Synopsis

In the past few years, instant image colorization has been made widely accessible and popular online. Before, it would take digital artists numerous hours to convert a black and white photo into a colored one. The quality is still more lacking than if a human artist took hours to retouch it; however, the fact that it takes a DL model almost instantly to colorize an image is a huge merit. However, since the release of the publication 'Colorful Image Colorization' (R. Zhang, P. Isola, A. A. Efros), rapid BW image conversion was made possible and has been developing even more since then. Even 'retouched' movies where old black and white films are colorized have seemingly became more widespread and commonly released ever since this paper was released.

Since I have started ML projects, I have been thinking of doing a CNN project where I felt like I could generally use for my own personal sake. I started digging/searching for ideas but couldn't decide what project to try out until I saw some old photos my grandma had of herself young. She cherished the photos and I wanted to let her enjoy the photos – now with color. Although I could use an online image color converted, I thought it would be much fun trying out the project myself. There have been lots of projects/code online related to it so I would have plenty of sources to learn from. I have initially built my model; however, due to limitations in the dataset and computing time, the results were not to my liking. After digging further through – I realized that even the most advanced, well-trained model still had some deficiencies and I decided to treat my experience as a learning one. I ended up also downloading a caffe model to see how a fully trained model would function.

Background

Black and white images only require 1 channel containing a value between 0-255 for each pixel. For color images, its pixels are stored in RGB components. The 3 color channels are added for final colorized image. To convert a black and white image into a colored one, each pixel value would need be componentized into each of the 3 RGB channels.

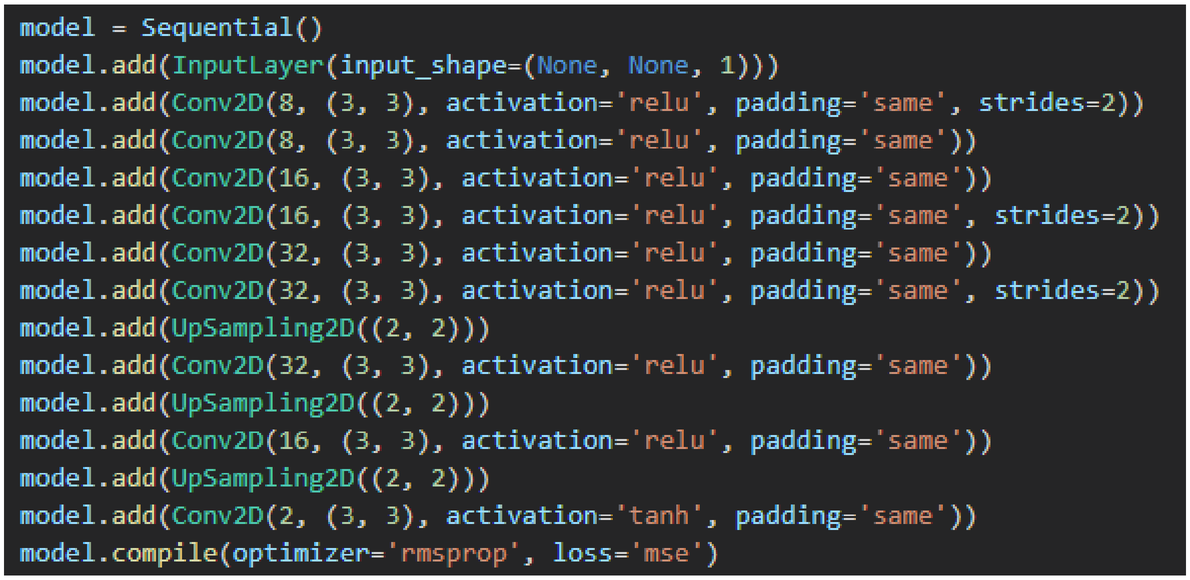
The core idea of image colorization utilizes using another color space that is not RGB to extract color feature/characteristic from the original BW image. Lab color space was the proposed and utilized for that purpose. To emulate the work done by Zhang el al., the Lab color space must be used. Lab color is thought to do a good job of how humans perceive color.

