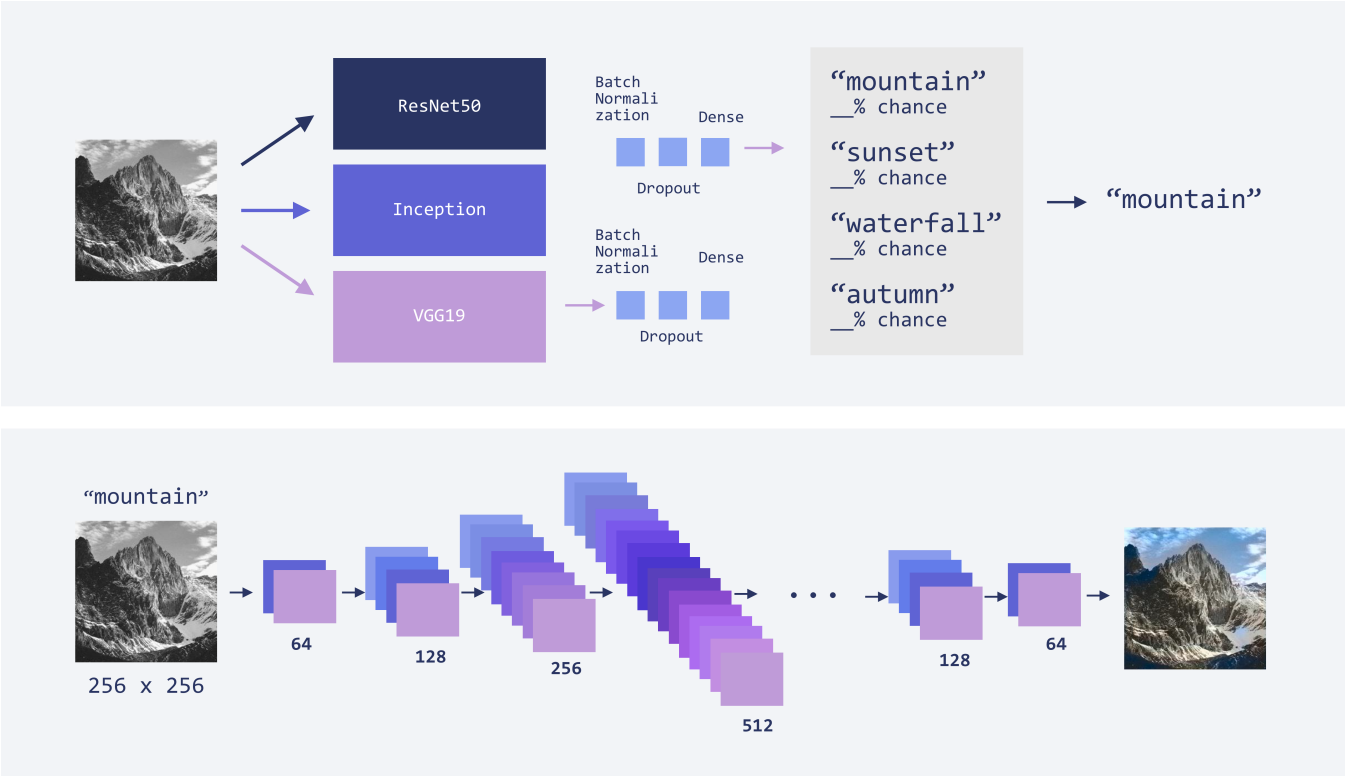

Fig. 5. (Top) Testing three pre-trained classifiers and selecting VGG19 as our classifier. (Bottom) Convolutional neural network that is trained to take in a category and black-and-white image and output a colored image. We create one such network for each category.

respectively. Our given image is black and white, so we can easily convert it to Lab and use it as the L layer for our final image. The task is to then train the model to learn the a and b color channels. Tackling the colorization problem in Lab rather than RGB allows the model to predict just two, rather than three, different image layers.

