Robotics Project

Colorization of Gray-Scale Images using OpenCV and Deep Learning

Group Members:

Shehroz Ashraf	294966
Zoraiz Ali	282629
Sannan Abbasi	293786

Introduction:

Colorization of grayscale images is the process of adding color to black and white images. This is a difficult task for humans as it requires a deep understanding of the scene depicted in the image and the ability to select appropriate colors for different objects and regions in the image. However, with the advancement of deep learning techniques, it is now possible to automate this process using neural networks.

Objective:

The objective of this project is to design and implement a deep learning model that can accurately colorize grayscale images. The model should be able to generate realistic and coherent colors for different objects and regions in the input image based on their visual appearance and context.
Once the model is trained, we will test it on a set of grey scale images to evaluate its performance. We will measure the accuracy of the model by comparing the generated color images with the ground truth color images. We will also evaluate the model qualitatively by visual inspection of the generated images.
We expect that the proposed model will be able to generate high quality colorized images that are similar

The Caffe Model:

Initially, we were going to use the Caffe model as said in the progress report. Caffe provides a fast and flexible architecture for training convolutional neural networks (CNNs) and other types of deep neural networks. It has a modular design that allows users to easily modify and extend the model architecture and training process. Caffe also provides support for GPU acceleration, which allows for faster training and inference on large datasets.
One of the main features of Caffe is its support for layer-based model design, which allows users to easily define the architecture of their CNNs by stacking different types of layers. These layers can be customized to perform different operations on the input data, such as convolution, pooling, normalization, and activation.
In addition to its support for model design and training, Caffe also provides tools for model deployment and serving, which allow users to deploy their trained models in production environments and serve predictions on demand.

Alternative Approach: Generative Adversarial Networks(GANs)

RGB vs LAB:

One of the more important aspects that we need to shed light on is that we colorized the grey images using the LAB format rather than the traditional RGB format. RGB (Red, Green, Blue) and Lab* (Lightness, a color component, and b color component) are two different color models that are used to represent and manipulate colors in digital images.
RGB is an additive color model that is based on the idea that different combinations of red, green, and blue light can be used to create a wide range of colors. In an RGB color model, each pixel in an image is represented by three 8-bit values that correspond to the intensity of the red, green, and blue channels. The range of values for each channel is 0-255, where 0 represents the absence of that color and 255 represents the maximum intensity of that color.
Lab*, on the other hand, is a color space that is based on the human perception of color. It is a device-independent color model, which means that the colors represented in this model will look the same on any device or display. In an Lab* color model, each pixel in an image is represented by three 8-bit values that correspond to the lightness (L*) of the color, and the a and b color components. The component represents the position of the color along the red-green axis, and the b component represents the position of the color along the blue-yellow axis.
One of the main differences between RGB and Lab* is the way they represent colors. RGB is based on the physical properties of light and how it is perceived by the human eye, while Lab* is based on the human perception of color itself. This means that Lab* is more perceptually uniform, which means that the distance between two colors in the Lab* space is proportional to the perceived difference between those colors.
Another reason we used the LAB format was that it is far easier to predict the combination of two numbers (A and B) as compared to predicting the R, G, and B values of an image (3 numbers). When looking at it mathematically, if we have 256 choices for a single number for the RGB format, there are 16 million possible combinations. However, there are 65000 combinations that can be predicted for 2 numbers of the LAB format. Intuitively, it is far easier for the computer when using the LAB format.

Methodology:

In our project, we use a generative adversarial network (GAN) to generate color images from grayscale images. The GAN consists of two models: a generator and a discriminator. The generator model takes in a grayscale image and outputs a color image in the Lab color space, with separate channels for the *a and *b components. The discriminator model receives the generated color image, as well as the original grayscale image, and determines whether the color image is real or fake. The discriminator is also trained on real, non-generated color images in the Lab color space to learn to distinguish between real and fake images. The generator and discriminator models work together to improve the quality of the generated color images.