Lab color space has 3 channels like RGB color space, but the channels are different. L channel is for lightness intensity, a channel is for green-red, and b channel is for blue-yellow. The L channel deals with the BW information from the input BW image. The a and b channel are responsible for the colors; thus, the model will be used to train/predict a and b channels. The combined parameters for L, a, and b channels are used to generate the output RGB image.

Dataset

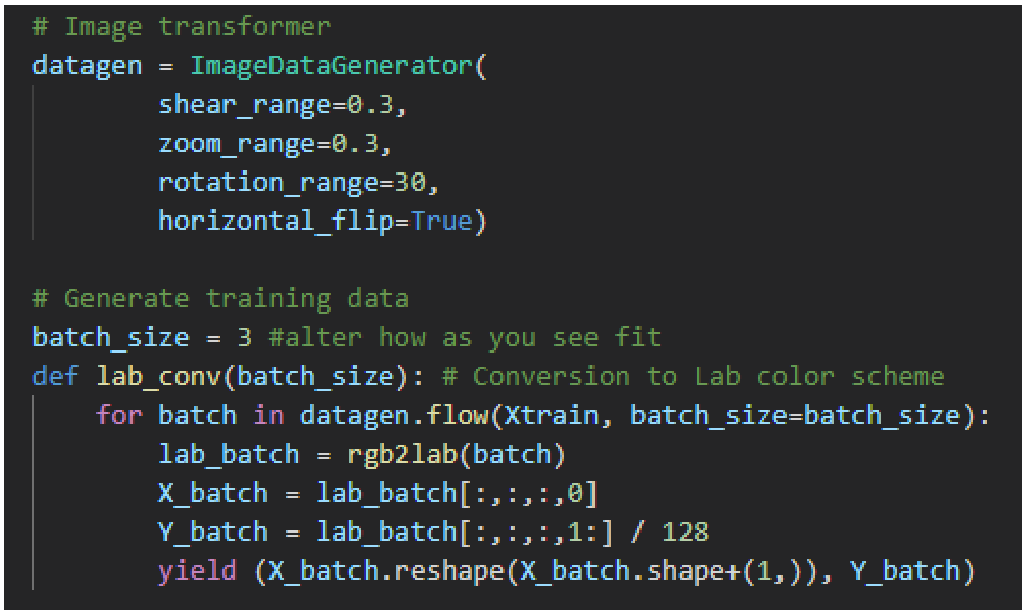
For my initial model, the images were pre-processed to be 256 x 256 pixels. Also, the value of the pixels were checked to see if they were in the range of 0-255 as per typical image recognition applications. If the pixel value of the image was out of range, they were made sure to be normalized. The dataset and original framework of the model was from/inspired by https://github.com/guilbera/colorizing and https://github.com/emilwallner/Coloring-greyscale-images. Due to its large size unable to fit on git, the model needs to be downloaded from https://dropbox.com/s/dx0qvhhp5hbcx7z/colorization_release_v2.caffemodel?dl=1 and copied in the Caffe Model folder.

The core idea is to train the model so that it knows how to colorize/learn features based on its Lab gradient representation. The model is to function as an encoder/decoder used commonly for CNN purposes (feature selection and extraction). Convolutional filters are used to turn one layer of the L channel into two layers of and b.



```
model = Sequential()
model.add(InputLayer(input_shape=(None, None, 1)))
model.add(Conv2D(8, (3, 3), activation='relu', padding='same', strides=2))
model.add(Conv2D(8, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(16, (3, 3), activation='relu', padding='same', strides=2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', strides=2))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(2, (3, 3), activation='tanh', padding='same'))
model.compile(optimizer='rmsprop', loss='mse')
```

