**IMAGE COLORIZATION USING CONVOLUTIONAL AUTOENCODER**

*Zahid Hossain*

**ABSTRACT**

The task of colorizing grayscale images takes tremendous skill and time. Professional colorists can take up to a month to colorize a photo. What if a machine learning model could do the task for us in mere seconds? This paper presents my exploration of training a neural network to perform the task of image colorization.

**1. INTRODUCTION**

There are endless ways to approach an ML colorization problem. In any case, one must prepare a dataset to train on, select or develop a model, fine tune parameters during training, and finally evaluate the results.

**2. SOURCE DATASET PROCESSING**

I chose to use my own personal images to create the dataset. It contains approximately 1500 images that comprise mostly of natural landscapes, but other scene types are mixed in. All the source images were first compressed using the open-source Caesium software to reduce their file size. This was done in an attempt to speed up any disk IO-bound operations. I then created a Python script to split, resize, and pad all the images. I split them at random such that 90% were used for training and 10% for validation. The images were resized to 2000x2000 squares with zero-padding. I took advantage of Python’s multiprocessing library to parallelize this task, achieving nearly 15x speedup when compared to a single-threaded for-loop implementation (1min vs. 15min).

A screenshot of a computer screen

Description automatically generated

Fig 1. Screenshot of portion of dataset

**3. MODEL SELECTION**

When researching models for colorization, I primarily saw the use of either convolutional autoencoders or GANs. I chose to use an autoencoder model because it seemed easier to train, but the results from GAN based models tended to look more realistic.

**3.1. Autoencoder Layers**

An autoencoder can be described with an encoder-decoder structure. The encoder compresses the input into a smaller latent space representation. The decoder will then upscale the latent space back to the same size as the input. For my model, the encoder was implemented using the first 6 layers of the ResNet-18 model [1]. The decoder was then implemented using the same type of convolutional layers but in reverse, with upsampling. This model design was inspired by Luke Melas-Kyriazi’s research [2]. His model used ResNet with 365 weights, but I changed it to use the default ResNet because I found it had very slightly improved color output.

**3.2. Color Space Choice (LAB vs. RGB)**

The model takes a 1-channel grayscale image as input and then has to output multiple channels to create a color image. Traditionally RGB is used, but this could be more unstable for the purpose of machine learning prediction [ref]. Using LAB colorspace for training, color information is stored in the A and B channels, and the L channel is equivalent to grayscale input. VERIFY THIS IS TRUE. That means we can train model to predict two channels instead of three, ideally increasing accuracy.

**4. PYTORCH DATALOADER PREPARATION**

Make custom class inheriting Dataset class to work with my local folder of images. Customized \_\_getitem\_\_ method to return separate tensors for L and AB channels used for training. Doing this so I can use PyTorch Dataloader.

**5. TRAINING**

Starting off by using 224px square images. Then moving up to 608px. Playing with learning rate starting at 0.1 and going down to 0.0001. Seeing improvement with smaller. Batch size of 15 (TRY DIFF SIZE?). Laptop has Nvidia RTX 3080 GPU. Average training time per epoch.

**6. RESULTS**

Show picturesssssss. Explain how dataset landscapey bias is no bueno. Doesn’t do too hot on skin. Also overall pretty desaturated, weird, meh.

**7. REFERENCES**

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep Residual Learning for Image Recognition”, 10 Dec. 2015, https://doi.org/10.48550/arXiv.1512.03385

[2] Melas-Kyriazi, Luke, “Image Colorization with Convolutional Neural Networks”, 15 May 2018, https://lukemelas.github.io/image-colorization.html