# Colorizing Grayscale Images

**Loki Sassone**
Electrical Engineering
A14661319

**Sonya Mohammed**
Electrical Engineering
A15107726

## Abstract

In our project we created a Pytorch model to colorize grayscale images. We first used the Cifar-10 dataset to train our model and then later trained and tested the model on the Animals-10 data set which has larger images. We used eleven layers of 2-D convolution layers followed by ReLu and BatchNorm layers modelling our neural network architecture over Zhang's Colorful Image Colorization paper.

## 1 Related Work

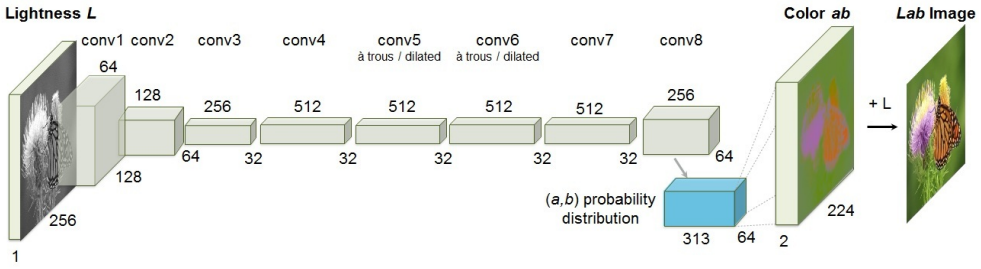
Image colorization is a popular deep learning method to colorize grayscale images. There have been many research papers and articles on this topic. For example Richard Zhang's Colorful Image Colorization paper uses a eight layer convolutional neural network to turn grayscale into RGB images. The most common use of image colorization is to turn old black and white images from the 19th and early 20th century to color images. DeOldify is another related work that uses deep learning to turn old images to more modern looking color images.

Figure 1: Original and colorized version of the "Migrant Mother", colorized by DeOldify

## 2 Problem Definition

The problem we are trying to solve is finding a way to recolor grayscale images to. The main issue in this is trying to find colors that are reasonable. The motivation behind solving this problem is being able to solve a task that humans cannot do well. Accurately recoloring a grayscale image is a difficult task for humans to do on their own yet algorithms can solve this problem much more accurately. We want to solve this problem and create vibrant colorizations that look realistic. In Figure 2 we can see Zhang's network that shows a solution implementation to the problem we are trying to solve.

Figure 2: Zhang's network example of the colorization issue




## 3 Approach

**Data:**

The Cifar-10 dataset was initially used for the problem. Cifar-10 provided a nice starting point as it contained small 32 x 32 images which ran faster on our model. However, the images were too small and compressed making it very difficult to visually verify what the image was and if it was colored correctly.

After running tests on the Cifar dataset, we switched to Animals-10 dataset as it provided larger images that were more perceivable and helped solve this issue. We cropped all of the images to be the same size to fit our current architecture model. Different square image sizes were input such as 50x50, 64x64, and 90x90 as higher pixel numbers crashed DataHub kernels yet we wanted a good size to allow images to be visually perceived better and lead to better, more accurate results.

**Model and Training:**

To solve this colorization problem we created a sequential model containing an altered version of Zhang's architecture. It uses eleven 2D convolution layers with ReLu layers and 2D batch normalization layers to help normalize the weights. The train function inputs our color images and manually converts the 3 RGB channels into 1 grayscale channel. It inputs each batch into the model and computes the loss for it as well. We utilized both L1 and L2 loss to test different loss functions in our model. Both Stochastic Gradient Descent and the Adam Optimizer were used to optimize our model during training.

## 4 Experiments and Ablation Study

- Architecture: We started our model with a simple eight 2D convolutional layer network with ReLu and a batch size of 64. We discovered our model had issues with normalization so we added a Batch Norm which improved our results accuracy. Adding bias into the convolutional layers also improved our results so we kept that in our final model. We experimented with both L1 and L2 loss and found similar results. For the optimizer we discovered that Adam performed significantly better than the Stochastic Gradient Descent so we shifted to this.
- Image Size: After shifting from Cifar-10 to Animals-10 dataset, we had larger images and more control over image size. We tried to shrink the animal images to size 128 x 128 and 90 x 90 however overtime we realized that these large sizes contributed to DataHub kernels crashing after every training. We shrank the images down to 64x64 and 50x50 to have a smaller size which made training significantly faster and as a result allowed longer training which better colorizations. The reduced image sizes however did impact the extent of colorization accuracy we could fully achieve.
- Data Amounts: We tested the model with different amounts of data and found that it performed much better when we had more images. It allowed colorizations to be more accurate than when we reduced the images.
- Batch Sizes: We tested different batch sizes but didn't see much of a difference. The lack of difference was likely due to epoch number being a larger factor on training than batch size. We decided to keep a batch size of 64.

