EECS 442: Image Colorization

Swati Bhageria, Devishi Suresh, Mathieu Noguier, and Eleazar Vega

1. Introduction

With color photography only becoming popular in the 1970's, the vast majority of photos taken are in black and white. Multitudes of photographs documenting history have never been seen in color, giving us an inability to feel as we have truly visited that time. Pictures documenting humanity's greatest achievements like the invention of flight or cities transformations post industrial revolution have not been viewed as our eyes would see them. Likewise, humanity's lowest achievements have not been documented in that manner either, even though we know how important it is to learn from our history in order to not make the same mistakes. Additionally, there is so much nostalgia that can be found in the idea of converting grayscale images to color. Whether it be seeing older family members or how the world used to be. The idea of seeing someone's grandparents at a much younger age in color would be remarkable and comforting, especially if the photo is of someone now longer around.

This is all possible now due to advancements in Computer Vision like CNN deep learning models and an increase in research which provides large and diverse datasets that make training more accurate and the models more robust.

Originally done through the three-color process which theorized that humans' eyes are able to see color due to the eye using three types of cone cells that are individually sensitive to different parts of the color spectrum. This was first suggested by James Clerk Maxwell.[find citation]. Below in Figure 1. Is the first color photograph done according to this theory.

Figure 1. Tartan Ribbon by Thomas Sutton

Another famous example of image colorization took place in 1917 through an innovation that allowed multiple filters to be used on the camera when taking the image. This innovative method was presented by an exposition done by russian photographer Sergey Prokudin-Gorsky and can be seen in Figure 2.

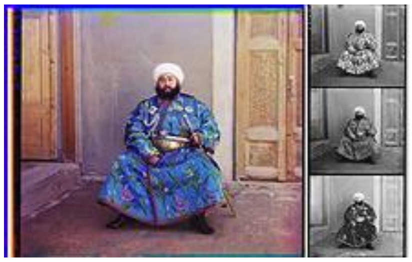
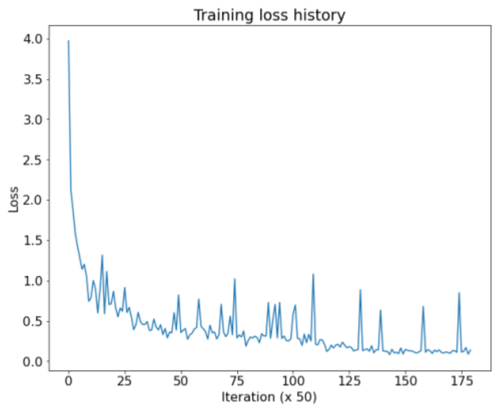


Figure 2. The Emir of Bukhara, Alim Khan by Sergey Prokudin-Gorsky

More modern related work is the Google funded DeepAI that enables users to colorize or restore old photos and also colorize video. DeepAI uses deep learning techniques to train on pairs of photos one being the greyscale photo and the other being the color photo. They allow users to submit images directly through a website and output the colorized image.

This paper proposes using deep learning to train and test on the SAMSUNG NEON dataset for image colorization. The SAMSUNG NEON dataset consists of 202 images of landscapes captured at different light settings. Taking the dataset through BasicNet, ResNet, UNet, and colorization models and encoders. Images are resized to (256, 256, 3) and converted to grayscale prior to feeding the dataset into the models to maintain consistency. Lastly, a comparative analysis is done which includes every model used and the same loss function throughout.

2. Approach



2.1 Preparing Data set

The dataset that we used to train the models was the Samsung Neon Dataset. We decided to use this training dataset because it provided a variety of different landscapes with a multitude of colors. This was perfect as it allowed the model to gain more experience with common colors that are often seen in nature. The dataset had around 4202 images so this dataset was not as computationally intensive to train it on.