In our approach, we use two loss functions to train the GAN: an L1 loss function, which treats the problem as a regression task, and an adversarial loss function, which allows the GAN to learn in an unsupervised manner by evaluating the realism of the generated images. The L1 loss function measures the difference between the generated images and the ground truth, while the adversarial loss function assigns a score to the generated images based on how "realistic" they appear. Using both loss functions together allows us to effectively solve the problem of generating color images from grayscale images.

We started by taking the image colorizer paper as our baseline. This would act as the base architecture for our GAN, as the provided generator and discriminator are more than capable of colorizing images provided they have a large enough dataset to use. Most of the code for the base architecture is taken from here.

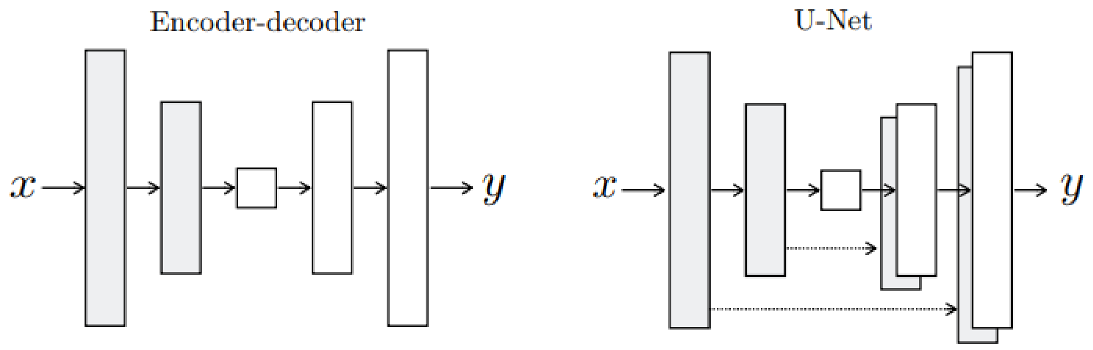


Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Starting with preprocessing, we resize and horizontally flip the images (only during training) and then read in an RGB image. We convert the image to the Lab color space and separate the first (grayscale) channel as the input for the models and the color channels as the targets. We use these preprocessed inputs and targets to create data loaders for the models. The code should be self-explanatory for those familiar with image preprocessing and data loading techniques.

For the generator, we use a U-net model. As the U-Net model processes the input image, it reduces the spatial dimensions by a factor of 8, resulting in a 1x1 image in the middle of the network. The model then uses up-sampling layers to increase the spatial dimensions back to the original size of the input image, producing a 2-channel output image.

The discriminator model is designed to classify images as either real or fake. It consists of a series of Convolutional-Batch Normalization-Leaky ReLU blocks stacked together. The first and last blocks do not include normalization layers, and the final block does not have an activation function, as it is incorporated into the loss function used for training. The model uses the stacked blocks to make a determination of the authenticity of the input image.

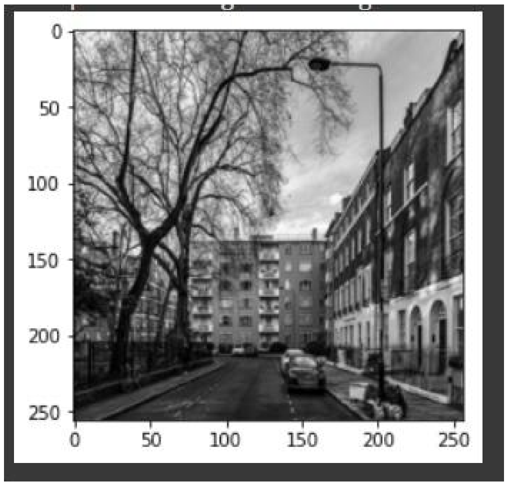
Initial training was done for 50 epochs, with better results being shown after the 20th one. However, it was terminated early in favor of a new approach.

