Inverting Film Negatives Using Multi-Layer Perceptrons

Kurt Gu kg10@williams.edu
Computer Science & Philosophy

1. Introduction

Film photography is seeing a resurgence in popularity in recent years. Users are taking analog pictures but what they share/publish are the digital versions of these pictures. However, existing mainstream hardware for digitizing film, such as scanners made by Epson, Noritsu, Fujifilm, and Imacon, have not been updated in decades, and are prohibitively expensive (a Nikon 9000ED scanner made 20 years ago now can sell for $3000, and does not even support Windows 7). The emergent alternative to digitizing film using scanners, is to use digital cameras in combination with a macro lens to take pictures of film directly. The advantages of this approach are clear: the hardware is more affordable, and digitizing a frame takes one second instead of several minutes with a scanner. However, since common print film (also known as color negative film) presents each frame in negative colors when developed, this approach has the inherent complication where users are required to invert their scans manually, which is time consuming and often inaccurate or inconsistent. This project explores a machine learning approach to automate the color film inversion problem, by training a neural network to learn the mapping of colors between the original negative and the inverted final prints.

2. Preliminaries

This project builds its model using a multi-layer perceptron (MLP), also known as a neural network, to learn the mapping function between the film negative and the inverted print. An MLP, somewhat obviously, is made of layers of perceptrons, each of which is a linear regression predictor of all the perceptron nodes in the previous layer.(See figure 1)

Therefore to understand how MLPs learn, we first need to understand linear regression. Linear regression produces predictions for an outcome variable Y as a linear combination of learned parameters θ and input features X. That is, for a given example, we have the following equation that determines how we obtain a prediction $\widehat{y}$ as a function of inputs $x_1, \dots, x_d$,

$$\widehat{y} = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_3 x_d.$$

The parameters θ can be learned by minimizing a suitable loss function L using gradient descent. I use the mean squared error, which is defined for n samples of data as follows: $L(\theta) \equiv \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$. To avoid overfitting, it is common to apply a regularization penalty in conjunction with the loss: I use the L2 penalty. So, the overall function being minimized

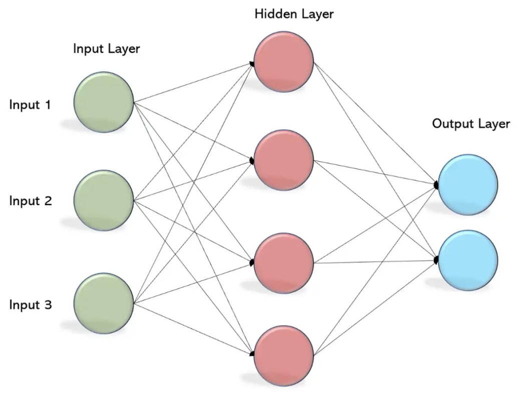

Figure 1: MLP, or multilayer perceptron

is then $L(\theta) + \lambda \sum_{j=1}^{d} \theta_d^2$, where λ is a hyperparameter that will be tuned using a validation set.

An MLP is simply a combination of layers of such linear models, which add up to produce a non-linear model. From domain knowledge in film photography, we know that the mapping is also non-linear between film negatives and the print paper for which they were originally designed. This indicates that it is reasonable to apply an MLP model to this problem.

3. Data

For this project, I'm creating my own dataset using my personal archive of film negatives taken over the years. The input matrix is the matrix of RGB values in each of the film negative scans. The outcome matrix is intended to be an RGB value matrix of scans of prints made using the aforementioned negatives. However, the supplies needed to create the outcome dataset was delayed in shipping, so for this draft I'm instead using inversions created manually in photoshop as a temporary substitute. The one advantage to this approach is that there exists a pixel-to-pixel corespondence of values between my input and output. The downside and potential issue is that instead of learning a stable mapping function that exists in the photographic paper, the model is instead learning my film inversion technique, and I, being human, am notoriously inconsistent! So for this draft dataset I tried to only apply minimal and relatively consistent color-grading to the outcome images.

In the project repository [link here] you will find a folder named "Dataset", with two folders inside labeled "Original_resized" and "Inverted_resized". Each original image, as it was scanned and cropped, is inconsistent in dimensions and each had a resolution of between 24 to 90 megapixels. A workstation might be able to handle these high-resolution images but not my laptop. So I've downsized all images to 256x256 pixels, and renamed them for easier indexing. I plan to make and add the actual paper scans in the upcoming week, likely in a higher resolution.

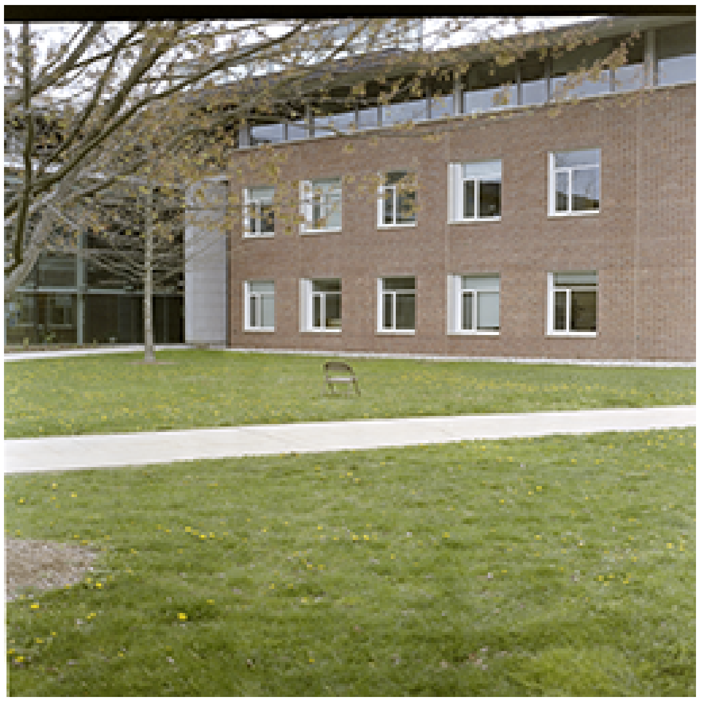
Figure 2: An inverted sample from a dataset