2.2 BasicNet Architecture

The encoder consists of 3 (4x4) convolution layers with (32, 64, 128 filters) and ReLU activation. This is then followed by a batchnorm layer. The outputs of the batchnorm layer are then passed into 2 (3x3) Conv layers (128 filters each) with ReLU activation. Next, we implemented the decoder block which is comparatively small. 1 (4x4) transpose convolution with relu activation which is followed by batch normalization layer. This unit is repeated twice before the last output layer which is again a transpose convolution with output channels equal to 3.

2.3 ColorizationModel Architecture

The Colorization Model consists of the pretrained ResNet 18 encoder which we got from Pytorch. We use the first 6 layers of this model for feature extraction, i.e. the ResNet 18 encoder pretrained on ImageNet is used as the encoder block. Once the features are obtained from this, we construct the decoder block. It consists of a (3x3) convolution layer (filters=128) followed by Batchnorm and ReLU. This is then upsampled by a factor of 2 after which is passed through 2 blocks of (3x3) convolution layers, Batch Normalisation and Relu activation. After upsampling again by a factor of 2, it is passed once through the same block mentioned in the previous sentence. The output layer is a convolution layer which is upsampled one last time to match input dimensions.

Unet Architecture

The Unet model was open source code that we used to implement another model. Below is a representation of how the model works.

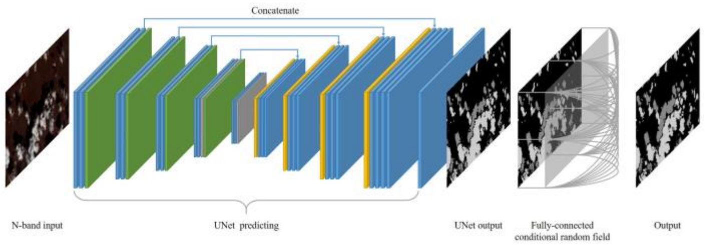


Figure N. Unet architecture visualization

ResNetUnet Architecture

The final model is the ResNetUNet model which is designed using inspiration from the ResNet and UNet architectures. We use the pretrained Resnet 18 model as the encoder and use this as our feature extractor. Further, these features are downsampled using blocks of convolution with relu activation which are then concatenated with outputs from the resnet18 model. In other words, we introduce skip connections similar to the UNet architecture with the difference here being that the features are now added from the ResNet encoder block. The decoder block is a simple UNet type decoder which upsamples these features to produce the required output.

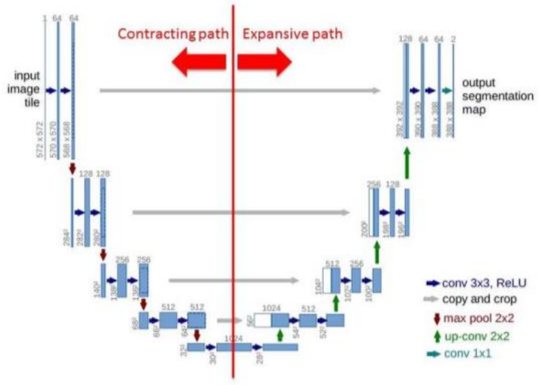


Figure N. UnetResNet diagram

3. Experiments

Data

When working on our experiments we used some of the images from the original dataset that we used to train it. We did so because we wanted to see how much of an impact training dataset had on its performance and we figured the best way to do so was to test it on something that it should be able to output something decent.

Metrics

When experimenting with the performance of the models that our group made, we wanted to use a combination of both qualitative and quantitative measurements to analyze the performance of them. The visual output of the models are extremely important for a project like this. Much of the problem in recent history of image colorization has been making it look as natural and realistic as possible. Simply looking at the output's performance through a numerical lens can be difficult as the average human might have trouble understanding. Nevertheless, we still felt that our quantitative output was still important so we can continue to improve our project.

Qualitative Findings

Our findings reflect the performance of all four of our models. Below our some of the best outputs for basicNet [fig 2], ResNetUnet [fig 3], Unet [fig 4], and ColorizationModel [fig 5]. We compared them to the same image in order to properly distinguish the performance of all four models. Looking at the results alone, we can conclude that the colorization model preformed the worst while the Unet one preformed the best. It may come as little surprise since it is the pretrained neural network, but if we are comparing only ours alone, we believe that ResNetUNet developes the best image.