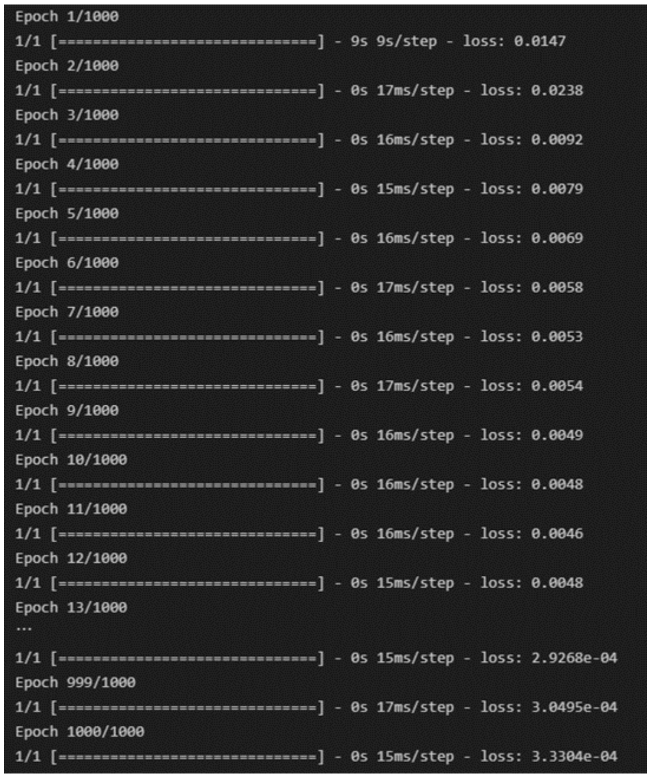
The initial training images were converted into Lab color scheme. The L channel is the image in black/white so a and b channels will need to be predicted to get it as close to the original L channel as possible. The resultant model was used on the test dataset to evaluate its accuracy. A caveat is that the accuracy is based in the Lab space so the low accuracy number may not translate into a well colorizing model.



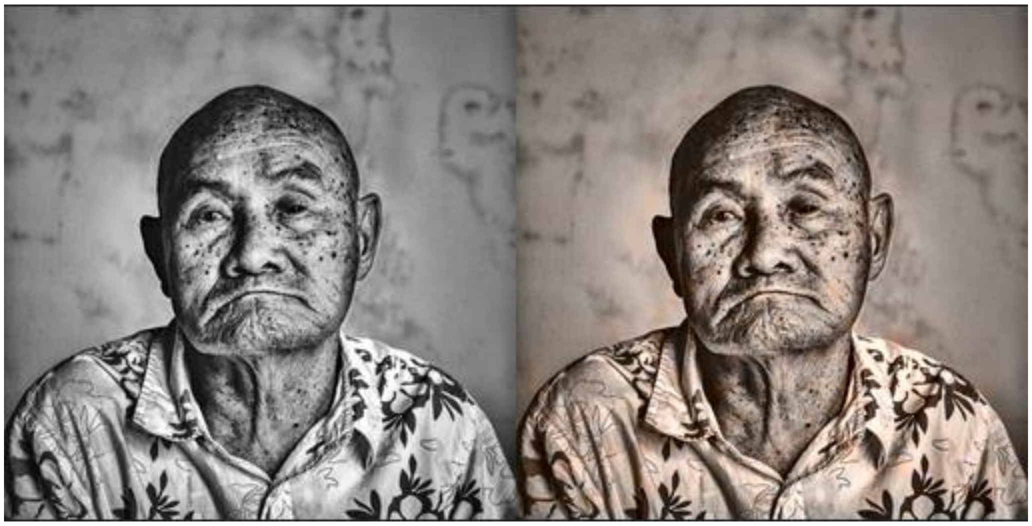
```
# Image transformer
datagen = ImageDataGenerator(
        shear_range=0.3,
        zoom_range=0.3,
        rotation_range=30,
        horizontal_flip=True)

# Generate training data
batch_size = 3 #alter how as you see fit
def lab_conv(batch_size): # Conversion to Lab color scheme
    for batch in datagen.flow(Xtrain, batch_size=batch_size):
        lab_batch = rgb2lab(batch)
        X_batch = lab_batch[:,:,:,0]
        Y_batch = lab_batch[:,:,:,1:] / 128
        yield (X_batch.reshape(X_batch.shape+(1,)), Y_batch)
```

