FilmColorizer - Final Report

Varun Pai and Dean Yu

Word Count: 1774

Permissions:

	Permission to post video	Permission to post final report	Permission to post source code
Varun Pai	Wait	Yes	Yes
Dean Yu	Wait	Yes	Yes

Introduction

The goal of this project is to produce an image colorizer using neural networks and ultimately colorize the black and white movie Casablanca. Several implementations of colorizers have been produced in the past using various neural network architectures, such as auto-encoders and GANs, hence neural networks are a viable option to solve this problem. This makes for an interesting task because of the potential to generalize this technique to many other images and media aside from the movie Casablanca. Vintage photos and many other media can be brought to life with our creation. Another equally important aspect of our creation is the fact that we are improving our knowledge in machine learning, and will hopefully be able to make a meaningful contribution to the field one day.

Illustration/Figure

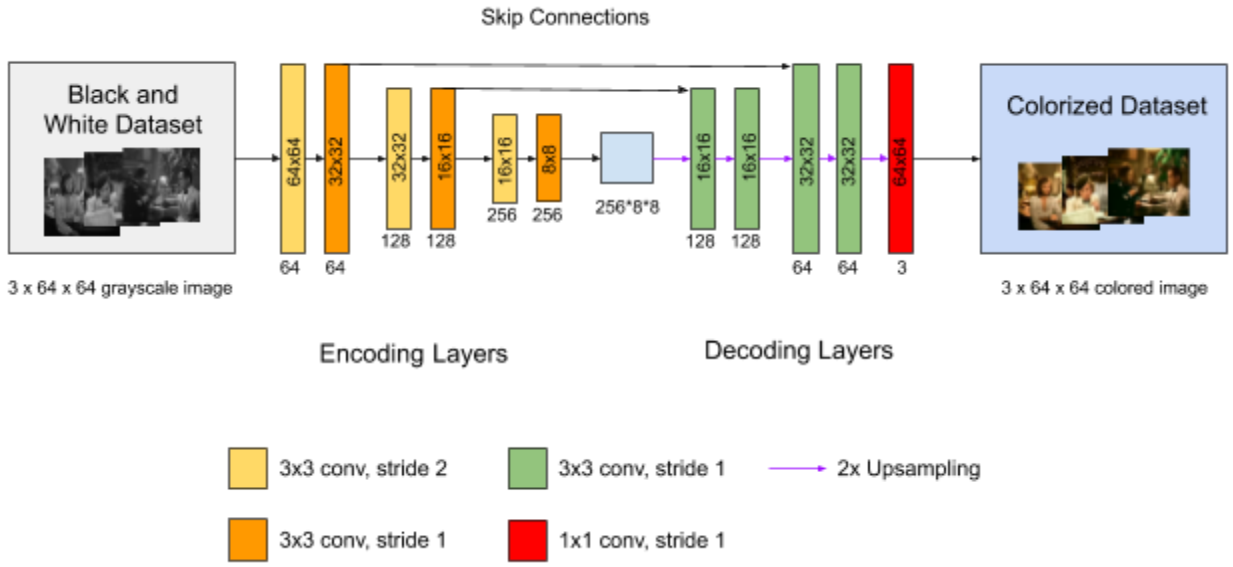


Figure 1: A figure illustrating the final architecture of the FilmColorizer CNN.

Background and Related Work

There has been much previous work in the field in the form of studies as well as existing commercial software. Examples of this work would include: two students in Stanford's CS216 course who used a deep convolutional neural network for colorization, by framing the task as a classification problem [1]; as well as a team from the University of Ontario Institute of Technology, who explore the applications of deep convolutional generative adversarial networks for image colorization alongside the U-Net architecture[2]. On the commercial side, an example of a widely available software in this field is ColorSurprise AI by PixBin. This is a model that has been trained on millions of pictures to accurately colorize images and determine whether the image input is a living object or not.

Data Source, Labeling and Processing

FilmColorizer uses two datasets in the training process: the original version of the black and white film "Casablanca" [3] serves as the data and a version of the film that was hand colorized by Turner Entertainment [4] serves as the labels. Both versions of the film (in mp4 format) are edited using Blender [5] so that they both had a video resolution of 64x64 and the frames were synced between versions.

A resolution of 64x64 was chosen to minimize the computation power required for the Neural Network and because we were already familiar with working with images of this resolution from Assignment 4. To sync the frames between versions, a manually calculated number of frames were removed from both versions, as they had slightly different intros. Then, they were both adjusted to use a frame rate of 24 frames per second.

Figure 2: A figure containing example images from both datasets.

After both video feeds were exported by Blender, the scene filter feature of the VLC media player [6] was used to extract stills of the frames from them at a rate of 1 frame per every 24. This process resulted in around 5,000 images from each version of the movie.