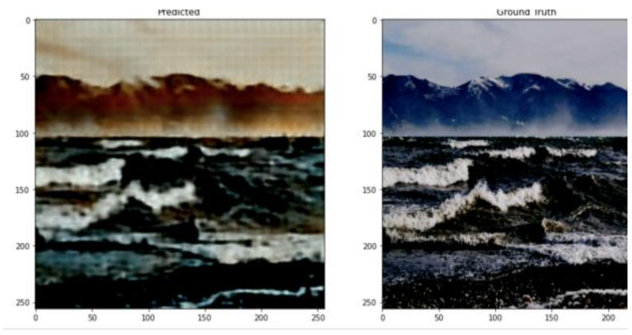


Figure 2. prediction (left) and GT (right) for BasicNet

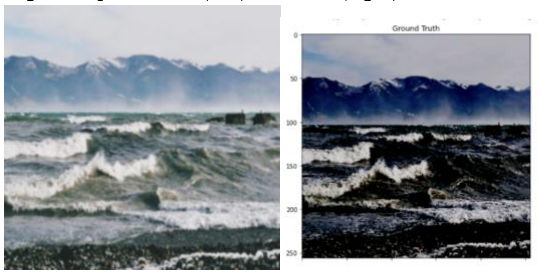


Figure 3. Prediction (left) and GT (right) for ResNetUnet

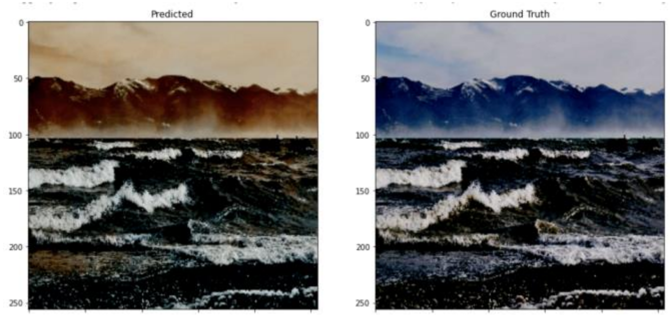


Figure 4. Prediction (left) and GT (right) for Unet

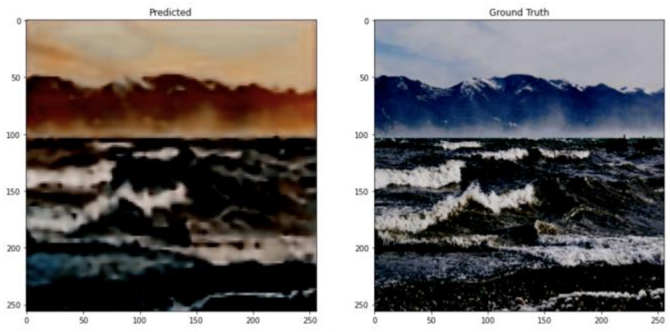


Figure 5. Prediction (left) and GT (right) for ColorizationModel

Quantitative Findings

For the quantitative side of our findings, we measured the average loss of our models along with the peak signal-to-noise ratio (PSNR). For the BasicNet model we were able to achieve a loss of 0.027654 and PSNR of 63.71329077744598 dB. For the colorizationModel we achieved a loss of 0.167537 and a PSNR of 55.88969848410901 dB. For the Unet model, we got a loss of 0.099848 and a PSNR of 58.137399765304274 dB. Lastly, for ResNetUNet model, we achieved a loss of 0.04 for PSNR of 72.3618. Based off of our number results, it appears that

4. Implementation

Most of the network implementations were constructed by our group with the exception of Unet which was a pretrained network.We used the pretrained network in combination with our own implementation of resNet to create a new model that takes inspiration from both. We didn’t need to spend much time gathering the data; we only used a single dataset as we felt it was sufficient for our purpose and less computationally expensive. The main libraries that we used were Pytorch, Matplotlib, and PIL. These libraries alone were enough to set up the networks and visualize them as well.