- Epochs: More epochs drastically improved our model and if given more time and computational resources more epochs would be run to improve the results.
- Learning Rate: We tested two different learning rates of 0.01 and 0.001 however we discovered minimal differences so we didn't further test different learning rates and instead focused on different parameters.

## 5 Cifar-10 Results

Our model performed quite well on the Cifar-10 data as the model's colorized version of the airplane is close to the actual RGB version of the image. There is some duller and darker parts however overall it performed accurate colorization that would likely be improved with more epochs of model training. As we can see, the images in Figure 3 are highly pixelated and it is difficult to make out the image as an airplane. We chose this image as it was one of the easiest to make out even with Cifar's low resolution. However due to not being able to visually interpret what most of the images were, we decided to move on to a bigger and better dataset with improved resolution sizes.

One interesting result during our tests with the Cifar data was how the learning rate didn't affect the loss graph much other than the lower learning rate seemed to have loss that oscillated more. Both losses for each learning rate were also very similar, (0.25 final loss for LR=0.001 and 0.23 final loss for LR=0.01). The losses can be seen in Figure 4 below.

Figure 3: Grayscale (left), Predicted (middle), and Actual RGB (right) version of image

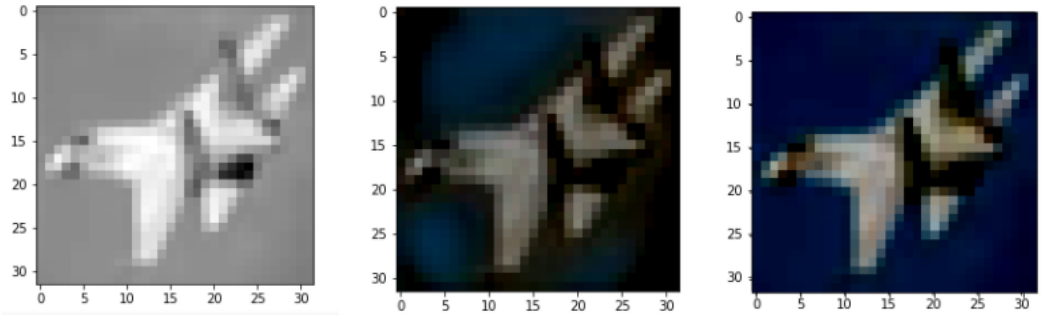
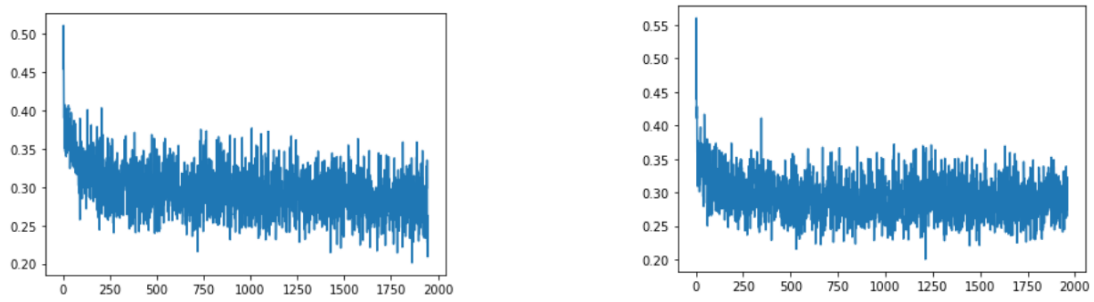


Figure 4: Graph of Loss over iterations LR=0.001 (left) and LR=0.01 (right)



## 6 Animals-10 Results

Our model performed visually better on the Animals-10 dataset due to less pixelation. The images still had to be reduced for computational resource purposes which caused some pixelation however the output quality is four times better than Cifar's when used with a pixel size of 64. Due to poor computing resources only a couple epochs were able to be performed.

In the examples seen in Figure 5 and Figure 6, we can see that the colorizations are quite accurate. The images are colored correctly although they are a bit duller in color. These results will definitely be improved with more epochs of training or by using the full resolution original Animals-10 images as input.