In Pytorch, after the datasets are loaded, images from both datasets have their pixel values adjusted to a range [-1,1]. The datasets are split between the train, validation, and test sets chronologically. The split was performed this way to minimize the possibility of overlap between the three datasets, where frames from the same or similar scenes end up in multiple datasets. The first 4000 frames sequentially were allocated to the train set, the following 1000 frames were given to the validation set, whilst the remaining frames were incorporated into the test set.

Architecture

For the architecture of our final model, we used a version of U-Net [7]. U-Net is both shaped similarly to, and behaves comparably to, an auto-enconder. U-Net consists of a series of encoding layers designed to compress the information present in the input to a smaller dimension, followed by a series of decoding layers. However, instead of targeting the original data, the decoding layer produces an output which comprises of k feature maps, each the size of the original image, showing how much each pixel corresponds to each class. For this specific application, k equals 3 and the three classes are the red, green, and blue colour channels. To ensure no information is lost, U-Net also contains skip connections, which concatenate the output of the corresponding encoding layer onto the input of a decoding layer.

The encoding layers comprises 3 sets of double convolutions. Each double convolution contains 2 convolution layers, each with batch normalization and ReLU activation on the output. Both convolution layers also have the same number of 3 by 3 kernels and a padding of 1. The first layer uses a stride of 2, to achieve the function of a max pooling layer and to reduce the computation required, while the second layer has a stride of 1. The first set of double convolutions uses 64 kernels, the second set uses 128 kernels, while the final set of double convolutions uses 256 kernels.

The decoding layers are made up of 3 upsampling layers, the first two of which are followed by double convolutions. These double convolution layers use a stride of 1, a padding of 1 and consist of 128 and 64, 3x3 kernels respectively. The final upsampling layer is followed by a convolution layer with 3 1x1 kernels used to generate a 3x64x64 rgb image as the output. To compute the loss and gradient of this model, a mean squared error loss function is used between the output image and the corresponding image from the colorized dataset which serves as a label.

Baseline Model

The baseline used for this project is a simple model that outputs the black and white dataset verbatim. This model was chosen over a neural network approach for the baseline as the final model is already one of the simplest neural network architectures conventionally used for image colorization. A traditional digital colorization technique was not used as the baseline since they are generally proprietary and thus not well documented on the internet; and also involve some degree of manual work.

As the baseline is so primitive, it is treated as a "Naive Baseline," similarly to only predicting 1 class for any input in a binary classifier, in that it is only used to compare model metrics (mean squared error loss in this case). Additionally, the final model should outperform the baseline in said metrics by a significant amount to be considered a viable solution to the film colorization challenge.

Quantitative Results

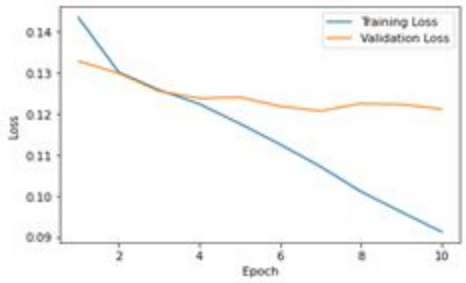


Figure 3: The loss graph of our final model

The main measurement used to quantify how well our model performed was how our loss function was performing throughout the training, validation and test epochs. We observed that as the epochs grew, the loss was trending downward. Our loss was determined by comparing how well the final model took in a grayscale image and produced a color image. This color image was then compared to the actual hand colored image, which in this case would be our label. The loss quantity was then computed by comparing how close the colors of each pixel were to the label image. For a task like this, this seemed like the most

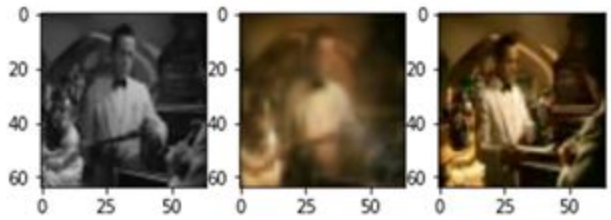
logical measure of performance. The results turned out quite well optimizing this loss function, as will be seen in the next section.

Model	Training Loss (MSE)	Validation Loss (MSE)	Testing Loss (MSE)
Naïve Baseline	0.1773	0.1783	0.2842
U-Net	0.0912	0.1212	0.1780

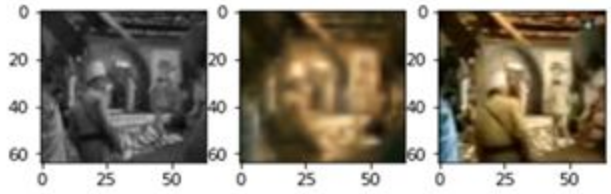
Figure 4: Table showing the training, validation and testing loss of our baseline and final model.