5. Conclusion

Our findings only show a fraction of the capabilities of image colorization. As deep learning advances, we will see more and more improvements in the way we can colorize the past.

ADDING ALL INFORMATION BELOW (HERE)

Introduction

- We want to solve image colorisation and we use deep learning methods for this
- Motivation is : old photos like ancient portraits and black and white videos and movies and pictures, we would like to see them in color format and get a better understanding of how that world looked like.
- Computer vision has methods like cnn's etc that can do this so we use them.
- Previous work (we need to add a few references and other people work to this)
- Summary : we do image colorsisation on our dataset and use 4 different deep learning architectures to solve this problem. We do a comparative analysis with these models and use the same loss function for all

(Add the other stuff mentioned in intro above)

Approach

- The general approach is described in the following steps :
    1. Prepare Dataset
    2. Design Model
        a. The model types
        b. The training setting
    3. Post Processing and Metric Calculation

Prepare Dataset

1. As mentioned before, we use the dataset from SAMSUNG NEON Challenge and it consists of 4202 images of landscapes captured at different light settings.
2. We resize the images to (256, 256, 3) and convert the input images to grayscale before feeding it to the model.
3. Since we choose to work and experiment in the RGB colorspace, we use the labels as RGB images which are also resized to (256, 256, 3). Both input and label are normalized.

Design Model

1. We start with BasicNet. The encoder consists of 3 (4x4) convolution layers with (32, 64, 128 filters) and ReLU activation. This is then followed by a batchnorm layer. The outputs of this is passed into 2 (3x3) Conv layers (128 filters each) with ReLU activation. This consists of the encoder block.
   The decoder block is comparatively small. 1 (4x4) transpose convolutions with relu activation is followed by batch normalization layer. This unit is repeated twice before the last output layer which is again a transpose convolution with output channels equal to 3.

2. The Colorization Model consists of the pretrained ResNet 18 encoder available in Pytorch. We use the first 6 layers of this model for feature extraction, i.e. the ResNet 18 encoder pretrained on ImageNet is used as the encoder block.
   Once the features are obtained from this, we construct the decoder block. It consists of a (3x3) convolution layer (filters=128) followed by Batchnorm and ReLU. This is then upsampled by a factor of 2 after which it is passed through 2 blocks of (3x3) convolution layers, Batch Normalisation and Relu activation. After upsampling again by a factor of 2, it is passed once through the same block mentioned in the previous sentence. The output layer is a convolution layer which is upsampled one last time to match input dimensions.

3. The UNet model is widely prevalent and used for a variety of applications. We use the UNet model as it is for this task.

4. The final model is the ResNetUNet model which is designed using inspiration from the ResNet and UNet architectures. We use the pretrained Resnet 18 model as the encoder and use this as our feature extractor. Further, these features are downsampled using blocks of convolution with relu activation which is concatenated with outputs from the

resnet18 model, i.e we introduce skip connections similar to the UNet architecture with the difference here being that the features are now added from the ResNet encoder block.
The decoder block is a simple UNet type decoder which upsamples these features to produce the required output.

Further details about the model architecture is present in out Jupyter Notebook.

Training Setting

We split our dataset into train, validation 70/30. And use random images for testing. We use the Adam optimizer with lr = 1e-2 and the MSE loss. We train each of these models for 50 epochs on Colab. Google Colab [NVIDIA Tesla K80 GPU] with 12 GB VRAM is the hardware used for training.

Post Processing and Metric

Once the predictions are obtained, we unnormalise the predicted image to plot and visualize the result. We use PSNR as an evaluation metric.

Scope of Improvement

Upon examination, we conclude that areas where we can potentially improve our performance by using a different Loss function, preferably the L1 loss function given the problem statement we are trying to solve. Since predicting and working with the RGB colorspace involves predicting 3 channel values separately, it makes it harder to solve this problem. So switching over to the LAB colorspace is another area we could explore. After a little reading, the GAN could also work well.

Conclusion

The detailed information is present in the code.