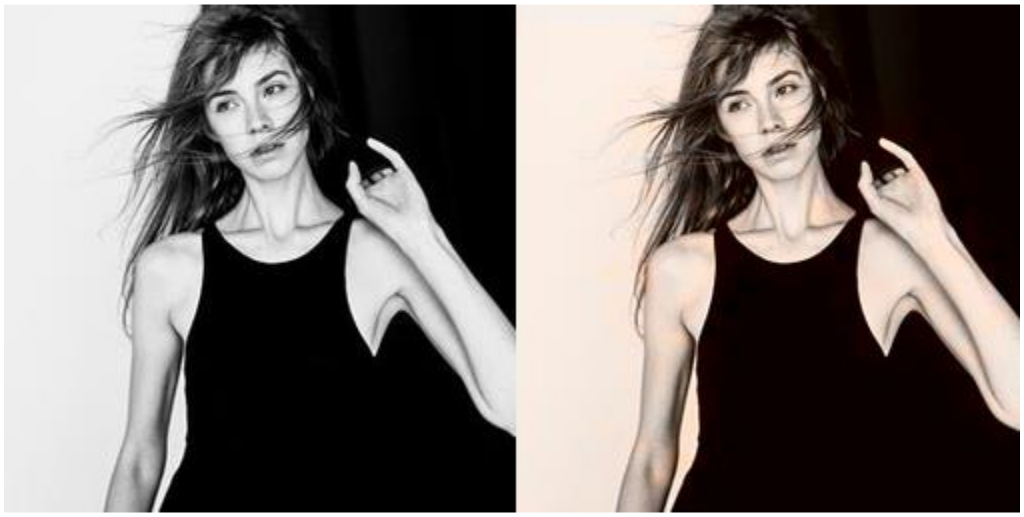
Summary of Dataset

The model showed extremely minimal loss. However, the resultant image was not satisfactory due the lack of images/computing power as mentioned in the synopsis. When I started this project, I had severely underestimated how much images/data I would require getting a satisfactory coloring. This statement was reflected by the github contributors I mentioned prior.



```
Epoch 1/1000
1/1 [==============================] - 9s 9s/step - loss: 0.0147
Epoch 2/1000
1/1 [==============================] - 0s 17ms/step - loss: 0.0238
Epoch 3/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.0092
Epoch 4/1000
1/1 [==============================] - 0s 15ms/step - loss: 0.0079
Epoch 5/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.0069
Epoch 6/1000
1/1 [==============================] - 0s 17ms/step - loss: 0.0058
Epoch 7/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.0053
Epoch 8/1000
1/1 [==============================] - 0s 17ms/step - loss: 0.0054
Epoch 9/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.0049
Epoch 10/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.0048
Epoch 11/1000
1/1 [==============================] - 0s 16ms/step - loss: 0.0046
Epoch 12/1000
1/1 [==============================] - 0s 15ms/step - loss: 0.0048
Epoch 13/1000
...
1/1 [==============================] - 0s 15ms/step - loss: 2.9268e-04
Epoch 999/1000
1/1 [==============================] - 0s 17ms/step - loss: 3.0495e-04
Epoch 1000/1000
1/1 [==============================] - 0s 15ms/step - loss: 3.3304e-04
```

