**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 PREPARATION**

Using my own images. Mostly nature-esque landscape stuff. Approximately 1400 images. Compress all using Caesium. Use Python script to split, resize, pad. Split into 90% training 10% validation. Resize to 2000px square with zero-padding. Script uses multiprocessing library to parallelize the task. Nearly 15x speedup compared to single-threaded simple for-loop.

**3. MODEL SELECTION**

Prioritizing what would seem like a simple yet effective model. Saw GANs, settled on this convolutional autoencoder? [Insert research paper reference]. [Insert medium blog post/GitHub reference]. VERIFY THIS IS ACTUALLY A CONVOLUTIONAL AUTOENCODER LOL. <Go into detail of the layers, size and stuff? Image?> Resnet for encoder [Resnet ref].

**3.1. Colorspace 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] Your mom et al.