Looking at the specific loss values, it can be seen that the final model loss is significantly lower than the baseline in all cases. There is a bizarre quirk where the testing loss in both cases is relatively high compared to the other 2, however, this may be a result of using the end of the film and a low number of frames for that dataset, making it not representative of the film the way the other 2 datasets are. The comparatively lower loss in the final model indicates that it has generally succeeded at reaching the goals of the project.

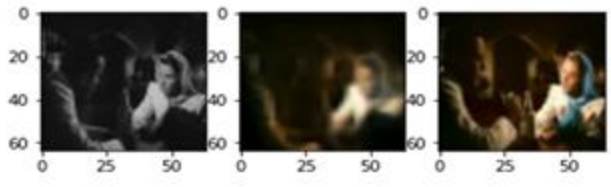
Qualitative Results



Training



Testing/Validation



Testing/Validation

Figure 5: Sample outputs

Discussion and Learning

From the images shown above, it can be seen that the general colors are correct when comparing the outputs to the labels. They are however blurrier and more muted, especially when it comes to the validation and test images, this is an interesting feature of our results. We can however see that the task of colorization is being accomplished. Overall, the results we obtained were good. However we did learn some key lessons, one of the most important being that overly complex models are not necessarily better, especially when there are significant time constraints. Originally we were going to use a GAN to perform our task; however we ran into significant trouble when we realized that the discriminator was consistently outperforming the generator, and thus the generator did not learn much. This would lead to very poor results such as the image shown below.

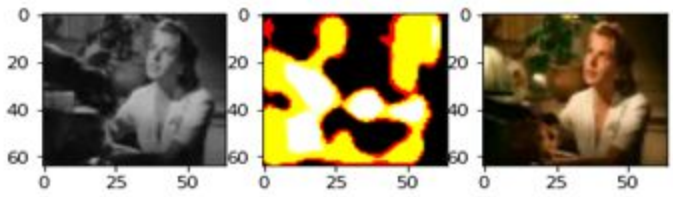


Figure 6: A GAN output

This was an issue we were not able to overcome, thus we had to make last minute model changes, and ultimately use our generator as our model. Fortunately it was an auto encoder and was able to complete our task. This showed us that a much simpler architecture could lead to much better results, as was just demonstrated. Had we been given more time, and had more experience with GANs, our results likely would've been better; however in this case simplicity was optimal. If we were to do another similar project, we would likely use an auto-encoder and put all our time into it, rather than making a last minute switch. The results from GANs can be very good; however it is a gamble to use them because training can be difficult.

Ethical Framework

There are a considerable amount of stakeholders in this project. The most important ones include our team, our professors, our future employers, and future prospective Machine Intelligence students. Through the lens of reflexive principlism applied to our team, we can see that justice applies. Justice applies because we are expected to avoid plagiarizing and be honorable students. In regards to our professor, the three principles of justice, beneficence, and nonmaleficence apply. As a professor he has a duty to uphold the rules and ensure students engage in a dignified manner, and ensure justice is served when necessary. He must also provide benefits for his students to the best of his ability; this can be in the form of fair deadlines and term work. Lastly, he should always look to avoid harm being caused to his students, thus nonmaleficence applies. In regards to our future employers, we can see that nonmaleficence applies because that must ensure that someone employed by them must have a safe environment. Lastly when it comes to future MI students, we can see that the same principles that apply to our team apply to them because they will be in this position one day, thus justice applies to them.

[1] “Image Colorization with Deep Convolutional Neural Networks.” [Online]. Available: http://cs231n.stanford.edu/reports/2016/pdfs/219_Report.pdf.

[2]K. Nazeri, E. Ng and M. Ebrahimi, "Image Colorization Using Generative Adversarial Networks", *Articulated Motion and Deformable Objects*, pp. 85-94, 2018. Available: 10.1007/978-3-319-94544-6_9 [Accessed 7 December 2020].

[3] Curtiz, M., Bogart, H., Bergman, I., & Rains, C. (1942). “Casablanca.” S.l.: Warner Bros.

[4] Curtiz, M., Bogart, H., Bergman, I., & Rains, C. (1988). “Casablanca.” S.l.: Turner Entertainment.

[5] T. Roosendaal, *Blender*. Blender Foundation, 2020.

[6] *VLC media player*. VideoLAN, 2020.

[7] Ronneberger, O., Fischer, P. and Brox, T., 2020. “U-Net: Convolutional Networks For Biomedical Image Segmentation.” [Online] Available : https://arxiv.org/abs/1505.04597