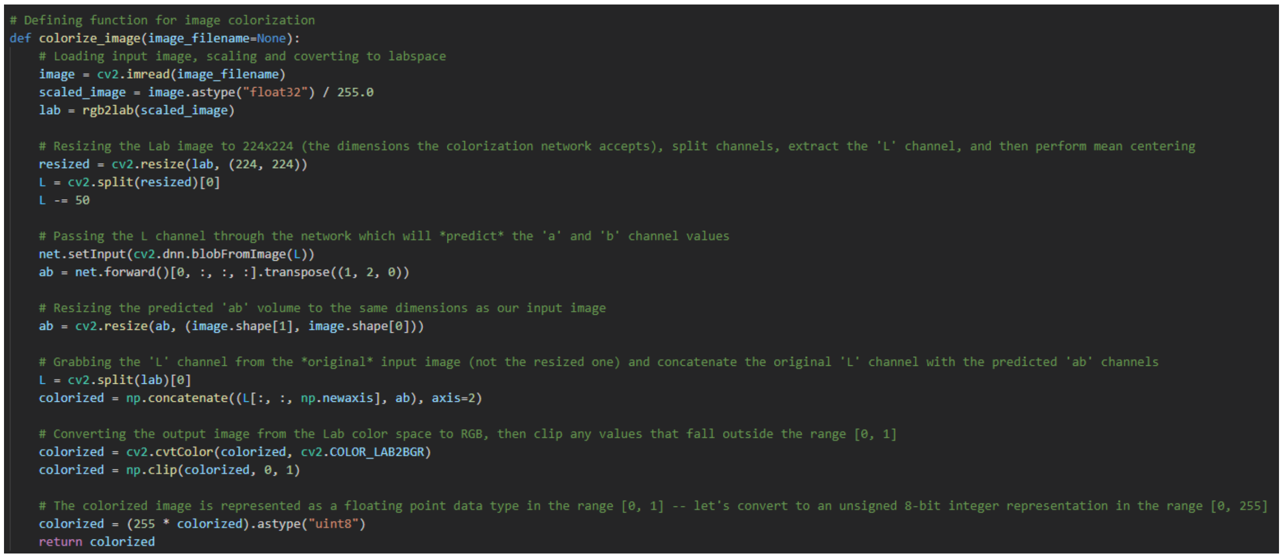




In the above image, the model was unable to fully color the initial BW image. However, it had started to pick up characteristics/features where it started to detect/identify humans in the center of the image. Outlines of the hair and patches of skin was showing shades of the correct color. The overall blacks such as the dress were not touched and was remained as the correct color.

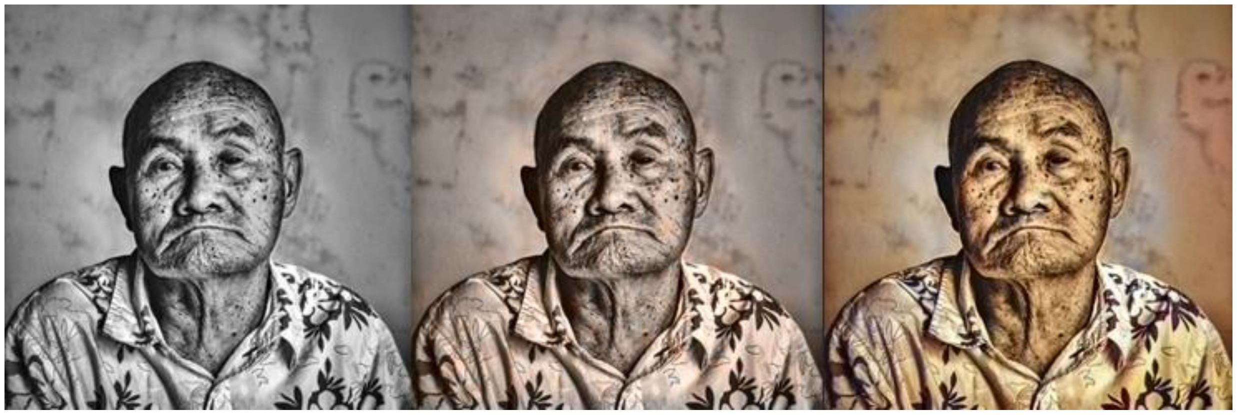
Utilization of Transfer Learning (Caffe Model)