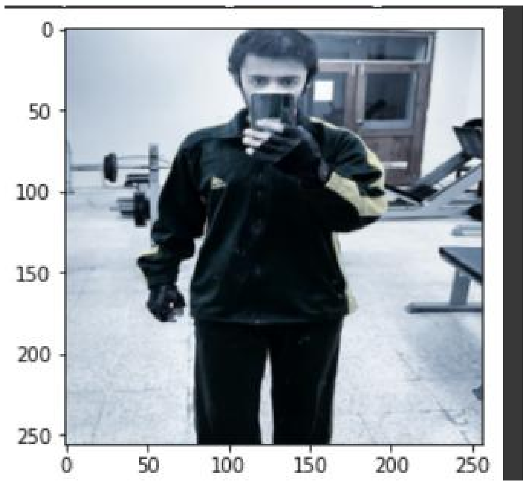
In this approach, we pre-trained the generator using a resnet18 model. The pretraining allowed for better, more coherent images in a lesser number of epochs.

Lastly, we wrote some code that would allow the user to upload a custom image to their drive. Once they'd have the image path, they'd give it to the **path** variable in section 3. Once that's done, they'd execute the remaining cells to get the colorized image as an output. Some results have been attached to this report, and the collab file is available in the appendix.

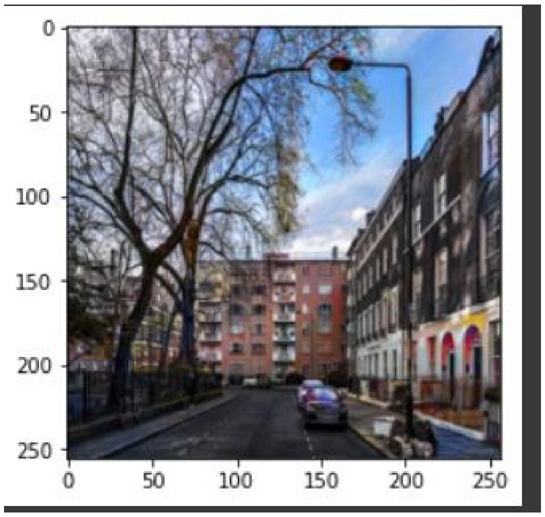
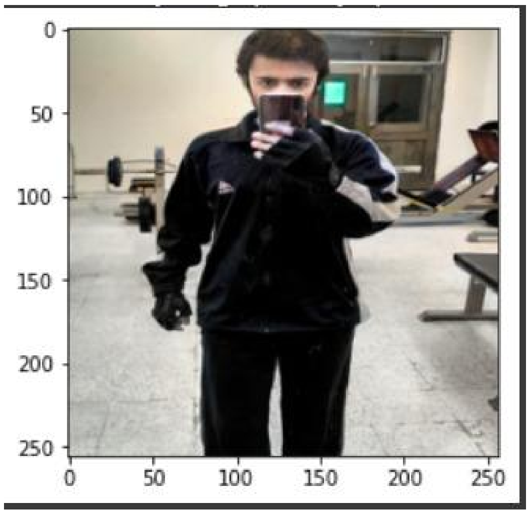
Results using GAN:

<u>Grey Scale</u>





<u>Predicted Color</u>

Conclusion:

Through this project we were able to achieve our goal and objective of automatic image colorization using deep learning. The results we got from both our models were very satisfying and accurate. Moreover, we learned about GAN as well and played around with images using basic OpenCV functions. All in all, this project proved to be a great learning experience and it further refined our understanding of deep learning and the different ways it can be used to do interesting stuff.

Appendix

1. https://colab.research.google.com/drive/1GNBFh_R-BGNFby7TXBFaiQ4xKc2NhsxY?usp=sharing (The code blocks come with detailed text highlighting the working)
2. https://colab.research.google.com/drive/1_nQM_UXbDkO3FBS9___F9zwv1GInfVrm#scrollTo=py_UoSP0eSEG (.CAFFE MODEL)