Figure 5: Original image of sheep in a field (left) and recolorized(right)

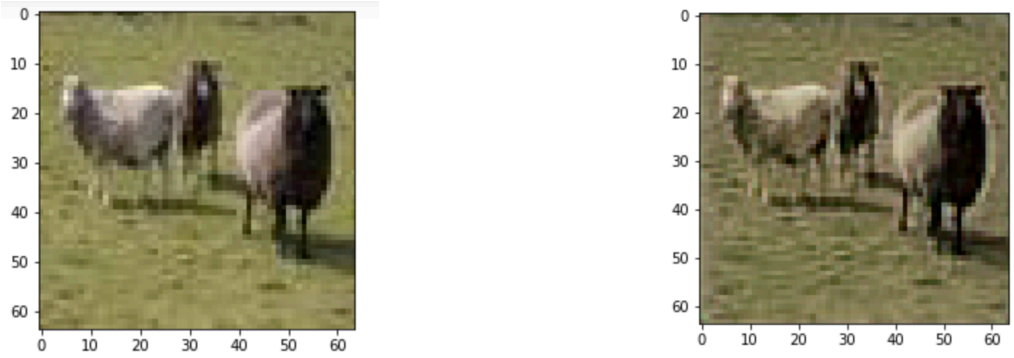
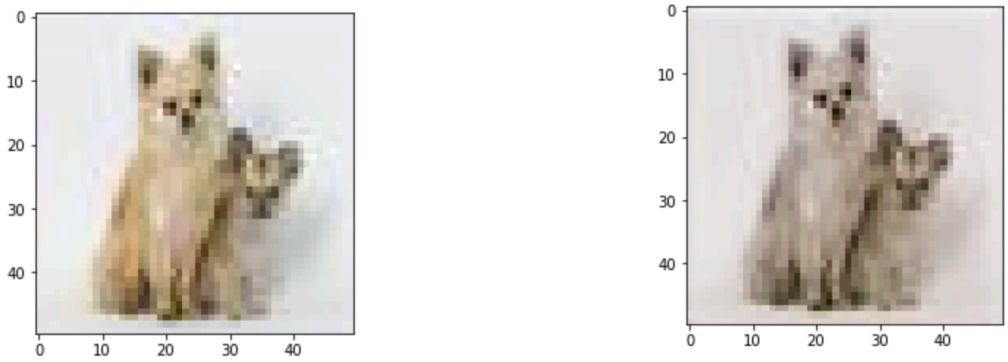


Figure 6: Original image of a dog (left) and recolorized(right)



## 7 Future Improvements and Conclusion

Some future improvements we could make is to run the training for more than just 2-5 epochs, such as 20 epochs. We weren't able to train our model on more epochs because DataHub's kernels crashed during training and we weren't able to see the results for higher number epochs. Due to DataHub's kernel issues we were also not able to properly test our model with a wider variety of hyper-parameters over time such as learning rate, loss function, and batch size.

Another future improvement we can make is to try perceptual loss instead of L2 or L1 loss. This is because perceptual loss can be used to compare high level differences, like content and style discrepancies, between images. Perceptual loss may lead to much more accurate results as it would allow lower loss if images are similar in color but the certain shade difference in color causes the loss to increase despite them being virtually the same. A combination of L2 loss and perceptual loss could also be used to tune accuracy.

Unfortunately, due to DataHub's lack of intensive resources, we lost important time since our kernels would crash after hours of training making the training futile. Due to this we were not able to get to

the aspect of Zhang's paper where he brightens the colorizations to reduce the dullness inherent in colorizing. Implementing this will help improve image colorization results.

Overall our model colorizes grayscale images decently with the current reduced image sizes and epochs. We were able to get strong accuracy vibrant images one night however the next days after DataHub drastically slowed down and our kernels would crash during training so we were not able to capture the rich colorizations we had previously produced with our model to display in this paper. With better resources we know our model will improve significantly and provide richer colorizations on higher quality images than we were able to obtain.

## References


CIFAR-10 and CIFAR-100 Datasets, www.cs.toronto.edu/ kriz/cifar.html.

Jantic. Jantic/DeOldify. GitHub, github.com/jantic/DeOldify.

Saxena, Ashish. Animal Image Dataset(DOG, CAT and PANDA). Kaggle, 26 May 2019,

Zhang, Richard, et al. Colorful Image Colorization. 28 Mar. 2016.