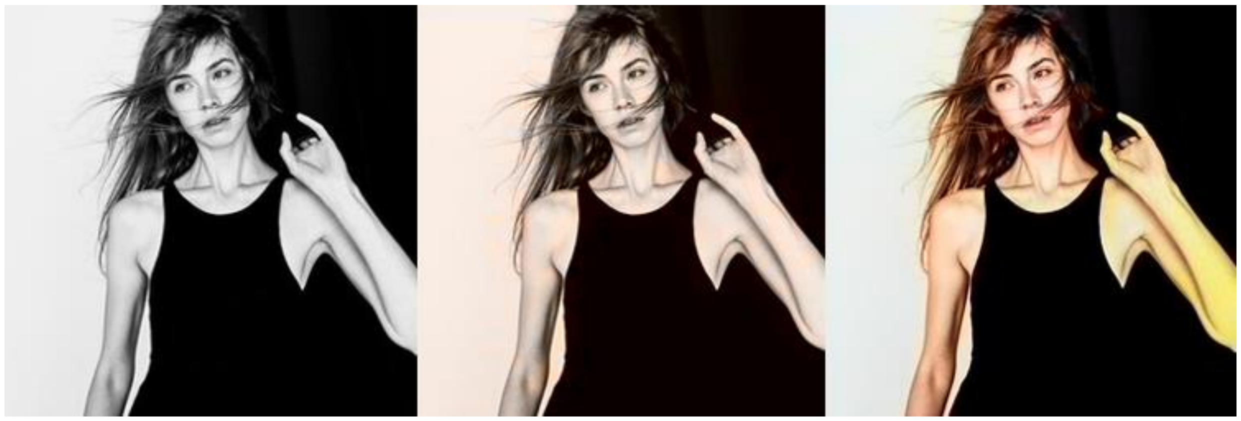
To see how the process would look with a fully trained/mature model, I downloaded the pre-trained Caffe model from dropbox.com/s/dx0qvhhp5hbcx7z/colorization_release_v2.caffemodel?dl=1. Using the packages provided by https://github.com/PySimpleGUI/PySimpleGUI-Photo-Colorizer, I used the Caffe model to colorize the above two images as well. The test images were once again converted to Lab and the Caffe model was used to predict the a and b channels with the given L channel obtained from the input images.



```python
# Defining function for image colorization
def colorize_image(image_filename=None):
    # Loading input image, scaling and coverting to labspace
    image = cv2.imread(image_filename)
    scaled_image = image.astype("float32") / 255.0
    lab = rgb2lab(scaled_image)

    # Resizing the Lab image to 224x224 (the dimensions the colorization network accepts), split channels, extract the 'L' channel, and then perform mean centering
    resized = cv2.resize(lab, (224, 224))
    L = cv2.split(resized)[0]
    L -= 50

    # Passing the L channel through the network which will *predict* the 'a' and 'b' channel values
    net.setInput(cv2.dnn.blobFromImage(L))
    ab = net.forward()[0, :, :, :].transpose((1, 2, 0))

    # Resizing the predicted 'ab' volume to the same dimensions as our input image
    ab = cv2.resize(ab, (image.shape[1], image.shape[0]))

    # Grabbing the 'L' channel from the *original* input image (not the resized one) and concatenate the original 'L' channel with the predicted 'ab' channels
    L = cv2.split(lab)[0]
    colorized = np.concatenate((L[:, :, np.newaxis], ab), axis=2)

    # Converting the output image from the Lab color space to RGB, then clip any values that fall outside the range [0, 1]
    colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
    colorized = np.clip(colorized, 0, 1)

    # The colorized image is represented as a floating point data type in the range [0, 1] -- let's convert to an unsigned 8-bit integer representation in the range [0, 255]
    colorized = (255 * colorized).astype("uint8")
    return colorized
```

The rightmost images for the two colorizations shows the work done by the Caffe model. It definitely performs better as the woman's hair color and complexion looks much more natural. The old man's colorization could be slightly better as his color saturation still leaves room for improvement. However, it is still a good improvement over the initial, quickly trained model.





As the results were satisfactory, I used the model to colorize my grandma's old photo as well. You look full of color grandma!