To train the model, we input in a set of RGB colored images, convert them into Lab images, then feed the L channel of the images as the input of the neural network and the a and b channels as the output. The model then tries to learn the mapping between the L channel and its two color layers, ab-channel. As neural networks typically learn, the network guesses weights randomly then periodically updates these weights as it learns from inputted images.

For our experiment, we train the model for 500 epochs on each image category, with 25 steps per epoch and a batch size of 25 images. The entire dataset passes through the neural network once per epoch. While training for 500 epochs, we periodically save the model every 50 epochs, which allows us to visualize the progression of the model and select the optimal number of epochs to run the model on. Our resulting network can be used to predict the color mapping for test images with similar L-channels.

B. *Classifier*

For our classifier, we decided to train existing classification networks on our grayscale images in order to categorize each image. We experimented with ResNet50, Inception, and VGG19 on our set of images in order to find which has the best performance. We create a one-hot encoding for each image to represent its class as output for the model. For each pre-trained model, we remove the top-level classification layers, add batch normalization, prevent against overfitting using dropout, and include a dense layer. We use Softmax activation to obtain the confidences for each class per image. In order to test this method, we initially only trained each of the models on 6 categories of image data and found that the optimal classifier for our purposes was VGG19.

Fig. 6. Colorized mountain trained on 10, 50, 100, 200, 350, and 400 epochs, respectively.

Fig. 7. Examples of well-colorized images: (Top) Images colorized by a *forest* specific model; (Bottom) Ground truth.

Fig. 9. The network performs poorly when applying a model from one category to an image from a different category. (Top) Predicted; (Bottom) Ground Truth.

V. Results

We were able to obtain substantial results in training the colorizer network. Training for 500 epochs allowed us to obtain a pretty well-defined model for each specific category. Figures 7 and 8 show examples of well-colorized images using the colorization network. In Figure 7, we apply a colorization model specific to *forest* for many images, and we can see the results of the colorizer against the ground truth. In Figure 8, we see another category, *autumn*. These images are also well-colorized, but compared to ground-truth, they seem to be a bit more vibrant than the original image. This can be attributed to the fact that many images contained in this *autumn* category contained extremely vibrant reds and oranges; therefore, when the network was tested on these images, it used that gray to color channel representation.

We also used various epochs as benchmarks as the model trained to see how the colorization model performed after each benchmark. We can see the colorization results using an model trained for 10, 50, 100, 200, 350, and 400 epochs in Figure 6. We can see that the network actually performs quite well after 100 epochs, but the colors are generally muted compared to the ground truth. However, 400 epochs causes the colors to be much sharper than the ground truth. We see at epoch 200 that the image is mainly blue, at which point it was probably overtfitting to blue in the dataset. Overall, we found 500 epochs to be a sufficient number of epochs to obtain a valid model. We use a Turing test and ask humans to determine whether the pictures are real or fake, and we found that many people were mostly fooled by the generated images.

We also wanted to see how the model performed on colorizing images using the wrong category and found some pretty interesting results, detailed in Figure 9. We can see the colorized images on the top and the ground truth on the bottom. The first image is a *city* image colorized using an *autumn* category. Here we can see the vibrant reds and oranges from *autumn* applied to the image, which is clearly incorrect. The second set of images shows a *sunset* image colorized using a *mountain* category; the final set of images is a *flower* image colorized using a *mountain* category. The last set is very far away from the ground truth representation and unlike any real flower representation. The colorized image contains the blue and purple hues learned from images in the *mountain* category and therefore colorizes it incorrectly. From this, we can see the value in training a model specific to each category of images, as it allows the model to learn a more accurate mapping from gray to color channel representation of images.