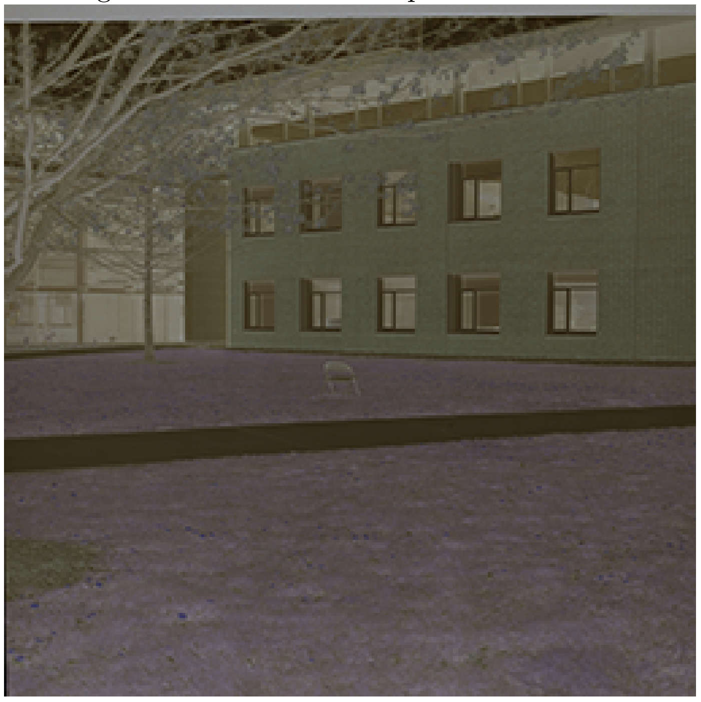
Figure 3: The original camera scan of the same image

4. Training And Validation Of Models

As of right now, I trained an MLP with 64x64 hidden layers using the MLPRegressor in sklearn. I use the RGB values in each original pixel as input, and the RGB values in each inverted pixel as output. So the overall network dimension is 3x64x64x64. For the final model, I'm looking into GPU-accelerated solutions so I can train larger networks with higher resolution samples.

So far I've created 33 images (fewer than I hoped because of time constraints). Due to the small numder of samples (although one can argue that each image contains $256^2 = 65536$ datapoints), I dedicate 31 images to the training set, one image as the validation, and one image as the test.

5. Results

So far, the best R^2 result of 0.833 I've obtained on the validation image is from the 3x64x64x64 network. I have attached on the next page the result produced by using this model to generate a prediction on the test image:

6. Ablation Study

I have yet to perform an ablation study

7. Discussion and Conclusion

So far I'm observing promising results from a preliminary and arguably primitive model, but as we can tell there are still noticible artifacts in the generated output. For the final model, I plan to explore with different models, higher resolution images, and a larger and more representative dataset.

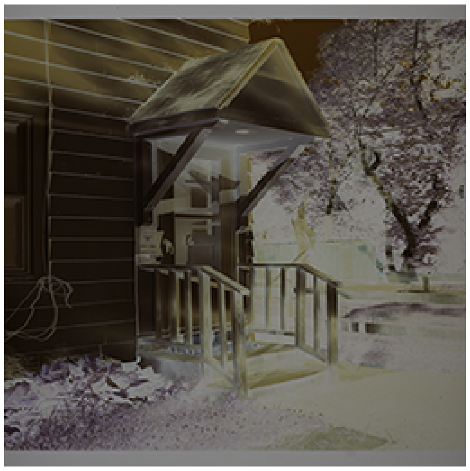

Figure 4: The test image

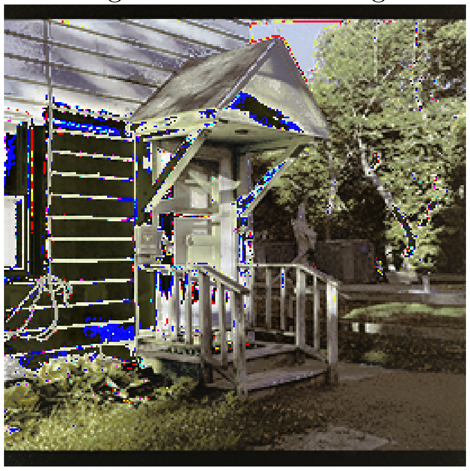

Figure 5: Result generated by the current network

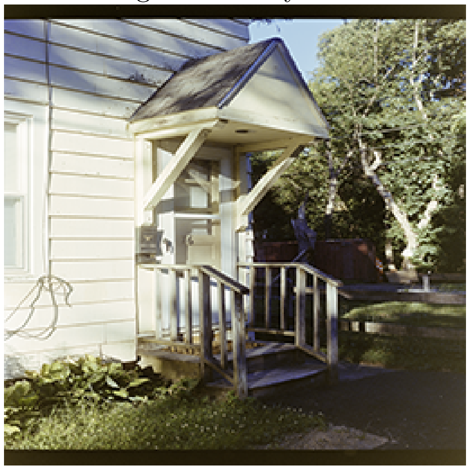

Figure 6: Manual inversion of the same negative

References