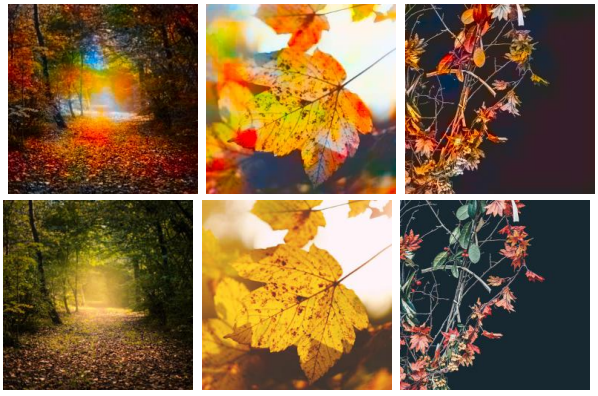

Fig. 8. Examples of well-colorized images. Though the outputted images do not necessarily look like the ground truth, the coloring seems at least plausible. (Top) Images colorized by a *autumn* specific model; (Bottom) Ground truth.

TABLE I
Top 3 Error

Category	ResNet50	VGG19	Inception
beach	0.371	0.319	0.423
city	0.905	0.15	0.65
autumn	0.302	0.396	0.45
waterfall	0.255	0.517	0.534
sunset	0.39	0.967	0.76
mountain	0.868	0.476	0.766

As for our classifier, we trained each pre-trained model on our specific categories for 500 epochs and found that the results were not as great as expected. For our image set, we found that VGG19 was able to perform the best on the classification task. We detail the Top3 error for each category on a test set of images using ResNet50, VGG19, and Inception in Table I. We see that the Top 3 Classification error rates for each category of images is especially high. We think that a plausible reason for this is the fact that all of the images from these categories are all somewhat similar. For example, there are some images from the *autumn* category that contain mountains, and images from the *sunset* and *autumn* category contain very similar colors and images. This might have led to poor model performance on the test set.

VI. Conclusion

Introducing a classification component into the traditional colorization model improved the quality of our colorized images. This can likely be attributed to the idea that images from a specific category typically share common color palettes, vibrancies, and shapes. We tested this hypothesis by scraping roughly 1,000 images from 20 categories, preprocessing this data, then feeding the images into the VGG19 image classifier then to the category-specific image colorizer. Both the classifier and the colorizer were trained for 500 epochs. We were impressed by the results: the colorized images our model predicted were reasonably close to the expected output or, at least, somewhat plausible. Future work could include improving the results of the classifier, since the classifier seems to be the limiting factor of our solution — the colorizer itself does very well on a single category. One potential solution to this is to separate the images into more well-defined categories. For example, an image of the sun setting behind the mountains would be colored differently if classified as a sunset as opposed to a mountain.

VII. Contribution

For this project, the work was split up as follows:

- Arlene: Image scraping, preprocessing data, colorizer setup, diagrams
- Pramoda: Colorizer tuning, classifier
- Both: Running experiments and interpreting results

For this project, I worked on several components. The first thing I did was create a scraper that automated web browser scrolling using Selenium and parsed the HTML using BeautifulSoup to find relevant image links. I then structured the data into a readily-usable format separated into training, validation, and test data. After reading a few papers on image colorization, I also implemented a basic image colorizer based on methods proposed by the readings.

VIII. Acknowledgment

We would like to thank the entire course staff of 6.819: Advances in Machine Learning at Massachusetts Institute of technology for providing us with the knowledge, tools, and resources necessary for this research. Specifically, we would like to thank Professors Bill Freeman, Antonio Torralba, Phillip Isola, and all of the TA's for their continued support in this process.

References

[1] F. Baldassarre, D. G. Morn, and L. Rodcs-Guirao, Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2, CoRR, vol. abs/1712.03400, 2017, software available https://github.com/baldassarreFe/deep-koalarization. [Online].

[2] F. Chollet, (2015) Keras, GitHub. https://github.com/fchollet/keras Available: https://arxiv.org/abs/1712.03400

[3] J. Hwang and Y. Zhou, Image colorization with deep convolutional neural networks, Stanford University, Tech. Rep., 2016. [Online]. Available: http://cs231n.stanford.edu/reports2016/219 Report.pdf

[4] E. Wallner, "Colorizing BW Photos with Neural Networks," FloydHub. 2017. [Online]. Available: https://blog.floydhub.com/colorizing-b-w-photos-with-neural-networks/

[5] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. ECCV, 2016